Recognising the Fine-Grained Actions of a Goal-Directed Activity From Multi-Modal Images

by

Seán Bruton, B.A. (mod.), M.Sc.

Thesis

Presented to the

Trinity College Dublin, the University of Dublin

in fulfillment

of the requirements

for the Degree of

Doctor of Philosophy

Trinity College Dublin, the University of Dublin

January 2021

To Rhiannon

Acknowledgments

I would like to thank my supervisor for all of his guidance and encouragement throughout the course of this work. I would also like to thank my colleagues, friends and family, especially my parents, for their unending support and patience during my studies. I would like to give a special thanks to Dr. Marie Morris of Royal College of Surgeons Ireland, who provided motivating problems for this research and generous guidance and help in the collection of data. Finally, I would like to thank my fiancé, Rhiannon, who kindly provided help and support in too many ways to list here.

Seán Bruton

Trinity College Dublin, the University of Dublin
January 2021

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Seán Bruton

March 1, 2021

Recognising the Fine-Grained Actions of a Goal-Directed Activity From Multi-Modal Images

Publication No. _____

Seán Bruton

Trinity College Dublin, the University of Dublin, 2021

Supervisor: Dr. Gerard Lacey

The ability to understand and respond to human activities can form the basis of many pervasive computing applications. Recognising the constituent actions of an activity can lead to a more detailed understanding of the activity and provide opportunities to develop applications for monitoring, training and assistance. We address the specific problem of recognising the fine-grained actions of a fixed-setting goal-directed activity from RGB-D videos. Our research is motivated by the specific case of recognising the fine-grained actions involved in clinical skills.

We design a novel convolutional neural network architecture, WeaveNet, for fine-grained action recognition from multiple image types. A spatio-temporal fusion method, Densely-Fused Action Images, is also presented for use in combination with WeaveNet. This combined architecture achieves an accuracy of 82.7% at a mid-level granularity on a benchmark dataset, an improvement of 9% over existing methods.

We contribute a system for recording fine-grained actions involved in human-object interaction tasks, specifically including clinical skills. The system is novel due to its ability to record actions from multiple viewpoints using RGB-D cameras in a synchronised way.

We present a dataset of clinical skill performances for the skill of venepuncture, including 60 performances, across 20 subjects, totalling over 15 hours of footage. The multi-modal, multi-camera characteristics of this dataset make it amenable to many fine-grained action recognition techniques.

Together, the fine-grained action recognition technique, the system for recoding human-object interactions, and the dataset of clinical skill performances, make a significant contribution towards the development of next generation pervasive computing applications.

Publications

Below is the list of publications to date directly related to this thesis.

S. Bruton and G. Lacey, "Recognising Fine-Grained Actions by Combining Colour, Depth and Flow", in *Irish Machine Vision and Image Processing Conference Proceedings*, pp. 28-35, 2017.

S. Bruton and G. Lacey, "Recognising Actions for Instructional Training using Pose Information: A Comparative Evaluation," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pp. 482-489, 2019.

Contents

Acknowledgments iii

Abstract v

Publications vii

Chapter 1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Research Focus . . . 3
1.3.1 Data Collection Pipeline . . . 3
1.3.2 Multi-modal Data . . . 3
1.3.3 Fine-grained Action Recognition . . . 4
1.4 Contributions . . . 5
1.5 Structure of Thesis . . . 6

Chapter 2 Background 8

2.1 Artificial Neural Networks . . . 8
2.1.1 Network Training . . . 8
2.1.2 Recurrent Neural Networks . . . 9
2.1.3 Convolutional Neural Networks . . . 11
2.2 Action Recognition . . . 12
2.2.1 Feature-Based Approaches . . . 13
2.2.2 Deep Learning Approaches . . . 15
2.2.3 RGB-D Action Recognition . . . 24
2.2.4 Recognition for Situational Support . . . 28
2.2.5 Fine-grained Action Recognition . . . 29
2.2.6 Relevant Datasets . . . 35
2.3 Vision for Skill Determination . . . 37
2.4 Summary . . . 38

Chapter 3 How do you take yours? Action Recognition Using 3D Tracked Poses and Recurrent Neural Networks 39
3.1 Introduction . . . 39
3.1.1 Motivation . . . 39
3.1.2 Contributions . . . 41
3.2 Design . . . 42
3.2.1 Data Acquisition . . . 42
3.2.2 Point Cloud Processing . . . 44
3.2.3 Pose Estimation . . . 45
3.2.4 Action Classification . . . 48
3.3 Implementation . . . 50
3.3.1 Data Acquisition . . . 50
3.3.2 Point Cloud Processing . . . 51
3.3.3 Pose Estimation . . . 52
3.3.4 Action Classification . . . 53
3.4 Evaluation . . . 57
3.4.1 Data Acquisition . . . 57
3.4.2 Point Cloud Processing . . . 58
3.4.3 Pose Estimation . . . 59
3.4.4 Action Classification . . . 62
3.5 Discussion . . . 64

Chapter 4 Fusion of Colour, Depth and Scene Flow for Fine-Grained Action Recognition 66
4.1 Introduction . . . 66
4.1.1 Motivation . . . 66
4.1.2 Contributions . . . 68
4.2 Design . . . 69
4.2.1 Deep Learning Architecture . . . 70
4.3 Implementation . . . 74
4.3.1 Dataset . . . 74
4.3.2 Classifier Training . . . 76
4.4 Evaluation . . . 77
4.4.1 Feature Fusion . . . 77
4.4.2 Temporal Fusion . . . 81
4.4.3 Method Comparison . . . 83
4.5 Discussion . . . 84

Chapter 5 WeaveNet – Fine-Grained Action Recognition by Dense Fusion of Image Types 86
5.1 Introduction . . . 86

5.1.1 Motivation . . . 86
5.1.2 Contributions . . . 88
5.2 Design . . . 89
5.2.1 Base Architecture . . . 89
5.2.2 WeaveNet . . . 92
5.2.3 Stranded Late Fusion . . . 97
5.2.4 Densely-Fused Action Images . . . 99
5.3 Implementation . . . 101
5.3.1 Datasets . . . 101
5.3.2 Classifier Training . . . 106
5.4 Evaluation . . . 109
5.4.1 Model Ablation . . . 109
5.4.2 Granularity Evaluation . . . 109
5.4.3 Comparison to Published Works . . . 110
5.4.4 Computational Analysis . . . 111
5.4.5 WeaveNet Fusion Analysis . . . 111
5.4.6 Densely-Fused Action Images Analysis . . . 113
5.4.7 OSCE-V Performance . . . 118
5.4.8 Qualitative Evaluation . . . 118
5.5 Discussion . . . 120

Chapter 6 Conclusions and Future Work 122
6.1 Conclusions . . . 122
6.2 Future Work . . . 124

Appendix A Recording Framework Candidate Camera Arrangements 126

Appendix B Multiple RGB-D Camera Calibration 130

Appendix C Point Cloud Processing Software Pipeline 135

Appendix D Cup Of Tea: Dataset Labelling Rules 138

Appendix E Tree Parzen Estimator Hyperparameter Selection 139

Appendix F Cup Of Tea: Action Classifier Hyperparameter Searches 140

Appendix G Cup Of Tea: Improving Data Quality 144

Appendix H Gated Recurrent Unit Temporal Fusion: Fifty Salads Test Split Evaluation 149

Appendix I OSCE-V Dataset Design **151**
I.1 OSCE Selection 151
I.2 OSCE Design 152
I.3 Subject Selection 153
I.4 Subject Briefing and Instructions 154
I.5 Recording Method 154
I.6 Action Labels 154
I.7 Evaluational Method 155
I.8 Dataset Uses 156

Appendix J WeaveNet: Late Fusion Parameter Study **158**

Appendix K OSCE-V Dataset Collection Details **160**
K.1 OSCE Steps - Venepuncture 160
K.2 Participant Information Sheet 162
K.3 OSCE-V Action Labels 164

Bibliography **169**

Chapter 1

Introduction

1.1 Motivation

The ability to understand and respond to human activities can form the basis of many pervasive computing applications. The activities in domains such as sports, music or medicine often involve the manipulation of objects in a specific manner to achieve a goal. One of the challenges of building next generation pervasive computing applications in these domains is understanding these human-object interactions, which has been a long-standing research problem in computer vision [1, 2, 3]. Recognising the constituent actions of a human-object interaction can lead to a more detailed understanding of these interactions and hence provide greater opportunities to develop applications for purposes such as monitoring, training and assistance.

Understanding human-object interactions would have possible applications in the medical domain. Clinical skills are human-object interactions involving dextrous manipulation of objects to achieve a strict goal. These skills are commonly performed in fixed repeatable settings. An example of a clinical skill is peripheral intravenous cannulation (PIVC). This is performed to facilitate the administering of fluids and medicines to a patient's bloodstream, by inserting a special needle, known as a cannula, into a vein on the patient, as shown in Figure 1.1. Objective Structured Clinical Examinations (OSCEs) are a formalised method of assessing clinical skills of medical students [4]. OSCEs were devised to improve validity and reliability of such assessments via the introduction of structured questioning and examination criteria [5]. In an OSCE for the PIVC skill, the student is assessed on their ability to correctly insert the needle into the vein on a simulated patient. One examination criteria shared across all OSCE skills is that the individual actions are performed in the correct order. For example, PIVC involves the necessary step of cleaning the cannulation site (the area of skin where the cannula needle is inserted) with an anti-bacterial swab. If this step is not performed before the insertion of the cannula needle, there is an increased risk of infection to the patient [6]. A student would fail the OSCE exam if they omit this step. Thus, recognition of the individual steps, or actions, could be used as the basis

Figure 1.1: A screen-shot of a person performing the OCSE task of Peripheral Intravenous Cannulation on a simulated patient vein. Image copyright Trinity College Dublin.

of a system that validates whether a student followed the correct order of steps.

The individual actions that are performed in an OSCE exam can be described at multiple levels of granularity. At a fine granularity, the constituent actions are bi-manual interactions with multiple objects, such as "insert needle into site" for PIVC. They may involve small, subtle motion patterns that may be similar across actions. Action recognition methods must understand the variations and validity of the different ways in which an action is performed, as well as reliably distinguish one action from another. A high level of recognition of OSCE actions would prove useful to applications, such as an expert supervisor system, that could use accurate recognition to provide appropriate prompts to a learner during practice.

There are other application scenarios that may benefit from accurate fine-grained action recognition during human-object interactions. In particular, understanding human-object interactions has applications in the area of collaborative robotics [7]. In a fixed setting, such as a manufacturing line, a robot could understand the intent of a person's manipulation of an object and assist in or repeat the desired task. Assistance in Activities of Daily Living (ADL) is another potential application area. Cameras focussed on a fixed setting, such as kitchen worktop, could enable applications to understand that a particular meal is being prepared and provide assistance by displaying a recipe or pre-heating an oven. By being able to identify the current action being performed, such a system would be able to prompt the person to perform the next action in an online manner, i.e. during rather than after a performance.

1.2 Problem Statement

Recognising the fine-grained actions involved in human-object interactions, such as OSCEs, can form the basis of many pervasive computing applications. Key to the feasibility of providing accurate online feedback in such applications is reliable rates of recognition. Based on this reason, our research addresses the problem of how best

to **recognise the fine-grained actions of a fixed-setting goal-directed activity from video data**.

1.3 Research Focus

The contributions in this thesis can be broadly categorised under the following topics.

1.3.1 Data Collection Pipeline

A system for capturing human-object interaction tasks, specifically involving fine-grained actions is contributed. To reliably recognise the actions involved in the task, a careful examination of the requirements of the system design is performed. We are focussed on recognition for fixed-setting activities, such as those that can be performed on a table, as in the case of OSCEs. This permits flexibility in design decisions regarding the number and type of sensors used. The ability to capture actions from multiple viewpoints is beneficial to possible applications. One benefit of such an arrangement is that it may combat undesirable occlusions of salient regions that may occur during certain actions. This mirrors the scenario of human-observed OSCEs, where the examiner is free to move about a task area to gain a better vantage point of key moments of the task. Another design decision is to utilise multi-modal cameras (i.e. colour and depth, or RGB-D). The use of such RGB-D cameras allows flexibility in possible approaches to the problem of fine-grained action recognition. The added complexity of using multiple RGB-D cameras requires careful design of the components of the collection pipeline. To maximise the utility of the data collected, techniques for the registration and synchronisation of the data received from the cameras are devised, implemented and evaluated. The data collection pipeline is demonstrated for use in capturing two different human-object interaction tasks in this thesis, however it is not restricted to these tasks and can be used to capture other tasks that involve fine-grained actions in a similar setting.

1.3.2 Multi-modal Data

With the advent of consumer RGB-D cameras, such as Microsoft Kinect, there have been many uses found for the extra modality afforded by these cameras. Such uses include estimating poses of humans [8], hands [9] and objects [10]. Such pose information is salient for the purposes of action recognition [11, 12]. In this thesis, we use sequences of pose information estimated from RGB-D data to perform action recognition. This approach serves to validate the usefulness of the data collection pipeline for capturing the actions of a human-object interaction task.

It is possible to spatially align, or register, the colour and depth images received from an RGB-D camera. In this thesis, we devise convolutional neural network (CNN)

approaches to action recognition that make use of these spatially-aligned image modalities. There exist other approaches in the literature that make use of colour and depth data to perform action recognition [13], however CNN approaches commonly treat the modalities distinctly, with a separate CNN branch for each image modality [14, 15]. Such approaches do not explicitly take advantage of the property of spatial alignment of the modalities. However, based on the observation that there exists latent structures across the modalities [13], we explore techniques to learn these latent structures. To do so, we examine approaches to early, mid and late fusion of features of spatially aligned image modalities in a CNN architecture. We demonstrate an action recognition approach where early fusion outperforms late fusion, and thus benefits from multi-modal data without the requirement of expensive parallel branches.

Many action recognition approaches benefit from the use of motion information (i.e. optical flow) in addition to spatial information (i.e. colour) [16, 17]. In this thesis, we explore the alternative use of scene flow as a motion representation. The availability of depth information allows accurate generation of this scene flow information [18]. Compared to optical flow, scene flow is three dimensional, as opposed to two, and is defined in word coordinates, as opposed to pixel coordinates. This richer motion representation afforded by multi-modal data is used in our action recognition techniques.

1.3.3 Fine-grained Action Recognition

In this thesis, we contribute fine-grained action recognition techniques. Recognition at a fine granularity allows identification of steps of a human-object interaction task, such as "insert needle into site" for the PIVC OSCE task. This is a challenging problem due to subtle differences between actions and multiple valid ways in which actions can be performed. Another challenge is that there are multiple valid arrangements of the context in which actions are being performed. For example, the presence of tomatoes on a chopping board for a meal preparation task does not necessarily mean that the current action is "chopping tomatoes". Fine-grained action recognition approaches must overcome these challenges to provide a reliable accuracy level.

The problem of understanding actions performed as part of a human-object interaction task is often broken down into two stages [19, 20]. The first stage involves the fine-grained recognition of actions from frames or short clips. The second stage involves the segmentation of an entire recording into action segments. Segmentation approaches may require the observation of the entire recording [20] or, more strongly, an ordered sequence of the actions performed [21]. Therefore, these segmental approaches are not well-suited for use in online applications that provide prompts or guidance to a person during a task. We focus on improving performance of recognition from frames or short clips as this method is well-suited for use in such online applications. Another reason that we focus on recognition is that in two stage approaches, segmentation performance is dependent upon the accuracy of the recognition stage [19], and thus it

warrants specific attention.

In this thesis, we propose novel approaches to utilising multi-modal information to improve fine-grained action recognition performance. Specifically, we take advantage of spatially aligned image modalities, such as colour and depth, through the use of novel CNN fusion strategies. We utilise dense connections [22] in a CNN architecture, which we call WeaveNet, to facilitate fusion to be performed throughout a network. In our ablation studies, this approach is shown to increase fine-grained action recognition performance compared to baseline approaches.

1.4 Contributions

The three major contributions of this thesis are:

WeaveNet We design a novel CNN architecture, WeaveNet, for fine-grained action recognition from multiple image types, such as colour and depth. We demonstrate that it is capable of learning efficiently from these multiple image types, and show that it scales efficiently with additional image types. We propose a method to allow the network to learn from spatio-temporal information, Densely-Fused Action Images, which improves performance further. This combined network achieves an accuracy of 82.7% at a mid-level granularity on a benchmark dataset, an improvement of 9% over existing methods.

Multi-camera RGB-D Capture Framework We contribute a system for recording performances of human-object interaction tasks. The system's novelty lies in the fact that it enables recording of fine-grained actions from multiple viewpoints using RGB-D data, in a synchronised way. The system is thoroughly evaluated and is validated for action recognition purposes using object and arm pose information for a collected dataset. A code framework is also contributed to allow other researchers to feasibly re-implement this framework and to devise efficient data processing pipelines using the framework.

OSCE-V We collect a dataset of OSCE skill performances for the skill of venepuncture, including 60 performances recorded using 20 subjects, totalling over 15 hours of footage. A dataset of goal-directed tasks in the form of OSCEs does not currently exist. The dataset is recorded with RGB-D cameras from multiple viewpoints. There does not exist another fine-grained action recognition dataset with such characteristics. These multi-modal, multi-camera characteristics make this dataset amenable to many possible action recognition techniques. This dataset could form the basis of future research into pervasive computing applications for OSCEs.

Each chapter of this thesis will include a clear outline of the contained contributions, including additional smaller contributions not discussed here.

1.5 Structure of Thesis

A breakdown of the remaining chapters of the thesis is as follows:

Chapter 2 Recent research in the field of Artificial Neural Networks is reviewed due to its relevance across the thesis. Action recognition research, most relevant to the thesis contributions, is examined in detail.

Chapter 3 A pipeline for action recognition of a human-object interaction task, based on tracked object and human poses, is designed, implemented and evaluated. One stage of the pipeline is a multi RGB-D camera data acquisition system employed to reduce the effects of occlusions. Another stage involves methods of estimating 3D poses of objects and arms. A novel method of identifying a recurrent neural architecture through a grid search is outlined in the classification stage of the pipeline. A description of a dataset of people preparing cups of tea is given. This dataset is used to evaluate the entire pipeline for action recognition purposes against benchmark algorithms. For this exploratory dataset, our pipeline achieves 83% frame-level accuracy.

Chapter 4 Techniques for fine-grained action recognition from RGB-D data are devised, implemented and evaluated against a benchmark dataset. A novel technique of fusing features of different image types, including colour, depth and scene flow, in a neural network architecture is described. This approach removes the need to train parallel network branches for each image type, cutting computational cost. The use of scene flow information is proposed as input to a neural network approach for action recognition from RGB-D data. Devised architectures, involving a combination of convolutional and recurrent neural networks, are evaluated for fine-grained action recognition purposes. The approaches outlined in this chapter are found to have benefits over the pose-tracking approach of the previous chapter.

Chapter 5 This chapter builds on the previous findings of the use of convolutional neural networks for fine-grained action recognition from RGB-D images. The WeaveNet architecture is devised, which directly tackles the problem of learning from multiple image types throughout a network. A novel method of including temporal information as part of the action recognition network is described and shown to improve performance. The collection of a dataset of venepuncture OSCE performances is described, and the performance of the WeaveNet architecture is reported on it.

Chapter 6 The main contributions of the thesis are discussed, and possible directions for future work are outlined.

In addition to an outline of contributions, each chapter will begin with the motivation behind the work of the chapter.

Chapter 2

Background

In this chapter, key areas of the literature that are relevant to the contributions made in tackling the problem of understanding fine-grained actions are reviewed. In this thesis, we make use of state-of-the-art techniques and neural network architectures, as well as contribute specific architectures for the task of fine-grained action recognition. As such, we review the latest techniques utilised to train such networks, as well as specific architectures for tasks such as learning from sequences and learning from images. The field of action recognition will be reviewed, with particular attention paid to works and sub-fields that apply to our research goal. Here, we identify particular unsolved problems that will be addressed in the remainder of the thesis.

2.1 Artificial Neural Networks

Artificial neural networks are a type of machine learning model that comprises of multiple layers of logistic regression models with continuous nonlinearities [23]. The hierarchical structure of these layers is referred to as an *architecture*.

2.1.1 Network Training

The recent renaissance of neural networks use for pattern recognition tasks can be attributed to several factors. The existence of large labelled datasets, such as ImageNet [24], has allowed these data-driven approaches to flourish. The ability of neural networks to be fine-tuned for a particular task [25, 26], having been initially trained on these datasets, has led to utility being derived from these datasets in research directions oblique to the original goals of the dataset [27, 28]. Another factor contributing to the success of neural networks has been the availability of highly parallel processors in the form of Graphics Processing Units, given the inherent parallelisability of the calculations involved in training neural networks. These factors are completed by recent research breakthroughs that have enabled the training of neural networks sufficiently deep to permit sizeable increases in performance across research goals.

Early deep neural network architectures were trained layer-wise using stacked Autoencoders [29, 30], a type of unsupervised training, to detect increasingly complex, hierarchical, features. A weakness of this approach is that layers are not trained against the same task and thus may be encoding features unrelated to the final task. An approach enabling end-to-end training by fine-tuning for a discriminative task [31] made use of the backpropagation algorithm [32]. Success using this training paradigm [33, 34] led to end-to-end training of deep networks without unsupervised pre-training [35].

The expressive power of deep networks can often lead to situations where overfitting to the training data occurs. Overfitting occurs when a model is optimised more towards correctly classifying a training set, rather than representing the underlying generating process of the data. A novel technique, called Dropout [36], was devised to help overcome this effect by randomly dropping connections between successive neurons so that later features are encouraged to more thoroughly use all available inputs.

Batch normalisation is a technique that made the training of increasingly deeper networks possible [37]. This technique normalises the inputs of individual layers in the network to overcome the shifts that may occur due to changes in previous layers as batches of data are propagated through the network.

The choice of initial values for the trainable weights in the network has been an important determinant of the training convergence of deep neural networks [38], with random initialisation identified as an impediment to training deeper networks by Simonyan and Zisserman [39]. Particular initialisation strategies [40, 41, 42] demonstrated improvements on this issue, with strategies devised for specific nonlinearities [43] enabling more stable network convergence.

Research on initialisation techniques has also focussed on the nonlinearities used in the network [43] to counter the problem of vanishing gradients in deep architectures. Nair and Hinton found that the Rectified Linear Unit (ReLU) [44] performed better than other commonly used nonlinearities for deep networks and avoided the problem of activations becoming saturated at a maximum value.

These advances contributed to the ability to train arbitrarily deep networks, however other significant advancements, such as skip connections, will be discussed in Section 2.1.3.

2.1.2 Recurrent Neural Networks

Neural networks have been specifically adapted for use on sequences of data, such as text, speech or time-series data. As we aim to understand actions across time, this area is of specific relevance to our research.

A particular type of network architecture designed for such data is the recurrent neural network [45]. In a recurrent neural network (RNN), the computation graph is permitted to be cyclic [46]. This cyclicity enables sharing of trainable network weights across individual timesteps of a sequence [47]. By examining the modelling capability

of RNNs, it has been shown that, given a sufficiently large number of neurons, any mapping of a single-dimensional sequence to a bounded range can be approximated with finite resources [48, 49].

A difficulty faced when training RNNs is that, if inputs sequences have discriminating features that span long intervals, it is challenging for the network to capture these long-term dependencies [50]. This difficulty occurs when the network is trained using backpropagation as error gradients become vanishingly small with increasing network depth (the *vanishing gradient problem*). A successful approach to overcoming this problem is the Long Short Term Memory network (LSTM) [51].

The key insight involved in the design of the LSTM is that long-term dependencies can be captured through the introduction of an internal state that is modified as a sequence is processed. The cell is controlled by a gating mechanism that allows or disallows modifications to the cell state based on the activated output of the current sequence element. Further improvements to the LSTM design included the addition of an equivalent "forget" gate [52] that clears certain cell states, based on the activated output, so that the cell can forget now redundant information.

LSTMs have been successfully applied to many real-world problems including speech recognition [53], image captioning [54], handwriting recognition [55], and language translation [56]. A recent observation of the properties of LSTMs [57] noted that, given enough resources, LSTMs can internally learn representations that correspond to high-level concepts. This property was evidenced by the observation that, when trained to generate text from a dataset of shopping website reviews, a single neuron in the LSTM was responsible for encoding the sentiment of the generated review.

There has been renewed interest shown in developing alternative cell architectures to LSTMs. The Gated Recurrent Unit (GRU) [58] is one such design. This design decreases the number of trainable parameters by using a single update gate in place of separate input and forget gates, as in LSTMs. This design, and further modifications upon it [59, 60], have been shown have similar performance to LSTMs, outperforming LSTMs in some instances when trained with the same number of parameters. More recently, Zoph and Le [61] used an LSTM to generate new recurrent neural network architectures, and trained these architectures using reinforcement learning with a reward based on the validation accuracy, in a method called Neural Architecture Search. As a result, a cell architecture was found that outperforms LSTM on a standard benchmark and generalises to other tasks.

We have covered research for the design and training of recurrent neural networks. However, there exist other directions of research involving learning from sequential data. These include Neural Turing Machines [62] and attentional interfaces [27, 56]. Such areas may represent avenues for exploration beyond this thesis.

2.1.3 Convolutional Neural Networks

Given the prevalence of image data in many machine learning tasks, a neural network design has been devised for processing such data. Convolutional neural networks (CNNs) are a type of neural network where the neurons operate on discrete grid-like structures. The neurons perform a convolutional operation (often implemented as a cross-correlation operation [47]), with the convolutional kernel representing the trainable weights of the neuron. Previously, we noted that recurrent neural networks share weights by applying the same operation across time, however for CNNs, weights are shared across dimensions of the input grid. The size of the convolutional kernels is bounded and is applied to subsamples of the input grid, with the same dimensions as the kernel, known as the *receptive field*, to produce output values. The subsampling is usually performed in a sliding window fashion, meaning that the convolutional operation is translationally invariant, a desirable property for recognition tasks. The first example of a CNN in this understanding was LeNet by LeCunn [63], which outperformed other contemporaneous methods at optical character recognition.

Recently, research interest in utilising CNNs has undergone a tremendous resurgence. Due to their ability to output image features, which can be trained or fine-tuned for specific tasks, they have become a versatile tool, applicable across many computer vision problems. Some of the problems to which they have been applied to include human pose estimation [64], semantic segmentation [65], and super-resolution [66]. For our purposes, we note that CNNs have been utilised to recognise human actions in video, which we will discuss in Section 2.2.2.

After the original CNN design introduced by LeCun, research into CNN architectures stalled. The research was revitalised by performance achieved by Krizhevsky et al. [35] with the AlexNet architecture in the ImageNet Large-Scale Visual Recognition Challenge (LVRC) [24]. The architecture involved the use of ReLU nonlinearities, dropout regularization, and max pooling (an operation whereby the maximum outputs of subsampled regions are taken to decrease the spatial dimensions and hence the amount of processing required). In this work, and subsequent work [39], it was found that network depth was strongly correlated with performance. Specific best practices were established via this work, such as using small convolutional filters in place of large receptive fields [39], the use of batch normalisation [67], and performing activation before convolution [68]. The unwanted effects of the phenomenon of vanishing gradients, whereby increasing network depth led to reduced accuracy, was tackled in the GoogLeNet architecture [69] by utilising different receptive field sizes in parallel and stacking the outputs. The problem of diminishing feature reuse [70] is also encountered in deep networks, where the input features become less useful for gradient learning of later layers due to diminishing information as they are convolved with more filters. These restrictions on the depth of the network was an impediment to the representational power and hence the performance of the network [71].

These problems encouraged different approaches to overcoming them. The approach of Highway Networks looked at information flow in the network and created pathways through which information could flow unimpeded [72]. Taking inspiration from the design of the LSTM [51], the pathways (or "highways") are controlled by gating mechanisms which regulate the flow of information. The most impactful approach, however, to the problems was that of He et al., who introduced the residual connection [73] (also known as skip connections). By introducing a connection parallel to layers in the CNN, through which information flows unchanged, the layers of the networks are reframed as residual functions (functions with the addition of the input). The rationale behind such an approach is that if the optimal function is closer to an identity mapping, it should be easier to find perturbations to be applied to it, than to learn a new function [73]. As an outcome, a deep architecture enabled by skip connections, the Residual Network (or ResNet), was able to outperform all previous architectures on many image recognition tasks of the ImageNet LVRC. Residual connections were rapidly adopted in other state-of-the-art architectures [74].

In work by Veit et al. [75], a weakness of the residual network approach was uncovered. It was found that deep residual networks leverage only short paths during training, and observed that their representational power comes from the fact that they behave like ensembles of relatively shallow networks. An architectural approach to overcoming the diminishing returns of increasing depth in residual networks, as a result of this phenomenon, was the Wide Residual Network [76]. This approach outperformed deep residual networks using a shallower network with an increased number of convolutional filters (or width) in the residual layers. Later architectural approaches include DenseNet, where dense connections between individual layers are used [22], i.e. skip connections that skip multiple convolutional layers. This approach effectively introduced many shorter connections throughout blocks (of layers) in the network to reduce the effects of diminishing feature reuse. The variety of these approaches indicates that there may be further research insights to be made in the architecture of CNNs.

Each of these discussed architectures has been benchmarked, and designed, against single image recognition tasks. Methods of action recognition can utilise such networks in a feature-extraction manner. However, potential insight can be gained by exploring how these CNN architectures can be adapted for use on video inputs.

2.2 Action Recognition

Recognising the activities of humans from video is a long-standing research problem in computer vision [77, 78, 79]. Being able to understand an activity from images has a large number of potential applications, from image and video retrieval [80], image description [81, 82] and surveillance [83], to pervasive computing applications such as smart home monitoring [84].

The terms activity recognition and action recognition do not have a consensus meaning in the computer vision community and are often used interchangeably. A proposed definition by Moeslund et al. [85] states that an *activity* is a large-scale event which is composed on individual *actions*, which can be defined as describable semantic events. An activity may also rely on contextual information such as scene, objects and persons present. Actions can, themselves, also be further broken down into atomic units known as action primitives [86]. For example, in the activity of playing basketball, the action of throwing the ball may be composed of the action primitive of moving one's arm upward. To use a linguistic analogy, action primitives can be thought of as phonemes, actions as words, and activities as sentences. For our purposes, the activity can be considered to be known a priori and activity recognition will be beyond the the scope of this research. The interested reader is referred to the literature review of Aggarwal et al. [87], for historical approaches, or the review of Vrigkas et al. [88] for more recent techniques.

The problem of action recognition can be broken down into different but related problems. One problem is the localisation of actions within a video [89], i.e. generating a bounding box for each frame of a video that encloses an action in question. Another such problem is temporal action localisation [90], which refers to the problem of identifying the frames of an untrimmed video in which an action occurs. The problem of recognising the action that is taking place in series of video frames can be referred to as action classification, however, commonly in the literature it is referred to with the more encompassing name of action recognition. In this thesis, we shall also refer to it as simply action recognition.

2.2.1 Feature-Based Approaches

There exists a wide-range of approaches to action recognition in the literature [91, 92, 93]. Due to the recent success achieved by extraction and classification of hand-crafted features [94], we examine this class of approaches.

Early feature-based approaches to action recognition extract global features from images and perform classification using sequential generative classifiers such as Hidden Markov Models [77]. Optical flow information has been used across a number of approaches [95], more recently in the form of Histograms of Optical Flow (HOF) [96]. Similar features used for action recognition include Histograms of Oriented Gradients (HOG) [97], and Motion Boundary Histograms (MBH) [98].

The pose of the human subject in an image is used in some approaches [99, 100], where sequences of estimates of the pose are used to train classifiers. By encoding the pose trajectories over multiple frames, or information regarding dynamics [101, 102, 103], spatio-temporal features for classification can be produced. Pose information has also been combined with the location information of objects in use to recognise actions [104]. By modelling the context of these objects relative to the human, discriminative

information for recognition purposes can be discovered [105].

Local image features have played a significant part of many action recognition techniques [106, 107, 108]. A feature common to many approaches [109, 110, 96] is the Space Time Interest Point [111]. Feature detection techniques extend similar methods from the 2D domain, such as Harris3D [111] which extends the Harris detector [112], to detect local structures in videos where values have a large variation in space and time. Other approaches of detection rely on saliency heuristics for blob detection to locate spatio-temporal interest regions [110]. Taking small local regions around detected interest points, histograms of motion or spatial information, such as HOF, HOG [96], or HOG3D (an extension of the SIFT descriptor to space-time [113]), can be used as feature descriptors. Methods of classification utilise discriminative classifiers such as Support Vector Machines [114, 115], sometimes in combination with other techniques such as Bag-of-Words [109]. An evaluation of these techniques by Wang et al. [115], noted that although these approaches worked well on gross motions, such as waving or jumping, they underperformed on more realistic settings. Conversely, approaches that densely sampled features underperformed on simple actions and outperformed on more realistic datasets.

The approach of densely sampling points was pursued by Wang et al. [95]. Optical flow information was used to improve the tracking of dense 2D features over time. A descriptor based on Motion Boundary Histograms was used for classification. The technique was later improved to remove camera motion and to detect the human so that descriptors were focussed on the human motion [94]. This approach and later variants [116, 117] were very successful, outperforming other state-of-the-art methods.

The approach of dense trajectories for complex subtly different actions lacks semantics and discriminative capacity [118]. As such, research has been performed that attempts to learn mid-level and high-level representations of actions. Techniques such as Actons [119] and Action Bank [120] learn a higher-level representation space for actions based on combinations of low-level features. Other approaches have focussed on drawing representations from key poses of humans performing the actions [102, 121], or constructing a hierarchy from human and object segmentation at a low level to a spatio-temporal representation at a higher level [122]. The mid-level approaches often utilise low-level features [119, 102] in an off-the-shelf manner as part of their approach. However, it is not clear whether these features are optimal for such hierarchical approaches.

For our purposes, the drawback of many of these techniques is that they can be computationally expensive and may prevent the timely classification of actions for the purposes of online applications. Furthermore, it has been shown, that for small actions, such as those involved in preparing a meal [93], spatio-temporal feature approaches struggle to classify correctly. Given that we are focussed on classifying fine-grained actions, this approach may prove limiting.

2.2.2 Deep Learning Approaches

Context

We discussed in Section 2.1.3, the successful application of convolutional neural networks (CNNs) to the problem of image recognition. However, the application of deep learning to video inputs was slow to begin in comparison. Early work [123] attempted to learn spatio-temporal features using convolutions over pairs of images, as a replacement for hand-engineered features. It was found, though, that it was difficult to surmount the hand-crafted feature-based approaches of dense trajectories [94]. The limiting practical factors, such as the training time required and the lack of large labelled datasets (akin in scale to ImageNet [24]), meant that the significant performance strides of the image domain [35] did not straightforwardly translate to the video domain.

Early Approaches

A number of key results established the applicability of deep learning methods to action recognition, with the empirical findings informing current best practices. Due to the success of the approach of dense trajectories on certain datasets [94], some work continued in this vein attempting to learn deep convolutional feature descriptors [118, 124]. By combining dense trajectories and learned convolutional features [118], it was possible to outperform then state-of-the-art hand-engineered features. The lack of suitably large datasets was addressed by Karpathy et al. who collected over 1 million weakly-labelled sports clips [125]. In this work, they also observed that CNNs trained on single frames, were near as accurate as those trained on clips of multiple frames. In order to make the learning process more tractable, some approaches attempt to first isolate the regions in the images containing the motion [89, 126]. The motion is first found using a salient region detector and then a CNN is trained to recognise actions based on the cropped images of these Action Tubes [89]. A disadvantage of this method is that it ignores often valuable information that may lie in the surrounding context of the action. Such a result was observed by Karpathy et al. when training on cropped images of sports actions [125]. The approach of training two separate convolutional networks, for colour and optical flow information respectively, was proposed by Simonyan and Zisserman [16] and widely adopted in other techniques [89, 127]. This work also observed the benefit of utilising pre-trained networks, finding that the spatial network (colour image inputs) performed better when pre-trained on the ImageNet dataset [24]. This transfer learning process was also shown to significantly improve recognition results on smaller datasets when pre-trained on a much larger action recognition dataset [125].

CNNs and RNNs in Combination

The previous methods utilise small clips of video for training CNNs [125], or stacked optical flow features to incorporate temporal information [16], however the amount of

temporal information for the neural network to learn from is limited by constraints on the size of the CNN. Given that recurrent neural networks (RNNs) are designed to learn from sequential inputs, the combination of CNNs and RNNs was explored. Donahue et al. [127] utilised an approach that learns image features for single frames using the AlexNet architecture [39], and an LSTM to obtain action recognition class distributions based on a sequence of these image features. The availability of pre-trained state-of-the-art CNNs can be straightforwardly exploited using this method. The method trains separate networks for the colour (appearance) and optical flow (motion) image inputs, combining the outputs by averaging. Marginal accuracy improvements were observed against using only a CNN on a single frame, with greater improvements for the optical flow input type. A similar approach was also employed in by Ng et al. [128], who observed that an alternate approach of pooling the outputs of the CNNs was competitive with the joint CNN and LSTM approach. It was also observed that the optical flow data has a higher learning utility when videos are more uniform.

Modified Convolutional RNNs

The approaches described thus far attempt to refit neural network architectures previously applied to other tasks, such as image recognition for CNNs and sequence prediction for LSTMs, to the task of action recognition of videos. Accordingly, researchers have investigated architectures specifically designed for video data [129, 130, 131]. Ballas et al. combine the GRU recurrent network [58] with a CNN. By replacing the input gate operation of the GRU with a convolutional operation, stacking layers of such cells, and using as inputs the outputs of progressive layers of a pre-trained CNN, the recurrent network is able to learn from sparse intermediate image representations across the frame sequence. A similar approach of replacing fully-connected operations with convolutional operations has also been performed for the LSTM cell [131, 129]. Utilising an attentional mask based on the motion information, Li et al. [129] were able to outperform other state-of-the-art techniques using a shallow network of such convolutional LSTM cells.

3D Convolutional Approaches

Another alternate neural network approach to action recognition has been to extend the convolutional operation to the temporal dimension (i.e. 3D convolutions) [132, 133, 134]. In this way, the convolutional weights are trained to recognise spatio-temporal structures. With an architecture similar to that of AlexNet [39], Tran et al. utilised 3D convolutional operations in place of 2D to achieve good performance [134]. The use of 3D convolutions was shown to be an advantage when attempting to learn from long-term dynamics that may be present in a video [135]. Due to the invariance of the convolutional operation across a spatio-temporal volume, it can be applied to video clip volumes of increased temporal length without an increase in the

number of trainable weights. A disadvantage of the 3D convolutional approach is that the increased dimensions of the 3D convolutional kernels may necessitate an increased amount of training data to learn to recognise spatio-temporal patterns. To address this, Sun et al. [136] demonstrated that spatio-temporal (3D) convolutional operations can be factorised into separate spatial (2D) and temporal (1D) convolutional operations, leading to a reduction in the number of weights to train.

Temporal Fusion

There has also been investigation of methods of fusion of the temporal information of a sequence of images as part of an otherwise 2D convolutional network. Karpathy et al. [125] found that progressive fusion across the temporal dimension, via the use of parallel spatio-temporal convolutional operations (i.e. 3D convolutions) with a fixed temporal receptive field, lead to increased performance. Ng et al. [128] noted that the technique of maximum pooling of output convolutional filters across time outperformed other tested techniques of temporal pooling. Feichtenhofer et al. [137] found that for the two-stream approach (i.e. separate network branches for appearance and motion), performing 3D convolution in combination with 3D pooling over spatio-temporal output features, increases performance compared with the baseline approach of averaging softmax activated outputs over time. In their later work [138], skip connections are introduced in the temporal domain so that the network learns from a larger temporal receptive field as depth increases. Finally, a rank pooling machine [139] has also been used to combine the temporal sequence of colour images into a single colour image, called a Dynamic Image [140], which can then be classified with a CNN.

Fusion of Appearance and Motion

The approach of training two separate networks, one for appearance and one for motion data, as devised by Simonyan and Zisserman [16], leads to the question of how to combine the outputs of the two networks. Fusion of the class distributions outputted by the two networks can be performed by weighted average [127, 89]. Alternatively, a discriminative classifier, such as a Support Vector Machine can be trained to produce a final class distribution [16, 128, 135]. A disadvantage of these approaches of late fusion is that information regarding precise co-locality of features across appearance and motion inputs cannot be utilised for learning purposes. Feichtenhofer et al. [137] investigated this problem by examining techniques of fusion of convolutional features of the two networks. It was observed that concatenation, followed by a convolution operation, was an effective technique of fusion of the convolutional feature maps of the two streams. It was also noted that performing this fusion later in the network, resulted in an increase in performance. In a later work by the same authors [138], the authors replace these VGG networks with residual networks [73] and utilise residual connections to fuse convolutional feature maps from the motion stream into the appear-

ance stream. Connections in the opposite direction led to worse performance, likely due to appearance features dominating over the motion features. More recent findings regarding fusion between parallel branches, such as those of the SlowFast architecture [141], will be discussed later in this section.

Temporal Sampling Approaches

The problem of action recognition is commonly addressed as a problem of classification of short clips [16, 137]. In reality however, correct recognition of an action can depend on long-range temporal structures, i.e. it may require observation of features separated in time. CNN approaches to action recognition which have a limited temporal receptive field cannot learn to recognise actions based on such features. We have examined approaches that use recurrent neural networks [127] and temporal pooling [125] to facilitate larger temporal receptive fields, however these approaches were used to recognise actions from short-term spatio-temporal patterns [142]. Temporal sampling strategies are used to facilitate learning from longer-range temporal structures.

One method of enlarging a network's temporal receptive field is to use strided temporal convolutions [135], however the stride that can be used effectively is limited. Alternatively, Temporal Segment Networks [142] established temporal sampling strategies that have been used in many later works [141, 17, 143] to learn from long-range dependencies in videos. In this temporal sampling strategy, a video is divided into a fixed number of segments and a short clip is selected from each segment. The CNN output for clips from different segments are fused using an aggregation function and used to predict a class score for an entire video. It was found that the unweighted mean performed comparably to an aggregation function with tuned weights.

Based on the observation that humans can recognise temporal relations between pairs of images, and infer the transformation, or action, that took place, a recent work sought to learn these temporal relations. Temporal Relation Networks [144] use the relational reasoning module of Santoro et al. [145] to describe temporal relations between observations in video. In the approach, linear network layers learn relations between ordered pairs of output CNN features for frames in a video sequence. Similar linear network layers also operate on ordered n-tuples ($n = 2, 3, \ldots$) of frame features. The outputs of these linear network layers undergo a further linear layer before combination by addition to produce a prediction. This technique improves action recognition for sequences with strong temporal dependencies, and can be used for other applications such as identifying representative frames for an action.

The temporal sampling strategy of Long-term Feature Banks [146] decouple the learning of short-term features from long-term modelling by maintaining a bank of previously sampled features. Such feature banks act as a 'memory' of a video, and can be constructed of output CNN features for regularly sampled frames. These long-term features are combined with short-term features using an attention mechanism,

referred to as a feature-bank operator, before use as an input to a classification layer. A temporal variation on the non-local neural network [147] is used for this feature-bank operator. Non-local networks are a technique specifically devised for learning from long-range dependencies by computing the network response at a position as a weighted sum of features at all input positions. By employing Long-term Feature Banks as an auxiliary component of action recognition networks, it is possible to add further temporal context by sampling long-term features.

An approach that permits 2D CNNs to efficiently access a larger temporal receptive field was explored in the work of Temporal Shift Modules [148]. This work uses shifting operations to exchange features between neighbouring frames within a 2D CNN architecture, effectively performing temporal fusion. The exchange of information within a temporal sample enables recognition performance in line with established 3D CNN approaches.

Based on the observation that in an untrimmed video for an action, salient clips can be separated by long segments of irrelevant information, a method of sampling the most salient clip, SCSampler, was introduced by Korbar et al. [149]. By using a lightweight model to estimate the saliency of all clips in a sample, the number of clips that require forward propagation through the heavier action recognition CNN can be greatly reduced. The sampler model can be made lightweight by operating on compressed video features or audio features. It can be trained independently of the action recognition method, or specifically for an action recognition model by ranking the importance of clips to the action classification. It is shown to outperform established temporal sampling strategies, such as Temporal Segment Networks [142].

The method of AdaFrame [150] is a temporal sampling approach that trains a reinforcement learning agent to select frames in a video to observe for classification. In this approach, a global memory representation is constructed by observing an entire downsampled video. An LSTM network is augmented with this global memory such that when given video frame features as input it predicts the video class, the utility of seeing subsequent frames, and the frame to observe next. A reward function is specified based on increasing classification confidence when observing another frame, and is maximised using policy gradient methods. In qualitative results, the technique learns to use more frames for difficult to classify samples.

Each of these discussed temporal sampling approaches can be used to improve action recognition performance when there exist long-term temporal dependencies.

Multiple Appearance Branches

The approach of using multiple branches for action recognition is not constrained for use only with multiple image modalities. There also exist methods that use multiple branches for the same image modality.

SlowFast [141] is an action recognition approach that uses different temporal res-

olutions for the colour image modality. Based on observations regarding human perceptions of motion being biased towards slow movements, the authors suggest a two-branch 3D CNN treatment of colour spatio-temporal information. This work argues that spatio-temporal information (i.e. video) requires a different treatment to the spatial information (i.e. images) because spatial isotropic invariance (i.e. all orientations are equally likely) does not extend to spatio-temporal information (i.e. all motions are not equally likely). In the formulation, a *slow* branch is primarily responsible for recognition of slowly refreshing categorical semantics from appearance information (i.e. colour), such as persons, hands, etc. As such, this branch can be trained using a few sparse frames and operates at a low frame rate. Conversely, a *fast* branch is primarily responsible for recognition of faster refreshing semantics involved in motions, such as clapping, waving, etc. This branch is trained with a larger number of colour frames and operates at a higher frame rate. The fast branch is lightweight relative to the slow branch based on the perceptual bias towards slow movements, and to reduce spatial processing redundancy across the two branches. Lateral connection between the two branches are made from the fast branch to the slow branch, with features undergoing a temporal convolution with stride before fusion by concatenation. In the evaluation, this approach increase performances across a number of benchmark action recognition datasets, exceeding state-of-the-art.

An earlier work that looked at using multiple branches within the same modality was the Appearance and Relation Networks by Wang et al. [151]. This network attempts to separate the learning of spatial appearance and temporal relations, while using only colour information as input. The network consists of blocks with parallel branches for these learning tasks. The appearance is modelled using standard 2D convolution operations operating on single frames. The temporal relation is modelled using 3D convolution operations to approximate the multiplicative interactions between patches across frames. On benchmark datasets, it achieves performance in line with Inflated 3D Networks [17], a 2D CNN inflated to 3D (i.e. initial 2D convolution filter weights are scaled and repeated across the added temporal dimension).

Similar to how SlowFast networks use different temporal resolutions in two CNN branches [141], the approach of Big-Little-Net uses two CNN branches to learn features at different spatial resolutions [152]. This Big-Little approach was extended from image recognition to video recognition by Fan et al. [152] in their Big-Little-Video-Net approach. In this approach, a deep branch operates on lower-resolution frames and a shallower branch operates on higher-resolution frames, together minimising computational costs. To capture short-term temporal dependencies, adjacent frames are passed as inputs to the two branches and features are fused by addition at multiple locations in the network. To capture long-term temporal dependencies, a Temporal Segment Network [142] approach is used with depth-wise convolution and temporal shifting across sampled frames employed to obviate the use of expensive 3D convolutions. The

approach is efficient when compared to baseline approaches of Temporal Shift Modules [148] and Temporal Segment Networks [142].

The works discussed demonstrate that action recognition performance can be improved by using multiple branches to treat the same modality information differently.

Multi-Modal Distillation

The two-stream approach to action recognition of Simonyan and Zisserman [16] can be criticised on the basis that inference can be costly, requiring two deep parallel CNN branches and the computation of optical flow features. This criticism extends to situations involving additional image types, such as action recognition from RGB-D data where an additional branch is used for depth images [118, 153]. In the literature, some approaches attempt to overcome the requirements of availability of additional image types and modality-specific models at test time [154, 155, 156]. These approaches fall under a specific learning paradigm, named Learning Using Privileged Information [157] based on the availability of 'privileged information' from additional modalities during training time. This problem can be tackled using knowledge distillation [158], a technique to combine the knowledge of multiple prediction networks into a single network [159, 160]. When performing distillation, the network with greater availability of information (and indeed representational power), a teacher network, is 'distilled' into a network with lesser availability of information, a student network.

There are various approaches of performing the model distillation task, for example one recent work does so by jointly minimising the difference between intermediate feature representations and the difference between the output distributions of student and teacher networks [161]. In this work, model distillation is used to reduce the number of frames of a video necessary for action recognition. We shall focus on works that perform distillation for recognition tasks from multi-modal image types.

An early work that looked at the problem of distillation across image modalities was that of Gupta et al. [154]. Here, a teacher network is pre-trained on the ImageNet dataset and a loss based on intermediate feature representations is used to train a CNN which takes depth images as input. This approach improves object recognition performance compared with solely training the depth network with a classification loss.

The problem of how to effectively leverage privileged information during distillation is addressed using a graph mechanism by Luo et al. [162]. The graph nodes take as inputs the outputs of modality-specific CNN branches. It can be trained in an end-to-end manner, learning to adjust the distillation based on output CNN representations rather than on a pre-specified loss, thus dynamically using more informative modalities when beneficial. This approach achieved state-of-the-art performance using only a single modality in a distilled model for RGB-D action recognition tasks.

An alternate approach of using distillation to perform action recognition on RGB-D was explored in the work of Garcia et al. [155]. In their approach, teacher colour

and depth networks are firstly trained independently on the classification task, before being fine-tuned jointly using a mid-stage fusion (by multiplicative connections of spatio-temporal features from depth to colour branches) and a late-stage fusion (by concatenation). To train student networks, a joint two-branch architecture is trained from initial weights of the depth network, however one branch is provided colour inputs while the other branch's weights are frozen. A loss is imposed based on differences of intermediate feature representations so that the colour network learns to 'hallucinate' the depth representations. A loss based on output predictions, compared with both teacher predictions and ground truth, is also imposed. The branch that learned to hallucinate depth representations is combined with the teacher colour branch for a final training step. This final model no longer requires depth information during inference and outperforms networks trained using a single modality. Performance however remains markedly lower than the teacher network trained jointly on both modalities.

In a follow on work by the same authors [163], an adversarial loss is used to train a network that hallucinates depth representations. The model used for this adversarial loss is the Conditional Generative Adversarial Network [164]. A hallucinating network serves as the generator while a discriminator network discriminates between outputs of the hallucinating network and those of a frozen teacher depth network. Using this method, losses based on differences of intermediate representations and teacher network predictions are no longer required. Furthermore, this approach reduces the RGB-D action recognition performance gap between the student network and teacher network.

The problem of requiring motion branches in two-stream networks was addressed by Crasto et al. [156]. In the first of two proposed approaches, a student colour network hallucinates optical flow feature representations using a loss imposed on differences between features late in the colour and optical flow networks. By using a similar architecture for the branches, the student branch learns to accurately emulate motion features, indicating that costly optical flow calculation may be unnecessary. In a second proposed approach, an additional classification loss is imposed on the colour network outputs for an action recognition task. The student network outperforms both the colour and flow networks and has comparable performance to the two-stream approach.

The discussed works show that CNN branches for specific additional image types are not necessary during inference to benefit from available image types. However, in all cases performance is maximised when all image types are used during inference.

Multi-Modal Self-Supervision

Self-supervision refers to the learning paradigm in which training is not performed using strong annotations but rather using relational information that comes from the input data. Examples of such relations include: how much an image is rotated by [165]; how to colourise a grayscale image [166]; relations between image patches [167, 168]; and temporal relations between video frames [169, 170]. Such relations can be readily

synthesised from the input data (e.g. a colour image can be converted to grayscale). These tasks are difficult for models to solve without learning useful representations of the data. Such learned representations can be subsequently refined on specific classification tasks, potentially benefitting from pre-training on large unlabelled datasets [154, 171]. Relations between multi-modal input data can also be used to provide supervision, such as using video and audio modalities to determine whether input samples come from the same audio-visual clip [172, 171] or whether modalities are aligned [173, 174]. Image types which are spatially aligned, such as colour, depth and flow, can also be used to perform self-supervision. We examine works that perform multi-modal self-supervision in the context of action recognition.

We have already discussed the work of Gupta et al. [154] that performed model distillation across image modalities. In this approach, mid-level representations from a colour network (previously trained using full supervision) are used to provide a training loss to an untrained depth network. This is performed using unlabelled image pairs across the modalities, and so can be seen as a form of self-supervision. The depth network is subsequently fine-tuned on a smaller dataset specific to this modality.

Another discussed work that utilises multi-modal self-supervision is that of Crasto et al. [156]. The first of their two approaches falls into the category of self-supervised pre-training, as the training loss for a colour CNN is based on differences between colour and optical flow features. A second stage of this approach involves training the final layers of this colour CNN using a cross-entropy loss on target dataset labels. As such, this self-supervised approach could be used to learn motion representations from optical flow features computed for any unlabelled colour video dataset.

In a recent work by Munro and Damen [175], multi-modal self-supervision is used when tackling the problem of domain adaptation for fine-grained action recognition. Domain adaptation (or in this case unsupervised domain adaptation) refers to using a model trained on one domain for a different domain (which is without labels in the unsupervised case). In this work, the domains are different kitchen environments from the fine-grained action recognition dataset EPIC-Kitchens (multi-task dataset) [176]. During training, a self-supervision loss is imposed based on whether colour and optical flow features are from the same actions. This loss is used in combination with domain discrimination adversarial losses for each modality and a loss based on the average-pooled classification scores for each modality. This approach thus can take advantage of multiple modalities to better align domains. The method outperforms state-of-the-art domain alignment methods and demonstrates classification improvements due to multi-modal self-supervision when using only a single modality for prediction.

These works show how additional image modalities can be used as a self-supervision pre-training task to subsequent classification tasks, as well as how multi-modal self-supervision can be directly integrated in training for a classification task [175].

Summary

We have looked at deep learning approaches to action recognition in videos. We note that as a representation of spatio-temporal information, the use of optical flow is pervasive in many of these techniques, and that it has a higher learning utility when videos are more uniform [128], which will be true in the case of our fixed-setting action recognition system. Approaches utilising flow often train two parallel convolutional neural networks branches [127, 16] imposing a heavy training and computational cost, and requiring careful consideration of fusion of the appearance and motion networks [137]. Multi-modal distillation techniques can reduce this burden, however they do so at the cost of recognition performance. The problem of training an efficient network on several image modalities (such as colour, depth, and flow), while maximising performance, will be tackled in this thesis. In the following section, we discuss work using deep learning to perform RGB-D action recognition due to its relevance to our research.

2.2.3 RGB-D Action Recognition

Context

With the emergence of consumer depth cameras in recent years, such as the Microsoft Kinect, interest has been shown in classifying human motions by utilising depth data. The recognition of these motions in this domain is often performed at the level of gestures [177, 178]. Aggarwal and Roo [87] define a gesture as an "atomic component describing the meaningful motion of a person" and an action as an organisation of "multiple gestures organised temporally". As such, we are interested in the classification of actions using depth data, as opposed to gestures, and we limit our review to techniques applicable to this task.

Feature-based approaches in RGB-D

Depth information affords the development of complex feature descriptors, with features developed specifically for action recognition. Histograms of Oriented Gradients (HOGs) extracted from images of depth information projected onto orthogonal planes have served as features for recognition [179]. Methods of extending the HOG descriptor to the spatio-temporal space have been pursued [180, 181]. One such extension generates quantized histograms of surface normal orientations of spatio-temporal motion patterns and has been shown to capture discriminative shape and motion cues for recognition of coarse actions [182]. Clustering such normals and extracting aggregate features based on a pyramidal subdivision of a space-time volume increased recognition performance [181]. A depth-based feature, the 3D Kernel Descriptor [183], has been applied to the problem of action recognition and been shown to reliably recognise coarse actions involving objects when combined with higher-level features (e.g. patches, spatio-temporal features) in a hierarchy [184]. Given that colour data may be

available in addition to depth data, Kong et al. formulated a technique of combining (HOG) features across the modalities [13] to learn a compressed shared feature space that can be used for classification of actions.

Pose Features

The availability of depth information has also enabled the development of techniques to reliably estimate human poses from the data [8]. Human action recognition techniques have utilised such pose features in their approaches. It has been shown that action recognition techniques that utilise human pose estimates outperform those that solely operate on dense-trajectories of low level features [185, 186]. Combining pose information with 2D image feature descriptors (HOG) has been shown to permit training of a generative action classifier using Hidden Markov Models (HMMs) [187]. Extracting mid-level representations of key action poses, in a view invariant manner, has also been shown to permit recognition using HMMs [188]. Using sequences of pose information over time, it is also possible to construct feature descriptors, such as spatial occupancy histograms [189, 190, 191] or joint angle affinities [180], to learn to recognise actions using discriminative classifiers. Higher order information, such as velocity of pose joints, can also be combined with such pose sequences to increase recognition performance [192, 193]. By exploiting the algebraic underpinnings of rigid body transformations, human pose sequences can be represented as curves in a Lie group, which can be discriminatively classified via a mapping to a vector space [194].

Hand Pose Estimation

In an OSCE, the subject primarily uses their hands to perform the task. One possible approach of understanding an OSCE performance would be to learn from estimates of hand poses. As such, we examine research in the area of hand pose estimation.

Early techniques made use of similar methods to the Kinect pose estimation technique [8], namely using random forests to predict pose joints based on hand-designed [9, 195, 196] or learned features [197]. However, weaknesses of these techniques were identified by Supančič et al. who showed that a nearest neighbour model outperformed much of the then state-of-the-art techniques [198], indicating that certain models did not generalise beyond the training set. The fact that the only models that outperformed the nearest neighbour technique made use of deep CNN architectures [198, 197] spurred the collection of larger datasets to better train these models [198, 199, 200, 12]. Later techniques made extensive use of CNNs to perform pose estimation [201, 202], using methods such as detecting individual joints by predicting heat maps using 2D CNNs [201] or using 3D CNNs to regress a full hand pose [202, 203]. Following the Hands in the Million Challenge [204], a review of submissions concludes that the hand pose estimation problem is largely solved for all but extreme viewpoints [205].

To tackle the problem of pose estimation for extreme poses, one recent work utilised

generative adversarial networks to synthesise depth maps for plausible skeleton arrangements [206]. Another work [207] utilises a point cloud representation to reduce complexity compared to 3D voxel-based approaches [208, 203]. A recent work [209] uses a point cloud autoencoder to enable training on unannotated data in a semi-supervised approach. Other recent approaches include connecting networks used for pose estimation sub-tasks [210] and using anchor points to more accurately regress joint locations by using local and global context information [211].

Each of the discussed techniques utilise depth images to estimate hand pose, but there also exists research dedicated to the more challenging problem of hand pose estimation using solely colour images [212]. A significant challenge of this problem is the lack of significant annotated datasets. Zimmermann and Brox [213] use a synthetic dataset to train an architecture that estimates 3D hand pose by using subnetworks for specific tasks such as predicting poses from 2D keypoints. Generative adversarial networks have been used to synthesise training data with losses imposed for geometric consistency [214] to improve estimation. Convolutional autoencoders have been used to generate poses and meshes for predicted hand joints when trained under weak supervision (via predicted joint keypoints for unannotated data) [215]. Graph CNNs have been used to predict pose and meshes for hands from colour images [216], again using weakly supervised training where corresponding depth images are used to provide a training loss. Weak supervision is also used in other works [217, 218], such as in the work of Yang et al. [217] who use point cloud representations constructed from depth information to train a network.

A challenging scenario in which to predict hand poses is during object interactions, where occlusions by objects and clutter can limit the available data for pose estimation [12]. Earlier approaches utilised depth data to perform this task [219, 220, 221, 12] such as the work of Choi et al. [220] which uses grasping hand configurations to better predict hand poses. Garcia-Fernando et al. specifically found benefits to using hand pose estimates for action recognition [12], however their results are based on a collected dataset in which ground-truth hand pose annotations are available from motion-capture gloves. The weak supervision approaches of hand pose estimation on colour images are also utilised for the object-interaction scenario in work by Baek et al. [222] which uses domain adaptation from hand only images to improve pose estimation performance in the hand-object scenario. This approach is adopted to deal with the problem of the lack of available annotated data of diverse objects and hand poses, which currently limits estimation performance. This limited performance in object interaction scenarios is the primary reason that hand pose estimates were not used in the action recognition approaches in this thesis.

Deep Learning on Poses

The sequential nature of human pose information over time lends itself to action classification using recurrent neural networks. By grouping sequences of poses of specific joints together as inputs, Du et al. [223] create a hierarchy of recurrent neural networks trained to classify coarse human actions. Veeriah et al. [224] modify the LSTM cell to include a gating mechanism that operates on the change of the cell state information to aid learning. LSTMs are also utilised by Li et al. [225] who train a network of such to perform classification and to temporally segment the action within an untrimmed sequence. Liu et al. [11] formulate the pose information at a single frame as a sequence itself to utilise LSTMs to recognise spatio-temporal patterns. Inspired by similar approaches in natural language processing, a gating mechanism is also introduced that operates on predicted and actual joint information to identify potential unreliable estimates. Deep learning approaches to action recognition from pose information have not been restricted to recurrent neural networks, with CNNs used to classify images of joint trajectories, colour-coded by time [226]. The performance of these action recognition methods that utilise pose information is, however, constrained by the performance of the pose estimation technique used.

Deep Learning on Depth

Convolutional neural networks operate on raw image inputs to discover hierarchical image features during training, and thus can operate on depth images themselves. However, until recently the application of CNNs to recognition of actions in RGB-D images was unexplored. Recent works utilising depth data have explored methods of using single image CNNs to perform recognition. Wang et al. [153], utilising a similar approach to Yang et al. [179], project depth information onto three orthogonal axes, and utilise a colour coding of combined depth sequences to produce a single colour frame for each. Three pre-trained CNNs are fine-tuned on the coded images, one for each aspect. The approach of Dynamic Images [140] was recently applied to RGB-D action recognition, with a single image produced via rank pooling of scene flow image inputs [227]. These 'action maps' are then classified using a pre-trained CNN.

Deep Multi-modal Fusion

Due to the availability of colour data in addition to depth, it is possible to train CNNs that operate over these multiple image types. An issue with training CNNs on this multi-modal data is that, if using a parallel network approach [16], each extra image type increases the number of parallel networks to train. In the domain of RGB-D gesture recognition, this parallel network approach has been popular [14, 15], with output class distributions of the individual networks combined by averaging. As shown by the work of Kong and Fu [13], there exists latent data structures that exists across

the modalities [13]. Using techniques that perform late fusion, such as combining by weighting final network outputs, precludes these individual networks from discovering such data.

Summary

In the techniques described above, the actions classified have been coarse actions such as 'jog', 'hand clap' and 'wave arm' [228]. The actions we are interested in recognising are finer-grained, such as those involved in an OSCE. Furthermore, pose information may be quite similar across actions sequences and thus difficult to classify using this information alone. Thus, it remains to be shown if the above techniques could be applied effectively to recognition of fine-grained goal-directed actions in a fixed setting. The problem of fusion of depth, colour and possibly other image types (such as motion information), in a deep learning architecture has not been fully explored and could enable discovery of joint salient structures that may improve recognition of the fine-grained actions, such as those in an OSCE.

2.2.4 Recognition for Situational Support

Context

The motivation of this thesis is the development of action recognition techniques that can be used in pervasive computing applications for goal-directed activities. In the research, systems have been developed for the purpose of providing situational support for goal-directed activities and such systems are reviewed here.

Sensor-based Approaches

There has been work performed to provide situational support for the elderly and infirm in their "activities of daily living" (ADL). Attaching RFID sensor tags to kitchen implements [229], situational support systems can recognise actions based on sensor data using generative classifiers such as Hidden Markov Models. Partially Observable Markov Decision Processes (POMDP) have been utilised by Hoey et al. [230, 231] to fuse the imperfect information received from numerous sensors. The POMDP model allows for a system to decide an action, such as whether to prompt the user, based on the belief states contained in the model. These belief states may include class prediction scores for observed actions. Such an approach has been used to develop systems that give real-time feedback to sufferers of dementia performing a goal-directed ADL such as preparing a cup of tea [231] or washing their hands [230]. Another approach that combines sensor information utilises accelerometers in combination with image features to train a discriminative classifier to recognise the actions involved in the goal-directed activity of preparing a salad [232]. Accelerometers have also been used to predict skill level, under different categories, of persons performing a surgical skill, using a

hierarchical stochastic rule induction framework [233]. However, a drawback of sensor-based approaches is that the presence of these sensors is less natural for users and may hinder them in performing the task using their typical technique.

Image-based Approaches

Dynamic graphical models has been used to perform recognition of performed steps of synthetic biology experiments [234] from video data to provide situational support. The performances are weakly supervised with a graphical model constructed of possible paths of actions through the task. A neural network autoencoder is used to encode the image data for use by the graph in segmenting the video sequence into the potential states. This weak supervision is shown to perform well, however it underperforms significantly in segmenting actions relative to supervised training. The Watch-n-Patch system [235] also utilises graphical models as part of a situational support system for ADL activities. This system generates action descriptors, based on skeletal joint motion and image features of objects in use gathered from RGB-D images, and performs clustering of these descriptors to learn distinct actions unsupervised. The activities in this case were composed of up to seven actions, and it remains to be shown how well such a system would perform given an activity composed of many actions.

Summary

Existing sensor-based situational support approaches may interfere with the ability of a person to complete a goal-directed task in a natural manner. The fine-grained nature and self-similarity of the actions involved in goal-directed tasks such as OSCEs may prove challenging for existing situational support systems.

2.2.5 Fine-grained Action Recognition

Context

The ability to recognise component fine-grained actions of a goal-directed activity would enable the development of online pervasive computing applications. The actions involved are human-object interactions with multiple objects in a fixed setting. As such, we review methods that attempt to tackle the specific problem of fine-grained action recognition.

Object-based

Approaches to fine-grained action recognition have focussed on tracking objects and associated motion patterns. Rohrbach et al. [93] tested methods for fine-grained action recognition on a collected dataset of cooking activities. It was found a pose-based descriptor approach, based on Fourier transform features, underperformed relative to

dense trajectories [95]. Stein and McKenna [236] attempted to recognise component fine-grained actions of the complex activity of preparing a salad, using the 50Salads dataset [232], by devising a custom feature descriptor. This descriptor is a histogram of tracklets described relative to the object in use. Ni et al. also focus on tracking objects in use [237] to understand fine-grained actions. In their technique, LSTMs are trained on outputs of pre-trained CNNs to regress bounding box coordinates of objects in successive frames. Support vector machines are used to classify sequences of pooled motion features from the object boxes. A drawback of this technique is that an extensive labelling of the object locations is required for training. Zhou et al. [122] track candidate object proposal regions across an entire fine-grained activity video. A graph of these tracked regions, pruned for consistency, is subsequently mined for mid-level part detectors which can be further pooled to be used as part of a discriminative classification method to identify actions.

Segmental Approaches

Other approaches use global image features to segment entire activity sequences. Kuehne et al. [238] utilise a generative framework to segment fine-grained activities. Video frames are encoded using Fisher Vectors based on dense trajectories, and a hidden markov model, combined with a context-free grammar, is used to segment a sequence. Richard and Gall [239] use a probabilistic model that models the segmentation and classification of actions jointly. A language and length model is combined with a discriminative classifier, and inference is performed using dynamic programming. A drawback of these segmental approaches is that they require observation of the entire sequence, and so use as part of an online application is not straightforward.

Deep Learning Approaches

The rise of deep neural network approaches has also led to the use of CNNs to perform fine-grained action recognition. Lea et al. [19] utilise a VGG-style [39] network to extract image features, and use a 1D convolution over the output image features to classify fine-grained actions of goal-directed activities. The authors also utilise a semi-Markov model to segment a performance video into actions, however this marginally improves accuracy for the 50Salads dataset [232]. Recently, the authors proposed an alternative approach for this segmentation stage [20]. Using an auto-encoder to compress the temporal dimension of input image features and decompress the resulting latent features over the temporal dimension results in an action label for each frame of a video. This approach, however, requires observation of the entire video. Another deep learning based technique for fine-grained action recognition [240] utilised parallel CNN streams on colour and pixel trajectories on subregions of tracked persons and on the whole image. An LSTM is used to classify actions over a larger temporal window based on the output image features of the CNNs.

Recently datasets such as AVA [241], Charades [242] and EPIC Kitchens [176] have been published. These datasets are each large in scale and labelled with many fine-grained actions. These datasets differ from that of a dataset of OSCE performances in that they are captured in a more natural and broader context than the constrained setting of an OSCE. Furthermore, context arguably provides more salient information for recognition of certain actions in these datasets, such as 'martial art', 'swim', and 'drive' in AVA [241]. However, as the novel techniques devised to tackle the recognition challenges of these datasets could be applied to the specific problem of fine-grained action recognition for OSCEs, we review such approaches.

Using spatio-temporal graphs to model fine-grained actions, as interactions between actor and object nodes, is an idea that pre-dates the deep learning era [122, 243]. However, in recent years works have explored approaches using deep neural networks in combination with graph structures [244, 245] Such approaches have been enabled by resurgent research interest in applying neural networks to graph-structured data [246, 247, 248, 249]. These developments have led to a number of works [250, 147, 143] that utilise graphs within a deep neural network to understand fine-grained actions.

The work of Baradel et al. focusses on interactions between actors and objects in a scene [250]. Inspired by research in relational reasoning for problems such as visual question answering [251, 145], this work performs visual reasoning on the object level using time as a causal signal. The proposed Object Relation Network reasons over detected object instances through space and time. Object features (regarding appearance, shape and class) are extracted from pairs of frames (consisting of a current frame and a previous frame) and are reasoned over by a symmetric function over a set of graph cliques of the object representations. Long-range dependencies are learned by passing the outputs of this function to a recurrent neural network to produce a prediction. A base spatio-temporal CNN, as well as being used to produce object appearance features, is used to model global motion and context, with an RNN trained on pooled outputs to predict an action. This prediction is combined by averaging with the object reasoning prediction. In experimentation on datasets such as Epic Kitchens [176], the object-level reasoning is shown to outperform other relational reasoning techniques [145] as well as baseline CNN approaches.

A property of certain types of graph neural networks is message passing between nodes [252]. During training of these networks, a graph representation is learned by propagating information between connected nodes. In a related work by Wang et al. [147], the non-local block was introduced. This non-local block can be inserted into existing CNN architectures, and is composed of a sequence of operations which combines information from an entire input (e.g. spatio-temporal feature volume for video) to produce each element in the output (e.g. a feature map pixel for video). These operations can be made to perform a similar operation to the self-attention mechanism used in natural language processing [253], where non-local information can

influence the output (e.g. the beginning of a sentence can influence understanding of the end). When applied to the problem of action recognition via a deep 3D CNN, non-local blocks facilitate long-range dependencies in space and time to be learned by the model. With multiple non-local blocks, the network is imbued with a message passing capability for this purpose, akin to graph neural networks. The addition of non-local blocks to a 3D CNN architecture is shown to improve performance on the Charades dataset [242]. As discussed in Section 2.2.2, the Long-term Feature Banks approach [146] uses non-local blocks to learn long-term temporal dependencies by combining outputs of an action recognition CNN with pre-computed CNN features from a video. This approach is shown to work well on datasets such as AVA [241], Charades [242] and EPIC kitchens [176].

In a recent work by Wang and Gupta [143], graph structures are used as a video representation to recognise fine-grained actions. Graph nodes, representing objects, are formed by pooling object proposal regions of 3D CNN output features. Two graph representations are used: a similarity graph, with the goal of learning strong edges between instances of the same object over time and between different but semantically related objects; and spatial-temporal graphs, with edges based on object region overlap in space and time. To train the graphs, graph convolutional operations are used in an architecture that aligns with that of Non-local Networks [147]. The outputs of the graph convolution network are concatenated with the pooled outputs of the backbone 3D CNN and used as input to a classification layer. This approach improves performance on the Charades dataset [242], and is complemented by the use of Non-local Networks in the backbone CNN.

The advantages of self-attention mechanisms [253], as used in Non-local Networks, are also exploited in the Video Action Transformer Network [254] for the fine-grained action detection task of the AVA dataset [241]. In this work, object proposal regions are used to extract regions of query features from intermediate Inflated 3D (I3D) CNN features [17]. These features undergo region of interest pooling [255] and preprocessing to give a fixed-size query vector. Embeddings of the entire intermediate I3D features are used as keys and values in a Transformer unit [253] which attends over the spatio-temporal context surrounding the query region. A self-attention mechanism adds the attention-weighted output to the query, which itself is then processed by normalisation and fully-connected layers. The final output of sequential transformer units is used to predict action classes for the query regions. In the analysis of the transformer units, they are shown to learn to track persons semantically, as well as to attend to faces, hands and objects. Classification performance is particularly improved for classes such as 'sailing boat' and 'watching tv' where context information is highly discriminative.

The Actor-Centric Relation Network is a spatio-temporal action detection method that uses the relational reasoning operator of Santaro et al. [145]. Differently to how Wang et al. [144] use this operator for temporal reasoning, this approach uses

a relation operator to combine information from the spatio-temporal video context. Temporal context is captured using a 3D CNN for feature extraction, while spatial context is captured by combining global feature maps with pooled features for actor regions [255]. The global feature maps are divided into grids, with cells akin to object regions, and are combined through convolutions with a grid of repeated actor features. The convolutions learn relations between actors and objects and improves results over baseline for the AVA dataset [241], more so for actions involving objects or multiple actors.

Another recent approach that utilises graph convolutions is the Structured Model for Action Detection [256]. The authors argue that the problem of video understanding requires a structured approach, rather than a generic 3D CNN approach. In their approach, actors are tracked over video clips and relations between tracked actor features and detected object features are reasoned about using graph convolutions. Detected actor bounding boxes are linked to from actor tubelets using an association module trained with a triplet loss based on actor appearance. I3D features [17] are extracted for each of the actor tubelets and for the detected objects. Two graphs are formed to reason about human-object and human-human interactions respectively using these features. To make the graph learning tractable, a single generic object class is used. The experimental evaluation shows how this structured approach improves performance over a generic baseline, particularly for actions involving interactions with objects and actions with long-term temporal dependencies.

A structured approach to action recognition was further investigated in a recent work, Action Genome [257]. This work draws on findings from cognitive science [258] that people encode activities into "consistent hierarchical part structures". The authors extend scene graph representations to the temporal domain to represent such compositional structures. Scene graphs are graph structures that encode relationships between objects, and have been combined with vision models to perform various image tasks [259]. In the video scenario of Action Genome, scene graphs encode relationships (attention, spatial, and contact) between actors and objects, and are predicted for video frames and used to populate a feature bank. Features from the feature bank are combined with I3D [17] outputs using an operator as per Long-term Feature Banks [146] (i.e. non-local blocks [147]). Scene graph labels are added to the Charades dataset [242] to facilitate study of scene graphs for video understanding. In experimentation, the Action Genome approach is shown to improve action recognition performance on Charades [242]. Furthermore, the spatio-temporal scene graph, with its compositional approach to actions and symbolic embedding of visual information, is shown to improve few-shot recognition capability.

Besides improvements due to the use of graph mechanisms, designing backbone CNN models is another area of active research that has led to improved action recognition performance. This research focusses on a number of issues identified in existing

approaches. One issue is the use of hand-designed optical flow features as a motion representation. Some recent works propose techniques where, instead of using optical flow, the network architecture is constructed to learn to represent motion from an input colour video clip [260, 261, 262]. Another issue that is addressed is the large computational cost of video models [263, 264, 265, 261, 148]. Such architectures may consist of a mixture of 2D and 3D CNN layers, where the reduction in 3D layers leads to less computational requirements [265, 261, 263]. One novel approach, STM (also discussed in Section 2.2.2) uses shifting operations to give 2D convolution operations access to neighbouring frame features [148]. One other issue addressed is that 3D network design is commonly inherited from 2D architectures designed for image recognition tasks [17, 147]. Commonly in these approaches [266, 267, 268, 269], the blocks of existing architectures are re-designed to better suit the task of video understanding. Other works [270, 271] add additional layers to such CNN architectures to make it more suitable to video tasks, such as Timeception [271], which uses multi-scale temporal convolution operations on outputs of a 3D CNN [17] to allow learning of long-range temporal dependencies for datasets such as Charades [242].

A recent work by Feichtenhofer, X3D [272], aims to tackle the question of whether using existing 2D architectures, expanded to 3D, is the optimal strategy for designing action recognition architectures. Furthermore, questions are tackled regarding interactions between parameters, such as spatial resolution, temporal sampling, network depth and width. This work expands a basis architecture into 3D across a number of architectural and sampling parameters. The basis architecture is based on a ResNet design, and is similar to the Fast branch of the SlowFast architecture [141]. It uses channel-wise separable convolution layers for efficiency purposes. The parameters along which the architecture is expanded are the temporal stride, temporal sample size, spatial sample size, network depth, layer width (i.e. number of channels) and the width of the bottleneck in each residual block. One interesting finding is that at a low-complexity level (i.e. low computation), high spatio-temporal resolution is favoured over wide channel dimensions. In terms of performance, the expanded models are shown to match state-of-the-art on Charades [242] and AVA [241], while requiring significantly less computation.

Summary

The wide variety of discussed approaches shows that there are no consensus techniques of recognising fine-grained actions in a fixed setting. The approaches to the fixed setting scenario frame the problem differently in cases, specifically looking at models of segmenting an activity sequence or classifying frames under supervision. We have also discussed approaches that use graph mechanisms to reason over actors and objects for fine-grained action recognition tasks in an unconstrained setting. The contributions in this thesis are focussed on utilising multiple image types to perform fine-grained

action recognition, a direction that has received less research focus. For example by utilising depth information, it is possible to calculate scene flow, which may provide a richer, more stable, representation for learning spatio-temporal patterns of fine-grained actions. In future work, it may be possible to combine the multi-modal approaches explored in this thesis with graph-structured approaches.

2.2.6 Relevant Datasets

Context

To assess fine-grained action recognition techniques as suitable for the specific case of OSCEs, it is necessary to collect a dataset of such performances. Given the fixed setting of an OSCE, there is less utility in the wider contextual information due to the similarity of each of the actions and the relative constancy of the background. Therefore existing large action recognition datasets [273, 274, 241] that encompass a range of activities in numerous contexts, such as sports, are not directly applicable in our case.

Fine-grained Datasets

In our case, we are specifically interested in datasets that involve fine-grained actions in a fixed setting. A dataset that encompasses fine-grained actions is that of MPII Cooking [93]. This dataset consists of a large number of kitchen actions being performed, with a single stationary colour camera focussed on the kitchen surface. This dataset also includes estimates of the poses of the humans, and estimations of dense trajectory features. This dataset has received particular attention in research attempting to classify fine-grained actions [122, 275]. Other fine-grained action recognition datasets have focussed on particular application areas, such as the Activities of Daily Living (ADL) dataset [276] which is composed of ego-camera recordings of people performing everyday activities. Another application focussed dataset is the MERL Shopping Dataset [240] which is composed of recordings from an overhead camera of people shopping for groceries from shelving units. These datasets are all recorded using a single colour camera. In the case of an OSCE, it is not clear whether a single colour camera can feasibly capture all the important granular steps of a performance.

The JIGSAWS dataset [277, 278] is a dataset of performances of three different robotic surgical tasks: suturing; knot-tying; and needle-passing. The data is composed of overhead video of the surgical implements and task area as well as kinematic measurements, such as position and velocity, from sensors attached to the implements. It is annotated with action segments for actions such as 'insert needle into skin', 'tie a knot' and 'drop needle at finish'. Additional annotations includes manual assessments of surgical skill levels by an expert surgeon. The dataset has been used to benchmark action segmentation and recognition techniques that use video data [19], kinematics

data [279], and combinations of the two modalities [280, 281]. It has also been used to benchmark techniques that perform automated skill evaluation [282]. The scenario of an OSCE is less constrained than that of robotic surgical tasks, with multiple possible approaches to performing actions, thus making action recognition arguably more challenging than it is for the JIGSAWS dataset.

RGB-D Datasets

Action recognition from multi-modal data, such as RGB-D, has also been performed and a number of related datasets have been collected. The MSR Action 3D dataset [228], the MSR Daily Activity 3D dataset [283] and the MSR Action Pairs dataset [182], all consist of RGB-D data recorded using a Kinect camera, as well as human pose information obtained using the Kinect [8]. In the MSR Action 3D dataset, the actions performed are broad full body actions such as throwing, jumping, reaching. The MSR Daily Activity 3D dataset [283] consists of actions such as 'use vacuum cleaner', 'lie on sofa' and 'use laptop'. The MSR Action Pairs dataset [182] includes actions that involve object interaction, however the actions are coarse-grained such as 'pull chair', 'pick up laptop', 'put on hat'. Other RGB-D datasets have focussed on capturing actions from multiple viewpoints [284, 285], to enable development of techniques of recognition from new viewpoints. The coarseness of the actions in these datasets, and the brevity of action clips, contrasts with the fineness, varying length, and consecutiveness of the actions involved in OSCEs. Recently, the NTU RGB-D dataset [286] was released with a focus on having a larger number of action classes, from many viewpoints. However the action classes are still broad human motions, such as 'punching', 'hugging', 'eating' and 'reading'. A dataset that most closely matches the OSCE setting is the 50 Salads Dataset [232]. This dataset consists of RGB-D data, recorded using a single depth camera, of people preparing a salad in a fixed kitchen setting. The action classes are labelled at coarse and fine granularities.

Summary

To our knowledge, there does not exist a dataset of fine-grained actions performed as part of a goal-directed activity that is recorded from a number of viewpoints. Additionally, there does not exist such a dataset that also incorporates depth information. By providing such a dataset to the research community, the variation of the types of techniques that can be applied can be analysed for relative performance. For example, 2D fine-grained techniques can be compared to 3D techniques as well as to multi-view techniques. Such a dataset would also provide the possibility of development of techniques for recognition from novel viewpoints. This dataset would provide a useful benchmark for assessing recent techniques of 3D action recognition and fine-grained action recognition. Given the usefulness of recognising the actions in an OSCE to possible pervasive computing applications, this dataset can form the basis of future

research in this direction.

2.3 Vision for Skill Determination

In this thesis, we are focussed on recognising fine-grained actions, specifically those involved in OSCEs. Recognising the fine-grained actions of an OSCE would allow online applications to validate the completion of the necessary steps of an OSCE, as discussed in Section 1.1. Works that perform skill determination are relevant due to their ability to give a score or ranking to a skill performance, and we review such works here.

Surgical skills, such as those captured in the JIGSAWS dataset [277, 278], have formed the target domain for many skill determination techniques [287, 288, 289, 290, 282, 291, 292]. A novel formulation of Hidden Markov Models is used for temporal modelling by Zhang et al. [289] to perform pairwise ranking for surgical skills. Another work analyses extracted sequential information in the frequency domain to determine motion quality and hence predict a skill assessment score [287]. Similar analysis of sequential information using entropy metrics can capture fluency and regularity of movements and hence predict skill levels [290]. However, the constrained setting of surgical skills makes these approaches difficult to apply to other less constrained settings such as OSCEs.

Sports is another domain in which which skill determination techniques have been applied [293, 294, 295, 296, 297]. In a work by Pirsiavash et al. [293], low-level and high-level (i.e. pose) spatio-temporal features are combined for use as inputs to a support vector regression model to predict judging scores for olympic events. Performance on this skill determination task was improved by Parmar et al. [294], who used CNNs to extract visual features and LSTMs to produce a prediction. In a work by Bertasius et al. [295], skill assessment is performed for first-person basketball videos. A convolutional LSTM is used to detect atomic events in videos, and a feature is constructed using a Gaussian mixture model which is then used to predict pairwise ranking of players. The detailed interaction between a performers joints are focussed on in the work of Pan et al. [298]. Here, convolutional features for joint regions in images are used to build relational graphs based on spatial and temporal relations, and graph convolution operations are used to output features for score prediction. Such skill determination for sports techniques may be challenging to apply in an OSCE scenario, where action quality may be secondary to criteria such as performing steps in the correct order.

In the work of Doughty et al. [299], a technique was devised that can be used to determine skill level for more general domains. The approach uses a pairwise deep ranking model to determine differences in skill level. By training a siamese CNN architecture of spatial and temporal streams with a ranking loss based on human annotated skill rankings, the model can predict a skill ranking for a video. This approach is shown

to work well across different tasks, such as surgical skills and pizza making, including for newly collected datasets. In a later work by the same authors [300], ranking performance is improved by using temporal attention modules. Separate modules attend to parts of a video that exhibit higher and lower levels of skill. A novel loss function is used that combines ranking losses from each of high-skill, low-skill and uniformly weighted network outputs. Another recent work by Parmer et al. [301] find that skill determination for sports is improved when trained using multi-task learning. Auxiliary tasks include fine-grained recognition of aspects of the skill performance (e.g. number of somersaults for diving) and caption generation.

In these discussed works, fine-grained action recognition may not be viewed as an essential step to perform skill determination. However, in the context of an OSCE performance, the level of perceived skill of a performer does not determine whether the task was performed correctly. In each OSCE, the sequence of actions is crucial for correct performance. For example, in the OSCE of cannulation, a person could perform all necessary actions fluently, including successfully inserting a canula into a vein, however if they checked for a vein (i.e. touched the site) after the cannulation site has been sterilised they would fail the assessment due to causing risk of infection [6]. Fine-grained recognition of the actions in an OSCE performance facilitates accurate segmentation, and hence can be used to determine if the correct seqeunce was followed.

2.4 Summary

In this literature review, we have identified the following problems that will be addressed by the contributions within this thesis:

- There has not been an investigation into the development of action recognition techniques for the specific domain of OSCEs.
- Incorporating temporal information, in the forms of both motion (i.e. flow information) and sequential data (i.e. multiple image frames), into a fixed-setting fine-grained action recognition system has not been extensively investigated;
- Effective utilisation of multiple image types, such as colour and depth, to better recognise fine-grained actions in a fixed setting requires further investigation; and
- The collection of a dataset of fine-grained actions performed as part of a goal-directed task in a fixed setting that includes depth and colour information from multiple viewpoints has yet to be performed.

Chapter 3

How do you take yours? Action Recognition Using 3D Tracked Poses and Recurrent Neural Networks

In this chapter, a system for recording human-object interaction tasks is designed, implemented and evaluated. An examination of the motivating factors of the system is performed and used to inform the system design. This system enables recording of fine-grained actions from multiple viewpoints using multiple image modalities, in a synchronised way. Thus it permits varied approaches to the problem of understanding human-object interactions. We contribute this system for use by other researchers by providing a code framework that makes re-implementation feasible. Using the capture system, a dataset of performances of a goal-directed activity, preparing a cup of tea, is recorded and labelled. Techniques for tracking the performer's hands and the objects in use are detailed. An action recognition approach based on Long Short Term Memory neural networks is described. A grid search methodology for the architecture and hyperparameters of the action recognition network is presented.

3.1 Introduction

3.1.1 Motivation

In this thesis, our research goal is to recognise the constituent fine-grained actions of a fixed-setting goal-directed activity, such as an Objective Structured Clinical Examination (OSCE). Recognition of the actions, and hence steps performed, can form the basis of pervasive computing applications (Section 1.1). In this chapter, we use an example task of preparing a cup of tea to explore methods of constructing a pipeline for performing this recognition. This task is a goal-directed activity involving distinct

steps to be performed in an order, and so can serve as a proxy for other activities, such as an OSCE.

The possibility of using fixed arrangements for the system allows greater flexibility in types of cameras used and placement of the cameras (Section 1.3.1). Since the release of consumer RGB-D cameras, such as Microsoft Kinect, the research community has utilised the extra depth information provided to make significant advances in the problem of tracking humans at various levels, such as at the body level [8] and hand level [9]. Object recognition and pose estimation is another area where the availability of depth data has enabled new techniques [10, 302]. These and newer techniques may allow for real-time tracking of objects and people, providing valuable input information for action recognition systems. In work by Pirsiavash et al. [276], it is suggested that recognition of everyday activities is "all about the objects". Thus we focus on estimating object poses for use in recognising actions involved in a goal-directed activity. Estimated human pose information is also used for this recognition purpose.

Accurate estimation of object poses, which can be facilitated by the capturing system outlined in this chapter, has potential benefits for areas beyond action recognition. One particular area that benefits from object pose information is robotics. A method based on the dataset and action recognition technique of the Watch-n-Patch system [235] (discussed in Section 2.2.4) uses object pose information as part of a robot assistant that observes a performance of an Activity of Daily Living and reminds the subject of a missing step by laser-pointing to the relevant object [303]. Object grasping is a common robotics task that can be improved by accurate estimation of object pose [304, 305]. Availability of the full object pose allows use of geometry information that may not be observable by the robot [304]. A large dataset for this vision-based task was recently made available that comprises RGB-D images with over a billion possible grasp poses for objects [306]. Another scenario in which a robotics task can benefit from object pose information is collaborative robotics in manufacturing. Here, pose information can be used by a robot to understand a user's intent and hence aid them in their task [7] The recording framework described in this chapter could be used in each of these scenarios to improve object pose estimation and hence improve performance of these robotics tasks.

A system for capturing the goal-directed activities is required. The design choices of this system are motivated by the following factors:

1. The system must **facilitate the recognition of fine-grained actions**, such as those involved in OSCEs.

2. All key actions of a performance must be **clearly observable** to allow for the classification of correct performance of these key actions (e.g. cannula insertion for OSCEs).

3. It is necessary that the **physical space accommodates the performance of**

the activity. Typically, an OSCE, such as Peripheral Intravenous Cannulation, can be performed on a surface the size of an office desk.

4. The physical set-up of the system should **not detract from a person's ability to perform the activity**. Other systems require sensors to be placed on objects being interacted with [232, 231]. The presence of sensors would interfere with subjects naturally using the objects.

5. The system should allow for **repeatability**. It should be feasible to re-implement. It should not rely on the overly strict placement of sensors and objects.

In this chapter, we present a capture system satisfying these factors, which enables six degrees of freedom (6DOF) object and human pose estimation from recorded video data. The problem remains, however, of how to use this pose information to recognise the actions of a human-object interaction. We demonstrate the use of recurrent neural networks to recognise actions based on sequences of object and human pose information. Specifically, we use the Long Short Term Memory (LSTM) neural network cell due to its success in other sequential modelling tasks [127, 307]. Our results show that an architecture composed of LSTM cells can recognise actions from sequences of pose information more reliably than established benchmark algorithms.

3.1.2 Contributions

The contributions made in this chapter are:

- We contribute a system for recording performances of human-object interactions. The system's novelty lies in the fact that it enables recording of activities from multiple viewpoints using RGB-D data, in a synchronised way.

- A code framework is also contributed, allowing other researchers to feasibly re-implement this framework and to devise efficient data processing pipelines using the framework, located at https://github.com/leaveitout/pcltools.git.

- A pipeline for action recognition of a human-object interaction, based on tracked object and human poses, is devised using this system and framework.

- A multi-camera RGB-D dataset, 'Cup of Tea', of performances of the goal-directed activity of preparing a cup of tea is presented. Models of objects used are included with the dataset so that other object pose-based techniques can be implemented.

- A method is presented of using grid search to identify a neural network architecture, involving LSTMs, that maximises action recognition from pose information. For the Cup of Tea dataset, the optimal architecture achieves 83% frame-level accuracy.

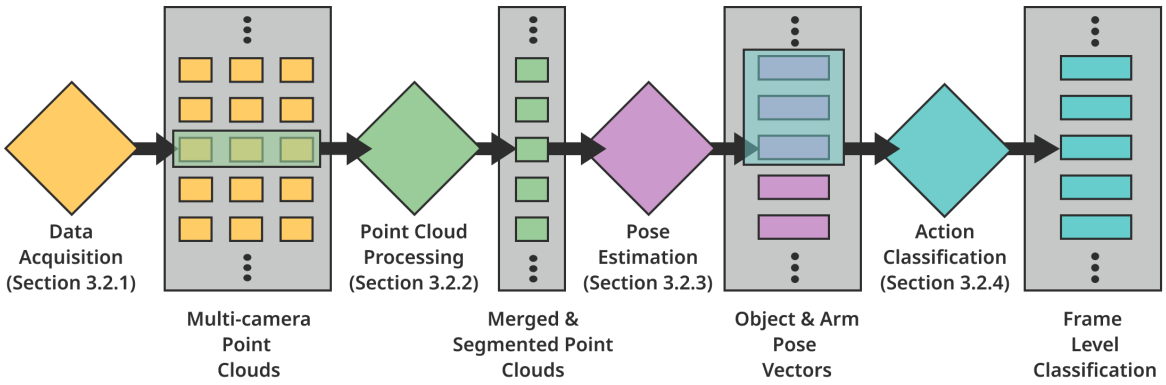


Figure 3.1: A proposed pipeline for the recognition of actions of a human-object interaction.

3.2 Design

In this section, we shall describe the individual stages of an overall action recognition pipeline shown in Figure 3.1.

3.2.1 Data Acquisition

Camera Arrangement

The possibility of using fixed arrangement of the system allows greater flexibility in types of cameras used and placement of these cameras. Based on an analysis of different candidate arrangements, we select a multiple RGB-D camera arrangement, as shown in Figure 3.2. This arrangement minimises possible occlusions of salient information that may affect later pose estimation stages. Another benefit of multiple RGB-D cameras is that it facilitates many future alternative action recognition approaches (e.g. volumetric approaches, multi-view approaches). A thorough analysis of candidate arrangements is performed in Appendix A.

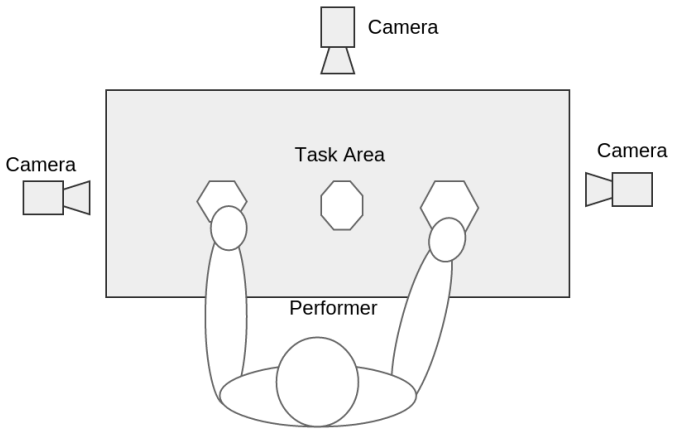


Figure 3.2: A multi-camera set-up including a number of fixed viewpoints.

Algorithm 3.1 Synchronisation Algorithm

procedure SYNCHRONISE DATA FRAMES
  Let $x_{t_i l}$ be a data frame from camera l at time t_i.
  Let $\mathcal{S}_p$ represent the p^{th} set of synchronised data frames , for $p = 1, 2, \ldots$.
  Let $\mathcal{T} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots\}$ represent the set of sets of synchronised data frames.
  Let c be the total number of cameras.
  Let τ represent an arbitrary amount of time.
  if $x_{t_j l} \in S_p$ for some j **or** $t_i - \min(t_j : x_{t_j m} \in \mathcal{S}_p) > \tau$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{S}_p\}$
    $\mathcal{S}_p \leftarrow \{x_{t_i l}\}$
  else
    $\mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{x_{t_i l}\}$
    if $|\mathcal{S}_p| = c$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{S}_p\}$
      $\mathcal{S}_p \leftarrow \varnothing$
    end if
  end if
end procedure

Synchronisation

The data received from multiple RGB-D cameras may not necessarily be synchronised across the cameras, and so a synchronisation method is required. This method should maximise temporal consistency of the data between the cameras so that the outputs of later merging stages exhibit fewer artefacts. To maximise consistency, an algorithm (Algorithm 3.1 below) was developed that associates data frames from each of the cameras with their nearest temporal neighbours. The algorithm also ensures that there always exists a full complement of frames in each set of frames.

Camera Calibration

Accurate extrinsic calibration of the RGB-D cameras will provide better quality data for later pose estimation stages of our pipeline. It is possible to perform this calibration using only colour images from the cameras and a known planar target. The quality of such a calibration is dependent upon the images of the target used. Due to the camera arrangement, it can be challenging to obtain good quality images of the calibration target from two cameras simultaneously. It is also possible that the positions of the cameras may change slightly over time, thus requiring recalibration. Therefore, a more automated technique for extrinsic calibration was sought.

The availability of the depth data permits use of a greater variety of techniques to perform extrinsic calibration. Our calibration technique involving depth data is composed of a coarse estimation stage and a refinement stage. The coarse estimation stage is required upon initial arrangement of the cameras, and the refinement stage can be used periodically to correct any slight changes in camera position.

Our extrinsic calibration technique operates on 3D information, and so it is neces-

sary to calculate 3D positions based on depth information. A *point cloud* is a set $\mathcal{P}$ of points, $\mathbf{p} \in \mathbb{R}^n$, representing 3D positions. Given intrinsic calibration parameters of the depth cameras, a point cloud can be calculated for each depth image. Furthermore, if the extrinsic calibration is known between the depth and colour sensors of an RGB-D camera, points in the point clouds can be associated with corresponding colour values.

An extrinsic calibration of two cameras estimates the transformation, $\mathbf{T} : \mathbb{R}^4 \to \mathbb{R}^4$, that maps points from one camera's reference frame to the other. To estimate this transformation between two point clouds, $\mathcal{P}$ and $\mathcal{Q}$, there must exist points $\mathbf{p}_i \in \mathcal{P}$, such that $\tilde{\mathbf{p}}_i = \mathbf{T}\tilde{\mathbf{q}}_j$, for some $\mathbf{q}_j \in \mathcal{Q}$, where $\tilde{\mathbf{p}} = [p_x, p_y, p_z, 1]^T$ is the homogeneous representation of $\mathbf{p}$. These known correspondences can be used in a cost function, such as the sum of square differences, to estimate $\mathbf{T}$ using nonlinear least squares methods,

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \left\{ \sum_{i=1}^{n} \left(\tilde{\mathbf{p}}_i - \mathbf{T}\tilde{\mathbf{q}}_i\right)^T \left(\tilde{\mathbf{p}}_i - \mathbf{T}\tilde{\mathbf{q}}_i\right) \right\}. \tag{3.1}$$

To find correspondences between point clouds, we identify large custom targets in the corresponding colour images using Harris corner detection and contour detection to locate specific keypoints. As each camera has colour and depth extrinsically calibrated, the 3D positions of these keypoints can be calculated.

As the number of correspondences for the coarse estimation is small due to limited targets used, it is possible that the transformation may be slightly inaccurate. A refinement stage is used to reduce this inaccuracy. This is performed using the Iterative Closest Point algorithm [308]. This algorithm minimises a heuristic by taking locally optimal steps, i.e. a greedy algorithm. The heuristic we use is a measurement of distances from points to planes at the nearest corresponding points,

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \left[\sum_{i=1}^{n} \mathbf{1}_{\mathcal{D}}\left(||\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i||\right) \left(\tilde{\mathbf{n}}_i^T \left(\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i\right)\right)^2 \right]. \tag{3.2}$$

Each component of the sum is weighted by an indicator function on the closed set $\mathcal{D} = \{d \in \mathbb{R} \text{ where } ||d|| \leq d_{max}\}$ that indicates whether a point has a close correspondence. The vector $\tilde{\mathbf{m}}_i$ represents the closest point in $\mathcal{P}$ to the transformed point $\mathbf{T}\tilde{\mathbf{q}}_i$, and $\tilde{\mathbf{n}}_i$ is the normal at point i. Further details of the camera calibration technique can be found in Appendix B.

3.2.2 Point Cloud Processing

The extrinsic calibration of the respective cameras provides a method of transforming the point clouds received from the left and right cameras to the coordinate reference frame of the central camera. When fusing multiple point clouds like this, some issues are encountered. Small errors in the transformation estimates may result in slight misalignments of edges and surfaces in the merged point cloud. Another issue is that edges and surfaces in the merged point cloud may be over-represented as a result of

being observed by multiple cameras. This issue leads to a varying density of points for different surfaces. As such, a method to re-estimate point positions based on multiple samples may provide a more accurate representation.

Moving Least Squares [309] is a method that can be used for generating a smooth interpolated output point cloud from scattered input point clouds [310]. The technique, which has its origins in differential geometry, generates samples by approximating a sampled surface with a polynomial function for a small local region. As this algorithm can be used to tackle the issues discussed, we employ it to improve the quality of the merged point cloud data. In our case, points in each local region are weighted using a Gaussian, and the surface is approximated using a polynomial of degree 2.

The output point cloud of this smoothing stage will include points that are not useful for later pose estimation. Using the known arrangement of the task table (Figure 3.2, it is possible to segment this point cloud to include only points that lie above the table, isolating points that relate to objects in use and the human performer's arms. The RANSAC algorithm can be used to find a table in a point cloud [311]. This algorithm iteratively fits a model of a plane to sample set of potential inlier points using a least squares estimate. This algorithm has the advantage of being resistant to outlier points (e.g. scenery besides the task table) present in the merged point cloud. Once the largest plane has been identified (i.e. the task table), Euclidean clustering [312] is used to isolate table points from other points that lie on this 3D plane. To identify all points above the table, we construct a convex hull around the table cluster and check whether a point lies within a polygonal prism extruded above this hull. The output of this processing stage should include only points relevant for pose estimation.

3.2.3 Pose Estimation

Object Pose Estimation

The technique for estimating object poses is composed of two stages: estimating the initial pose at the beginning of a video; and tracking the object through the remainder of the video. The first stage identifies an object on the task table using the Linemod technique [10]. It is assumed that all the necessary objects will be placed upon the table, prior to the task. This technique combines colour gradients and surface normals to generate templates for known objects which can be efficiently searched. This technique has the benefit of working for objects that may have little surface texture, or are partially occluded. The templates are collected by performing three-dimensional scans of the objects and calculating the templates of the objects at different possible poses. By encoding poses with each template, the six degrees of freedom (6DOF) pose of the object can be estimated.

The second stage tracks an object's 6DOF pose as the video progresses. It is possible to use Linemod to estimate poses in each frame. However, as Linemod uses single RGB-

D frames, it may become unreliable when an object is significantly occluded in a frame. Given an initial pose, a 3D registration technique can be used to fit object poses frame-to-frame using the merged and segmented point clouds, based on the assumption that objects can only move small distances between successive frames. To perform this registration, we use the Generalised Iterative Closest Point (GICP) algorithm [313], an extension of the point-to-plane algorithm described in Section 3.2.1. This algorithm uses a probabilistic model for a point-to-point cost function, which is more robust to incorrect correspondences than other iterative closest point algorithms [313]. These incorrect correspondences are more likely to be present when registering objects of complex geometry than in the previous case of camera calibration (Section 3.2.1). Point clouds of scanned objects can be used to perform the registration. To ensure consistent densities of points across the source and target point clouds, the points are filtered using a voxel grid of fixed cell dimensions before registration.

Given the initial pose of the object, estimated using the Linemod algorithm, and the incremental poses, found via registration of the scanned version of an object transformed to the previous pose, a series of poses for each object will be produced. The poses are encoded as a transformation matrix, $\mathbf{T} : \mathbb{R}^4 \rightarrow \mathbb{R}^4$. In homogeneous form, this transformation matrix is composed of a rotation, $\mathbf{R} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ and a translation $\mathbf{t} \in \mathbb{R}^3$, $T = [\mathbf{R}, \mathbf{t}; \mathbf{0}^T, 1]$. However, in engineering a feature to use based on these values, utilising a larger parameter space makes learning more challenging [314]. A more dimensionally compact representation of a transformation can be found by using unit quaternions to encode the rotation. Thus each pose can be encoded as $[\mathbf{t}, a, b, c, d]$, where the rotation is encoded by the unit quaternion $\mathbf{z} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, $a, b, c, d \in \mathbb{R}$.

Arm Pose Estimation

To estimate an arm pose, we identify points corresponding to subjects' arms in the merged and segmented point clouds. Using the estimated object poses, we remove points that lie within a threshold distance of any points of the transformed object point clouds. Due to noise and slight misalignments of objects, it is necessary to perform further segmentation. Under the assumption that arm points all lie within a certain distance of each other, Euclidean clustering [312] is used to identify the two largest clusters of the remaining points. A lower bound on possible cluster size ensures that only clusters large enough to be arms are selected.

We estimate the arm poses from the cluster information. We denote a candidate arm cluster as $\mathcal{Q} = \{\mathbf{q}_i \in \mathbb{R}^3, i = 1, 2, \ldots\}$. We represent a pose using three components: $\mathbf{a}_1 \in \mathbb{R}^3$, the inner arm point in the task area; $\mathbf{a}_2 \in \mathbb{R}^3$, a point representing the position of the subject's wrist; and $r \in \mathbb{R}$, the width of the subject's arm as it appears to the central camera sensor. The rationale behind using the inner arm point is that it will contain information, in combination with object pose information, indicating which object is being used. The rationale behind the outermost arm point is that it will

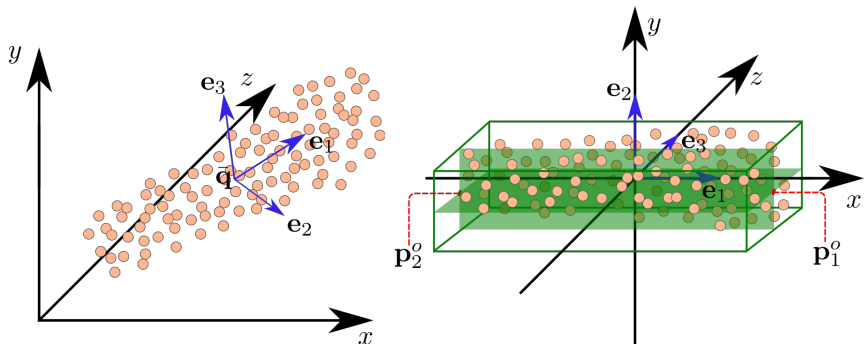


Figure 3.3: Left, a cluster centroid, $\bar{\mathbf{q}}$, and the three eigenvectors, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, is shown for an arm cluster. Right, the centroid is subtracted, and the eigenvectors are axis-aligned. The centres of the two axis-aligned bounding box faces perpendicular to the principle axis, p_1^o and p_2^o, are used as the two end effectors for the arm pose.

contain information indicating the type of interaction that is being performed based on the angle between the two arm points. The rationale behind the width feature is to capture information to allow discrimination between arm poses of different rotations about an axis through the arm, such as those involved in turning motions. To calculate these features, the extents of an oriented bounding box over the cluster set $\mathcal{Q}$ are examined.

To determine these bounding box extents, Principle Component Analysis (PCA) is used [315]. The centroid, $\overline{\mathbf{q}}$, is subtracted from each point, $\mathbf{q}_i$ in a candidate cluster. A data matrix, $\mathbf{X}$ is constructed, with each row representing a mean subtracted point. The eigenvalues of the matrix $\mathbf{X}^T\mathbf{X}$, are calculated along with the corresponding eigenvectors, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, representing the principle components of the cluster set (see Figure 3.3, left). These eigenvectors can be composed into an orthogonal rotation matrix that transforms the three principle axes of greatest variance to the Cartesian axes, $\mathbf{R} = \left[\mathbf{e}_1^T, \mathbf{e}_2^T, \mathbf{e}_3^T\right]$. The new data matrix, $\mathbf{Y}^T = \mathbf{R}\mathbf{X}^T$, facilitates the determination of the bounding box extents. The maximal and minimal extents along each row of this matrix define the axis-aligned bounding box for the set of transformed arm points. The centres of the two faces perpendicular to the principle axis (see Figure 3.3, right), $\mathbf{p}_1^o$ and $\mathbf{p}_2^o$, can thus be found. Thus, the inverse transformation, $\mathbf{a}_i = \mathbf{R}^T\mathbf{p}_i^o + \bar{\mathbf{q}}$, can be applied to these points, to transform them back to the original reference frame, to get the desired end effectors. The final feature component is the width of the observed arm cluster, r, corresponding to the bounding box width.

The two clusters centroids are inspected to distinguish left from right arms. If a single cluster is detected, geometrical rules based on the position of the arm centroid and the angle of the arm relative to the camera are used to determine handedness. The threshold position and angle for these rules can be defined based on the known scene arrangement.

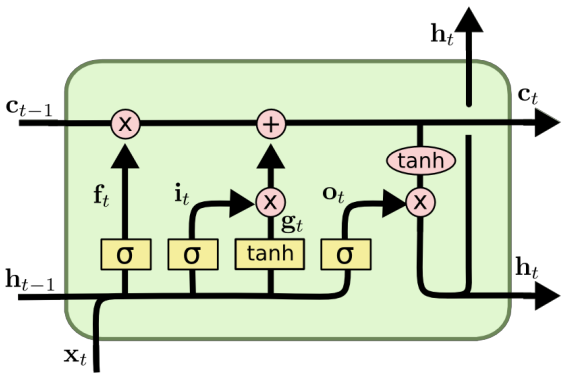


Figure 3.4: The Long Short Term Memory cell includes a **c** variable that is added to during the cell update. Nonlinear activation functions control the information removed (with a forget gate) and added (with a memory gate) to this cell value. Figure is adapted from original, with permission [316].

3.2.4 Action Classification

Given the individual feature components described in Section 3.2.3, the feature vector is defined as the concatenation of the object features, $[\mathbf{t}_i, a_i, b_i, c_i, d_i]$ for each $i \in \{cup, pot, bowl, jug\}$, and the arm features, $[\mathbf{a}_{1,o}, \mathbf{a}_{2,o}, r_o]$ for each $o \in \{left, right\}$. The dynamics of these poses, and the relationships between them, will include information regarding the current action being performed. For example, a teapot being moved towards a cup may indicate a tea pouring action. A sequence of pose features is used to capture temporal dynamics for classification.

Recurrent neural networks (RNNs) have proven useful in sequential learning tasks, such as in the area of natural language processing. Recent work has looked at utilising RNNs to learn from sequences of image features extracted from videos to perform action recognition [127]. Long Short Term Memory Networks (LSTM) are a type of RNN structure designed to overcome the vanishing gradient problem [51], discussed in Section 2.1.2. This is achieved by allowing old (potentially useless) information to be forgotten and new (useful) information to be recorded in the cell state.

An LSTM cell, as shown in Figure 3.4, is updated with the following equations:

$$\mathbf{i}_t = \sigma([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_i}); \tag{3.3}$$

$$\mathbf{f}_t = \sigma([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_f}); \tag{3.4}$$

$$\mathbf{o}_t = \sigma([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_o}); \tag{3.5}$$

$$\mathbf{g}_t = \tanh([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_g}); \tag{3.6}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \cdot \mathbf{f}_t^T + \mathbf{g}_t \cdot \mathbf{i}_t^T; \tag{3.7}$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \cdot \mathbf{o}_t^T. \tag{3.8}$$

The memory of a cell, $\mathbf{c}_t$, is controlled by a series of gates during each update. The forget gate, $\mathbf{f}_t$, controls the amount of information to be forgotten via a nonlinear sigmoid. Similarly, the input gate, $\mathbf{i}_t$, and input modulation gate, $\mathbf{g}_t$, control the

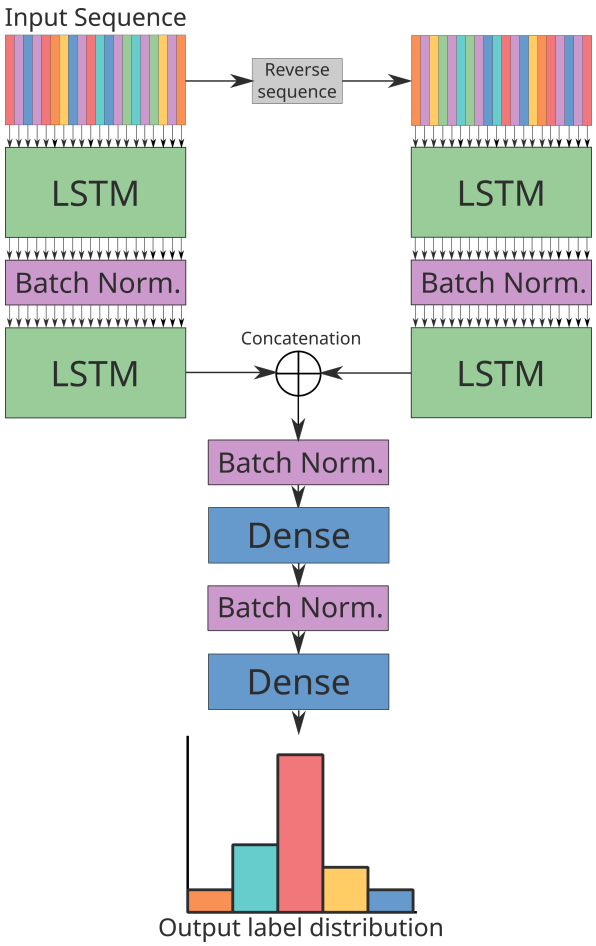


Figure 3.5: An example architecture used to classify sequences of pose information into action labels.

amount of new information that is allowed into the cell state.

The LSTM cells are used as part of a larger neural network architecture to perform classification of sequences of pose features. The recognition performance is dependent upon the architecture, and so different architectures need to be tested. Decisions regarding the architecture include the number of LSTM layers to use and the number of cell units per layer, i.e. $|\mathbf{c}_t|$. Previous work has shown that reversing a sequence of data and feeding it to a separate LSTM cell increases performance for tasks such as speech recognition [317, 318]. Decisions on whether to train LSTM cells on forward sequences or both forward and reverse sequences, illustrated by the 'reverse sequence' operation in Figure 3.5 also need to be made. These decisions are made using an automated procedure of identifying an optimal architecture, which will be discussed in Section 3.3.4.

In our architectures, the output of the LSTM layers undergoes further transformations via a series of fully connected, or *dense*, layers. The output of the layers undergo an activation function, with intermediate dense layers outputs undergoing a Rectified Linear Unit (ReLU) activation [44], $f(x) = \max(0, x)$. The outputs of the final layer undergo a softmax activation to produce an output distribution across the possible label classes, $\sigma(\mathbf{x})_i = e^{x_i} / \sum_{k=1}^{K} e^{x_k}$, where $\mathbf{x} \in \mathbb{R}^K$.

Neural networks are prone to overfitting to the training set, and hence, it is necessary to deploy regularisation techniques to overcome this. Dropout is an established technique that has been utilised for regularisation purposes of neural networks [36], whereby connections between neurons are randomly dropped to discourage co-adaptation of neurons. Dropout is applied to the inputs of each of the dense layers for regularisation purposes. For the LSTM cells, a variant of dropout for recurrent neural networks is used, known as recurrent dropout [319], whereby the dropout mask is identical across each of the timesteps in a sequence. Finally, batch normalisation [37] is used between the layers of the network to counteract internal covariate shift between the layers of the network.

3.3 Implementation

3.3.1 Data Acquisition

Figure 3.6: The arrangement of the recording set-up. Top-left image shows a vibration motor attached to an RGB-D camera (Asus Xtion). Right image shows the placement of the cameras in relation to the task table. Bottom-left image shows a colour image taken from the centre camera during a recording.

A recording system was constructed according to Figure 3.2. This set-up involves an office desk, serving as the task area, being observed by three depth sensors as per Figure 3.6. The cameras used, Asus Xtion Pro Live cameras, provide streams of both colour and depth information. A structure was erected to support the cameras and ensure they remain relatively stationary. Adjustable camera mounts allow fine-tuning of the camera's pose to ensure that each observed the entire task area. The cameras are all connected to a single nearby PC via USB.

Using specific camera driver settings, it was possible to ensure that the colour and depth information from an individual camera was received synchronously by the

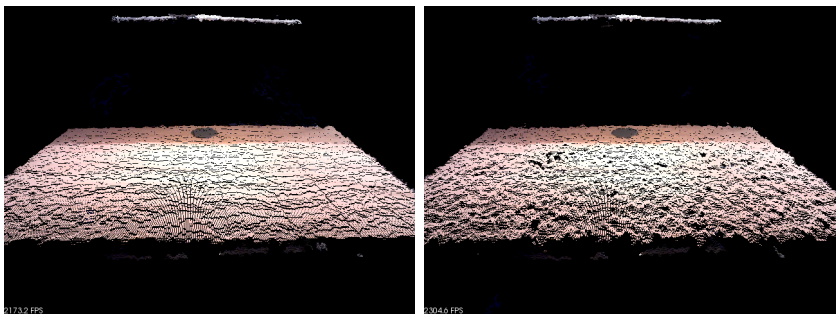


Figure 3.7: Point cloud renders demonstrating the effect of noise when multiple Asus Xtion cameras observe a scene. More noise is observable in the right image, where three cameras observe the scene, compared to in the left image, where a single camera observes the scene.

host machine. However, the data from each of the cameras was not guaranteed to be synchronised and so the Algorithm 3.1 outlined in Section 3.2.1, is used to synchronise frames. In Algorithm 3.1, the threshold parameter, τ, was set to be 16.7 ms, which is half of the average frame time at 30 frames per second. A multi-threaded approach is used to ensure that all camera acquisition threads are begun simultaneously.

To capture task performances, data from each camera must be recorded simultaneously. This recording is challenging for limited hardware due to the large amount of data produced. A multi-threaded program was developed using C++ to perform this recording. A producer-consumer design was used in conjunction with a circular buffer data structure for each camera to maximise the benefits of the multi-threaded approach. This design allows simultaneous recording to be performed in real-time using a single PC (Intel i7 4770K, 24GB of RAM and a solid state disk for storage). This implementation allows for easier reproduction in line with the factors of Section 3.1.1. By using a single computer solution, potential synchronisation complications of a networked solution are also avoided.

Due to the nature of the structured-light depth cameras, interference is an undesirable side effect of using multiple sensors to observe a scene. If multiple cameras' patterns fall on the same surface, a single camera is unable to disambiguate its own projected infra-red pattern, resulting in holes and noise in the depth maps, as shown in Figure 3.7. The "Shake 'N' Sense" technique [320] is used to address this problem. This technique involves vibration of affected sensors which causes the patterns of other cameras to appear blurred relative to its own pattern. In our implementation, vibration motors are attached to the cameras, as shown in Figure 3.6, and the vibrational frequency is tuned to minimise the presence of noise and holes in the data. The advantages of this technique are that it maximises the amount of data available and has zero computational cost.

3.3.2 Point Cloud Processing

A code framework was developed to allow for efficient processing of point cloud data. This framework is architected using object-oriented design and efficiently implemented

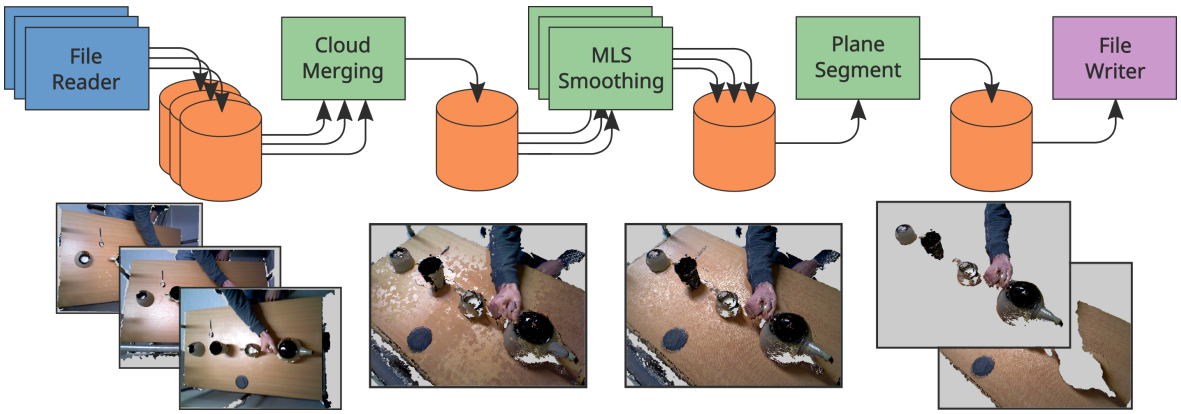


Figure 3.8: An example pipeline implemented using the devised software framework. The pipeline is responsible for reading three sets of camera recordings from disk, merging, smoothing, segmenting and saving the outputs to disk.

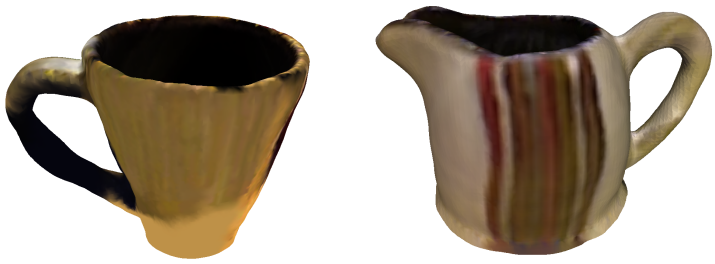


Figure 3.9: A rendering of scanned objects as meshes.

in C++. Properties of this framework include: it allows different techniques to be specified easily and succinctly; is adaptable to different input types; it makes use of multi-threading to maintain efficient performance; and that it is extensible to new use cases. Further details about this framework are available in Appendix C. This framework is used to instantiate the point cloud cloud processing pipeline design of Section 3.2.2, as illustrated in Figure 3.8.

3.3.3 Pose Estimation

To estimate object poses using the algorithm, we scan the objects of interest using 3D scanning software, examples of which are shown in Figure 3.9. The point clouds generated by these 3D scans are used to generate the object templates for the Linemod method, as well as for the GICP incremental registration method of tracking object pose as detailed in Section 3.2.3. A pipeline for generating object and arms poses for an entire sequence is composed using the point cloud processing framework (Section 3.3.2).

3.3.4 Action Classification

Dataset Collection

To evaluate our system, a dataset was collected of RGB-D videos of people performing the task of preparing a cup of tea, called the "Cup of Tea" dataset. The dataset is composed of 24 samples recorded using three Asus Xtion Pro Live RGB-D cameras. A total of eight subjects were recorded performing the task three times. There were no restrictions imposed on how they prepared the cup of tea, other than the order. They were first asked to prepare it in the order of their preference. They were then asked to read and remember the order and attempt to complete it in that order. Finally, they were asked to perform this task again. This method was chosen to ensure that a range of orders was captured so that an action recognition system cannot simply memorise the order of actions. The videos were all manually labelled with one of the five actions for each frame: 'pour tea'; 'pour milk'; 'add sugar'; 'stir'; and 'background'. The rules for when these labels apply are listed in Appendix D. A total of 25,913 frames were recorded, equating to an average video length of 38 seconds.

Network Training

Each network is trained to recognise actions of the Cup of Tea dataset. Given a sequence of pose features for a time t, $(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t)$, where $n \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^d$, and d is the length of the pose feature vector, we wish to estimate the output probability mass function, $\hat{p}_Y(y)$, where $y \in \mathcal{C}$, the set of action labels. Each network we train is a nonlinear differentiable function of the inputs, parametrised by the weights, $\hat{p}_Y(y) = \mathcal{F}(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t; \{\mathbf{W}_i\})$, where $\{\mathbf{W}_i\}$ is the set of weights of the network.

Given the true probability mass function, $p_Y(y)$, for the network inputs $(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t)$, we seek to establish a loss function to minimise during network training. A commonly used measure between an estimated and a fixed reference distribution is the Kullback-Liebler divergence [23],

$$D_{KL}\left(p_Y(y), \hat{p}_Y(y)\right) = \sum_{c \in C} p_Y(c) \log\left(\frac{p_Y(y)}{\hat{p}_Y(c)}\right). \tag{3.9}$$

It is equivalent to minimise the categorical cross entropy, up to a constant value for the entropy of $p_Y(y)$, which is the case when this distribution is fixed,

$$H(p_Y(y), \hat{p}_Y(y)) = -\sum_{c \in C} p_Y(c) \log\left(\hat{p}_Y(c)\right). \tag{3.10}$$

Due to the balance of labels in the Cup of Tea dataset, using the above loss function, training proceeds in a way that, in aggregate, penalises more for the more represented labels in the dataset. This can lead to the network dedicating more of its learning potential to these specific labels and less to the under-represented label classes. To counteract these effects, weights can be applied to the label classes in the loss function,

$$H_w(p_Y(y), \hat{p}_Y(y)) = -\sum_{c \in C} w(c) p_Y(c) \log\left(\hat{p}_Y(c)\right), \tag{3.11}$$

where $w(c)$ is the weighting for the class c. Class weights are chosen to be inversely proportional to the fraction of these class labels in the training set, subject to $\sum_{c \in C} w(c) = 1$.

We minimise a loss function based on the categorical cross entropy, H_w, for each minibatch (i.e. small training subset B where $|B| = N$), formulated as

$$J(\{\mathbf{W}_i\}) = \frac{1}{N} \sum_{j \in B} H_w\left(p_Y^j(y), \mathcal{F}(\mathbf{x}_{t_j - n}, \ldots, \mathbf{x}_{t_j}; \{\mathbf{W}_i\})\right) + \sum_i \lambda_i ||\mathbf{W}_i||, \tag{3.12}$$

where an $\mathcal{L}_2$ regularisation loss is imposed on the set of weights scaled by a set of parameters $\{\lambda_i\}$. To train the network, back propagation [32] is used, In this method, each weight, W_i, is updated based on a function of the gradient of the minibatch loss with respect to the weight. The Adam optimisation method [321] is used for this update function with a learning rate of 0.001. This method precludes the need for manual learning rate scheduling, via adapting the learning rate based on estimates of lower-order moments at the gradients.

Each network is trained for 250 epochs of the training set, with a minibatch size of 1024. The weights of the dense layers and LSTMs in the network were initialised with Xavier uniform initialisation [40], and the offset biases were initialised with zeros. All networks are implemented using the Tensorflow deep learning framework [322]. We utilise a leave-one-subject-out cross-validation, testing on an individual subject for each fold (effectively 8-fold cross validation). This cross-validation has the benefit of characterising the performance of the system for an unseen subject, identifying cases which the system may find challenging to classify correctly.

Performance Metrics

Metrics representing desirable properties of the classifiers are selected to evaluate the classification techniques. Some commonly used evaluation metrics for multi-label classification problems are precision and recall. Precision is the ratio of true positives to the total number of predicted positives, and thus provides a measurement of how many of a classifier's predicted true samples are correct. Recall is the ratio of true positives to the total number of positives. It provides a measurement of how well the classifier performs at identifying all of the positive samples. Each of these is desirable properties for classifiers, and depending on the domain, may be more or less important. A measurement that combines precision and recall into a single value is the F_β score. This is the weighted harmonic mean of the precision and recall,

$$\mathrm{F}_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}. \tag{3.13}$$

In our case, we select the value of β to be 1 so that equal weight is given to the properties of precision and recall, denoting the measurement by F_1.

In our case of multi-label classification, these metrics are not uniquely defined. In the multi-class situation, we consider each label separately, in a one-versus-all manner. Thus, a true positive is a predicted label class that matches the ground truth label class for the class in question. False positives, true negatives and false negatives can be identified similarly for each class in this one-versus-all manner. Thus, we can calculate the identified metrics for each class. To produce a single metric for the multi-class case, we take the unweighted mean across the metrics of the individual classes. To calculate the metrics over the eight folds, the predicted and ground truth labels for each of the test splits are combined, and the evaluation metrics are calculated for each label and combined by averaging, as above. This follows methodology used in similar works, such as 50 Salads [232], following recommendations based on analysis of cross-validation evaluation [323].

Hyperparameter Selection

The selection of algorithm parameters can be time-consuming and involves expert knowledge that is difficult to convey. Due to elongated training times of neural networks, techniques such as exhaustive searching over grids of possible values and random search can be infeasible. However, there exist methods of searching that can reduce the number of search iterations required. One approach utilises Tree-Structured Parzen Estimators (TPE) to model the target function, whereby each of the sampled points is represented with a Gaussian distribution in the hyperparameter space [324].

To identify a point in hyperparameter space to sample next, x^*, a commonly used test is Expected Improvement [325]. This can be defined as the expectation, under some model M of a fitness function $f : \mathcal{X} \to \mathbb{R}$, that $f(x)$ will negatively exceed some threshold y^*,

$$\mathbf{EI}_{y^*}(x) := \int_{-\infty}^{\infty} \max(y^* - y, 0) p_M(y|x) dy. \tag{3.14}$$

where $\mathcal{X}$ is the space of the hyperparameters. Further details of how this expectation is used to select hyperparameters can be found in Appendix E.

In our case, as well as model parameters, we wish to select architectural parameters, such as number of LSTM layers, and whether to additionally train on reversed sequences (i.e. bidirectionally). Other decisions include input data sampling decisions, such as the sequence length and temporal stride used when generating sequences (measured in frames), and regularisation parameters. All hyperparameters selected are shown in Table F.1 in Appendix F as well as the prior distributions for each. In each

search step, we seek to maximise the mean F_1 score across all of the cross validation test splits, effectively performing a grid search on these test splits. Fifty search iterations were performed, with each iteration tested using this cross-validation scheme. The HyperOpt library [324] was utilised to perform this hyperparameter search.

Performance Benchmarking

Benchmark recognition algorithms are selected to evaluate the performance of our LSTM action recognition approach. These algorithms are the ensemble models, Random Forest [326] and Gradient Boosted Decision Trees [327]. These models are selected due to their performance on high-dimensional recognition tasks [8, 9] and their use in related works [236].

The Random Forest algorithm is an ensembling technique whereby decision tree classifiers vote on a prediction, with individual trees trained on subsets of the dataset. The selection of the features for each classifier is performed by drawing independent and identically distributed samples from a distribution with the same dimensions as the feature space. The split threshold used at each node is determined by the value of a metric, such as the information gain, at the data points in a certain feature dimension. The randomness imbued in the technique make it robust in terms of noise [326] and thus a good candidate classification technique.

Gradient Boosting is another ensembling technique, commonly applied to decision tree classifiers. Boosting algorithms [328] train additional classifiers to fit weighted versions of the training data. This weighting decorrelates the classifiers by focussing on errors of the previous classifiers [329]. Gradient Boosting [327] generalises previous boosting algorithms [328] to work for any differentiable loss function. The additional classifiers are trained to approximate the gradient in the loss function with respect to the current model. This approach has the advantage of directly optimising the cost functions; however, it is prone to overfitting when noise affects the gradient.

Hyperparameters of these classifiers are also searched over using the previously outlined technique. These hyperparameters were: the sequence stride; the sequence length; the number of tree estimators to use; and the maximum depth of the individual trees. As in the case for the LSTM classifier, we maximise the mean F_1 score, calculated across all of the cross-validation test splits (i.e. grid search over test splits), and perform fifty search iterations. For Random Forests, the criterion used for selecting splits during training was the Gini impurity [326]. For Gradient Boosted Decision Trees, the deviance was selected as the loss function, and a learning rate (which affects the contribution of additional trees) of 0.1 was used. Data normalisation is utilised only for the Gradient Boosting classifier, as it was observed to affect results for the other classifiers adversely. The hyperparameters searched over are shown in Table F.2 in Appendix F as well as the prior distributions for each.

Without synchronisation			With synchronisation		
Recording number	Mean	Std. dev.	Recording number	Mean	Std. dev.
1	6.07	1.05	1	2.08	0.98
2	8.74	2.79	2	1.55	1.11
3	5.11	0.99	3	1.43	0.70
4	8.04	2.30	4	1.12	0.76
5	5.40	0.94	5	0.86	0.80
Average	6.67	1.63	Average	1.41	0.46

Table 3.1: Mean frame spread values for recordings of 5,000 frames, without and with camera activation synchronisation. The average values are calculated over five recordings. All values are reported in milliseconds.

3.4 Evaluation

3.4.1 Data Acquisition

The techniques for synchronisation, detailed in Section 3.2.1, were used to ensure temporal consistency of the data from the cameras so that the output of later merging stages would exhibit fewer artefacts. An algorithm for matching data received from the cameras, and a method of activating the cameras to minimise the variance of the timings of data, were devised. An inspection of the timings is performed to evaluate these techniques.

Qualitatively, we examined multiple tests of the timings for data received from three sensors for an initial 300 milliseconds and observed that the timings of the data received from each of the cameras are more closely aligned, potentially ensuring better temporal consistency, when the synchronous camera activation method was used.

An experiment was performed whereby multiple recordings were taken under the unsynchronised and synchronised camera activation methods. The frames from each of the cameras are then associated in line with Algorithm 3.1. The standard deviation of the timings is then calculated for each set of associated frames, which we shall refer to as *frame spread*. These statistics are calculated for five recordings of 5,000 associated data frames, and the mean and standard deviation of frame spread for all the data frames in each recording is calculated.

Without synchronised camera activation, a mean frame spread of 6.67 ± 1.63 milliseconds, averaging over the five recordings, is observed. However, when the synchronisation was used, this number was reduced significantly to 1.41 ± 0.46 milliseconds, a 79% reduction in frame spread. Thus, this technique better ensures temporal consistency of data recorded using the framework. The frame spread results are listed in Table 3.1.

The Shake 'N' Sense [320] method was used to improve the quality of the depth information when multiple cameras observed the same scene. To validate our implementation, we perform experiments that assess the effects of the technique upon data

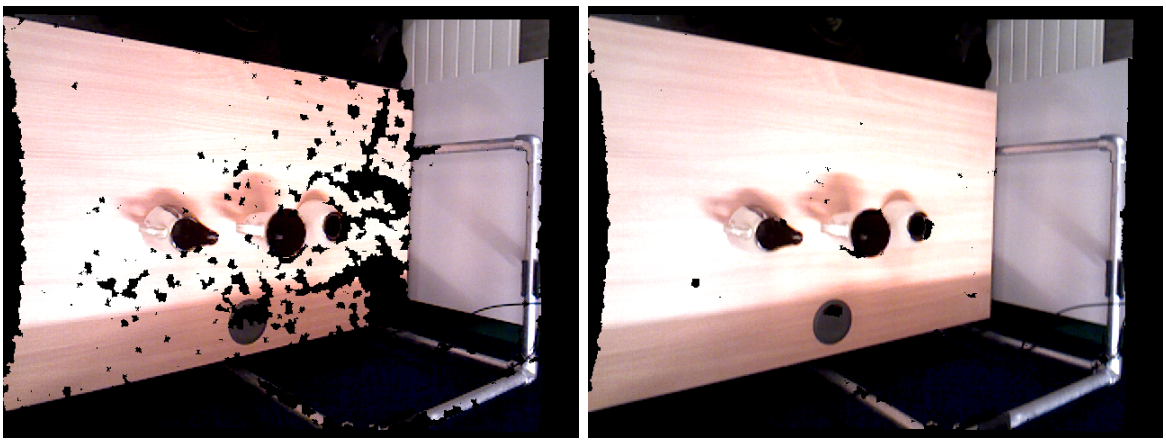


Figure 3.10: A qualitative result showing the difference in data quality observed when the Shake 'N' Sense implementation is used (right) compared to when it is not used (left).

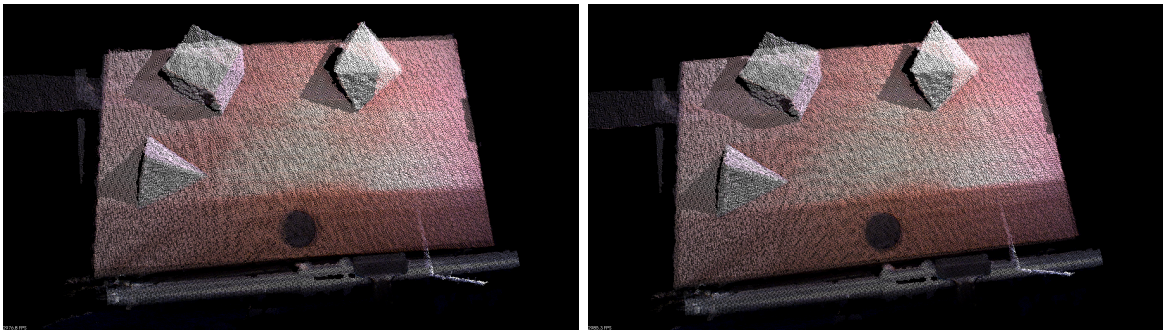


Figure 3.11: A point cloud visualisation of registration results before (left) and after (right) the calibration refinement stage. The gray circle in lower-centre the images, represents a feature of the desk, and is observed to be better registered in the right image (where it appears less blurry).

loss and noise. We find significant decreases for both of these aspects when the camera shaking methodology is used. Thus, in our particular case of three depth cameras observing a small close scene, the camera shaking methodology is suitable for increasing the amount of valid data available for later processing, as demonstrated in Figure 3.10. Full results and analysis are available in Appendix G.

In Section 3.2.1, a custom two-stage technique for the calibration of multiple depth cameras was detailed. To evaluate this technique, we qualitatively examined registration results before and after the second stage of refinement. An example of such results is shown in Figure 3.11. It is observed that the refinement stage results in better-aligned point clouds, with less observable alignment artefacts, such as along the edges of the tables, and around the surfaces of the calibration target objects, in registered point clouds.

3.4.2 Point Cloud Processing

We qualitatively assess the results of the techniques used to process and segment merged point cloud data.

Finding the largest plane in a merged point cloud was sufficient to capture the

points relating to the table. The plane was made more reliable by using the moving least squares smoothing technique. The Euclidean clustering method, used to cluster points, involved tuning a threshold distance parameter. It was found that a higher threshold resulted in parts of surrounding structure being included as part of this table cluster. However, the resulting included points are easily discarded at later stages of the pipeline (such as arm pose estimation) due to their small size under clustering.

The convex hull estimation and prism extrusion is found to be an effective method of segmenting the points that lie above the table region. The minimum height value above the table from which the prism is extruded can reliably be set so that it minimises the inclusion of points from the table while maximising the inclusion of points above the table.

3.4.3 Pose Estimation

Object Pose Estimation

The ground truth poses of the objects in use in our experiments is not available. Thus, to evaluate the object pose estimation technique, a qualitative analysis is performed. We determine correctness based on correspondence to perceived object poses.

Overall, the object pose estimation performs reliably under this analysis. Of the 24 recordings, with four objects tracked in each recording, there are two instances of an object's pose tracking being irrecoverably lost. There are four further instances where the alignment of an object's pose is perceptively incorrect for a portion of the recording. Another perceptible issue is slight misalignments of object attributes, such as handles and spouts.

In the two cases where pose tracking is lost, the object (the cup) briefly becomes fully occluded during the performances by a combination of the subject's arms and other objects. As a result, the GICP algorithm converges to the nearest present points, and thus the object tracking is unable to converge to the correct object points once the object reappears. To overcome this issue, a detection step could be used to ensure that the object is visible prior to the GICP stage.

The four cases of incorrect pose alignment also relate to the cup, however, the cause is different. During the task performance, the cup is filled with tea which changes the perceived topology of the object. As a result, the GICP algorithm attempts to register object points no longer available in the point cloud. Other nearby points may increase registration instability as the algorithm may incorrectly identify these as correspondences. This instability is challenging to overcome while maintaining generality. An image classification stage, to determine whether the cup has become filled, could be used to choose different object models to register (e.g. empty cup, full cup).

Some estimated object poses exhibit slight misalignments around object attributes such as handles and spouts. The objects in use, in this case, had smooth surfaces,

Figure 3.12: Pose estimation results for different sample recordings. In the first three rows, hand poses and object poses are estimated reliably. In the fourth row, object pose estimation errors are shown. These are due to misalignments of the cup due to changing topology and tracking losses due to complete occlusions. In the final row, arm pose estimation errors are shown. These errors are due to incorrect segmentation of left and right arm clusters when the hands are close together, and incorrect estimation based on inclusion of points from the subject's body.

which are large relative to the object's total size, and are also highly symmetrical. These object attributes affect the convergence of the GICP algorithm, with correspondences from symmetrical surfaces leading to early convergence, without establishing correspondences for protruding object attributes (e.g. spouts and handles). GICP algorithm parameters can be tuned to identify correspondences at greater distances to overcome this issue. This tuning, however, leads to increased incorrect correspondences with other points, such as those from the subject's arms. Results of the object pose estimation technique are shown in Figure 3.12.

Arm Pose Estimation

Similar to object pose estimation, ground truth data for arm pose estimation is unavailable. We adopt a qualitative assessment methodology, with correctness determined based correspondence to the perceived pose.

Overall, arm poses are estimated reliably for the majority of the performance videos. As the technique relies on a single merged point cloud, errors do not accumulate over time. Observed errors include confusion between arms and inclusion of object parts or other body parts in the pose estimation. It was found that six of the performance videos were affected by such issues to varying extents.

Arm pose estimation errors occur due to violations of the assumptions of the technique. It assumes that the input point cloud is segmented to include only points relating to the arms (with some tolerance to noise via clustering). In cases where object pose is irrecoverably lost, points relating to the untracked object affect the arm pose estimation stage. Subjects leaning over the task table also lead to the inclusion of points that do not relate to their arms. A smaller error occurs when a subject brings their hands together briefly causing the cluster segmentation to identify a single arm cluster. Finally, a brief error occurs when a subject reaches at an acute angle to reach an object on the opposite side of the table, causing the technique to identify the handedness of the arm incorrectly.

Due to the assumptions of this technique, it inherits errors from object pose estimation and segmentation steps. Should object pose estimation performance improve, this technique would exhibit fewer errors. However, to improve robustness to these errors, a method such as RANSAC could be employed to fit an arm model in the presence of outliers. This would involve the regression of a range of candidate arm poses and shapes, and hence would require a method of representing this wide range of poses and shapes. Techniques dedicated to the estimation of arm poses [8] and hand poses [9] could be employed. However, of these techniques, there are fewer that deal with the more challenging problem of pose estimation during human-object interactions [330, 219]. These techniques that apply data-driven methods [219] have been trained for specific objects which may not directly translate to our specific use case. Results of the arm pose estimation technique are shown in Figure 3.12.

3.4.4 Action Classification

We identified hyperparameter arrangements that maximised the F_1 metric for each of the tested actions classifiers. In the cases of Random Forest and Gradient Boosted Decision Trees, there are regions of better performance in hyperparameter space, in particular, maximum depth of trees has a significant effect upon results. In the hyperparameter values of the optimal LSTM classifier, we note that a small learning rate in conjunction with relatively high dropout values are used indicating that the training required significant regularisation to prevent overfitting. Visualisations of the hyperparameter searches and optimal LSTM classifier can be found in Appendix F.

The optimal LSTM network performs better than the optimal benchmark classifiers by a significant margin, as can be seen in Table 3.2. The accuracy of the classifier is 82.9% calculated over all of the test splits. The F_1 metric, which we optimised against, is 81.72%, which is 8% above the other classifier results. We also observe that the classifier has a smaller standard deviation across the action classes, indicating that the classifier learns to discriminate the actions more evenly. It should be noted however that, as a deep architecture with a larger number of trainable parameters, the LSTM network has greater representational ability than the other benchmark classifiers. It should also be noted that we searched over architectural and regularisation parameters to identify a final LSTM network architecture. This may give an advantage over the benchmark classifiers which were searched over smaller sets of parameters.

Classifier	*Accuracy*	*Precision*	*Recall*	*F1 score*
Random Forest	76.21	82.54 ± 17.13	71.11 ± 20.59	73.55 ± 14.91
Gradient Boosting	75.68	**85.06** ± 18.12	70.45 ± 20.94	73.23 ± 13.89
LSTM	**82.90**	81.46 ± 8.65	**82.20** ± 7.62	**81.72** ± 7.49

Table 3.2: The classification metrics for the tested classifiers.

The results indicate that the deep LSTM network, trained on pose information, can represent the dynamics of the human-object interactions. We analyse the confusion matrices for the classifiers, as shown in Figure 3.13, to characterise performance on individual actions For the Random Forest and Gradient Boosting classifiers, we observe that the classifiers are strongly biased towards predicting the *"Background"* action class. The ground truth action class distribution is heavily dominated by the *"Background"* and *"Pour Tea"* classes which may explain this bias somewhat. The *"Pour Tea"* class is identified well by all classifiers possibly due to the large motion of the teapot. The classes *"Background"*, *"Place Sugar"* and *"Stir Tea"* are more difficult to disambiguate as these do not necessarily involve the movement of a tracked object, and the arm pose dynamics present are subtle. Inspecting the confusion matrix for the LSTM classifier, it confuses the same three action classes to a lesser extent, indicating that it was better at learning a representation of these actions from the input pose information. For the LSTM, the *"Stir Tea"* class is predicted incorrectly as *"Place*

Sugar" in 17% of cases. Given that both of these actions involve similar arm interactions with a cup, it is expected that these two action classes would be the most difficult to differentiate.

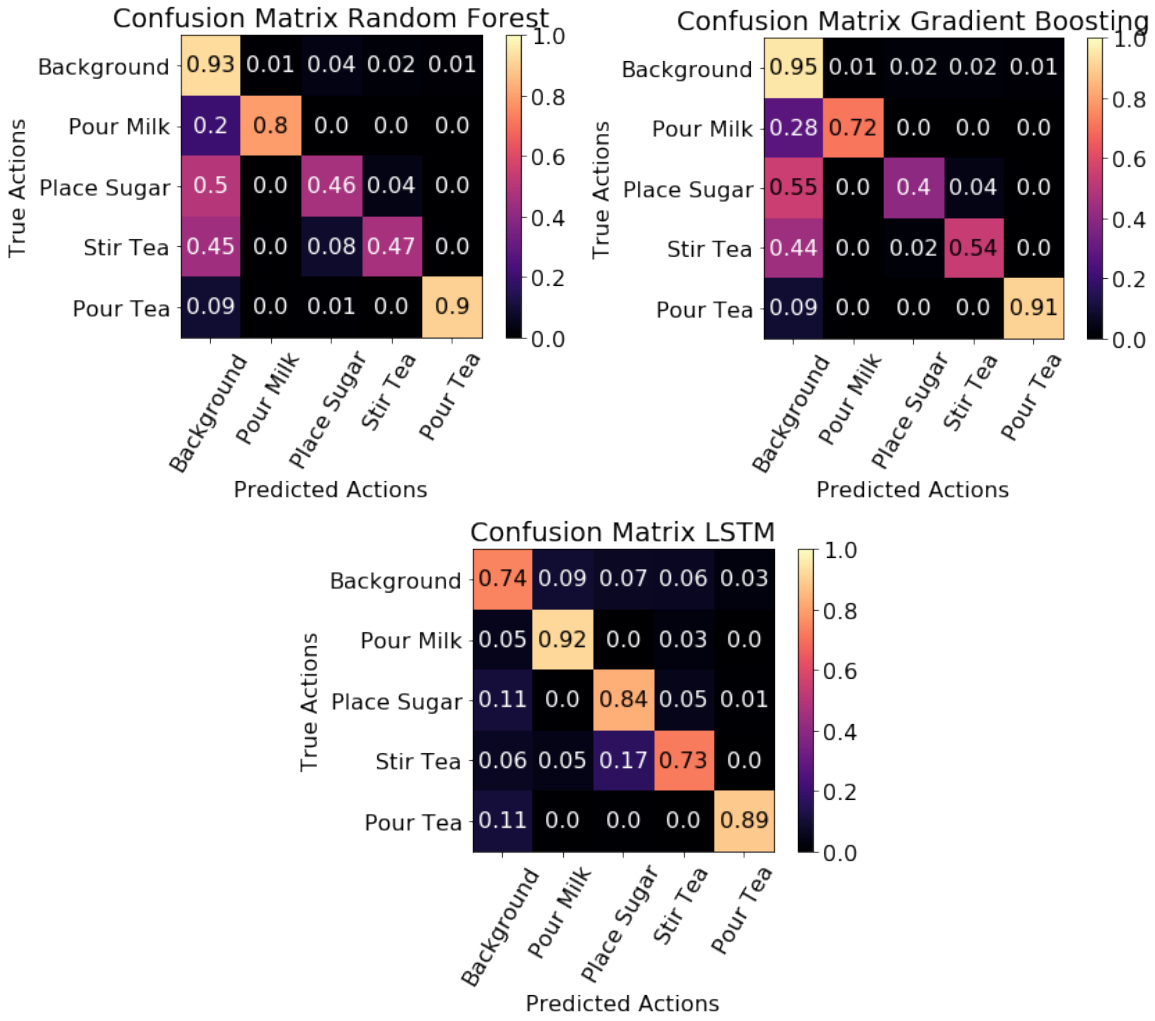


Figure 3.13: Confusion matrices for the three tested classification techniques. The LSTM classifier has more even distribution of correct predictions over all of the classes than the Random Forest or Gradient Boosting classifiers.

Split	*Accuracy*	*Precision*	*Recall*	F_1 *score*
1	87.43	84.85	84.80	83.21
2	92.79	92.78	92.12	92.28
3	70.22	78.02	73.05	71.54
4	70.53	67.95	71.53	65.93
5	90.55	89.22	85.43	86.37
6	87.55	84.22	88.02	85.37
7	93.44	92.41	93.60	92.54
8	67.41	61.88	66.30	59.53

Table 3.3: The classification metrics using the hyperparameters of the best performing LSTM for each of the test splits.

To further understand where the LSTM classifier underperforms, we examine its performance for each of the test splits. We calculate the previously used metrics for

Figure 3.14: The training loss profile for the LSTM classifier. The feint green lines represent individual test splits of the datasets. It is observed that some are overfitting (by not converging) while others may be underfitting (by converging too early).

each test split, with the precision, recall and F_1 score calculated for each action class, and report the unweighted mean in Table 3.3. We note a large difference between the maximum F_1 score (93%) and minimum F_1 score (60%) for the splits. We observe that three specific splits underperform relative to the others, namely splits 3, 4 and 8. These splits contain the most significant errors of object pose estimation and arm pose estimation, as detailed in Sections 3.4.3. As such, improvements to these pose estimation methods should result in better performance for these splits.

To gain insight into the effect of pose estimation errors upon classifier training, we collate the training profiles for each of the splits, plotting training and validation loss Figure 3.14. We observe that the classifier is overfitting for certain splits, evidenced by increasing loss values after fifty epochs. Earlier, we noted from Table F.3 that the best performing hyperparameters included significant amounts of dropout. Thus, this level of regularisation may be present to limit this overfitting. Improving pose estimation may reduce regularisation requirements and hence improve scores for splits that may be underfitting currently. Finally, the variance in splits may also be due to variations in the motion patterns of the subjects when performing actions. The results could improve with additional dataset samples that capture a broader and more representative range of motion patterns.

3.5 Discussion

In this chapter, a system was designed, implemented and evaluated for the recording of table-based human-object interactions from multiple viewpoints. We devised a pipeline for the tracking of 3D poses of arms and objects from point cloud data received from multiple cameras. We showed how recurrent neural networks, such as LSTMs, are capable of recognising constituent actions of such interactions using estimates of 3D

poses of arms and objects. Finally, we collected a multi-camera RGB-D dataset of performances of the task of preparing a cup of tea, and we validate our approach upon this dataset.

The data acquisition stage of the pipeline was designed based on its use case of capturing human-object interactions, such as OSCEs. The design relies on multiple depth cameras to ensure all stages of a performance are observed. The use of multiple RGB-D cameras allows a wide range of potential action recognition techniques to be applied in a pervasive computing application. However, due to this bespoke arrangement, issues needed to be overcome to increase the practicality of the system (e.g. camera calibration) and the usefulness of the data (i.e. Shake 'N' Sense). A code framework was developed to minimise the code needed to create potential software pipelines, and to provide increased performance via a multi-threaded architecture. As such, it is feasible for other researchers to reimplement such a system for different purposes. This system thus contributes a useful test harness for studying human-object interactions and for multi-camera reconstruction of small close scenes.

We have demonstrated the use of an action recognition pipeline involving the tracking of 3D poses of objects and estimation of arm poses of the subjects, that is capable of classifying the actions of the Cup of Tea task at 83% accuracy. A neural network architecture and set of hyperparameters were found using Tree Parzen Estimators, using fifty search iterations. As part of the analysis of the classification results, we identified areas for further improvement in the pipeline and proposed potential methods to overcome these weaknesses. In particular, we note that the object pose estimation method is sensitive to topological changes and entire occlusions of objects. The level of accuracy obtained with this method also validates the proposed data acquisition stage.

The pipeline described in this chapter could be applied to numerous real-world problems that require the understanding of human-object interactions. A potential application of such a pipeline is smart assembly line monitoring, where human interactions with a fixed set of objects can be understood to examine if they have completed all necessary actions.

The overarching goal of this thesis is to devise action recognition techniques that could be used as part of pervasive computing applications for human-object interactions. In this chapter, we have devised and evaluated an action recognition technique for human-object interactions involving rigid objects. However, certain human-object interactions may involve deformable objects that may change appearance during an interaction. Given that this was an issue identified in the evaluation of this pipeline, our later work will address these issues to increase robustness in application domains.

Chapter 4

Fusion of Colour, Depth and Scene Flow for Fine-Grained Action Recognition

In this chapter we explore techniques of using convolutional neural networks to recognise fine-grained actions through combinations of different image types afforded by depth cameras, including depth and scene flow. In the previous chapter, we analysed techniques of recognising fine-grained actions via the tracking of objects and poses, finding that recognition is highly dependent upon performance of the pose estimation techniques, which are prone to inaccuracies in scenarios such as occlusions. A benefit of a convolutional neural network approach, over the previous pose-based approach, is that it can learn directly from the image data to recognise fine-grained actions. The convolutional neural network is trained on images from a single centrally-located overhead camera. This chapter also addresses a problem of convolutional neural network approaches to action recognition, whereby it is necessary to train parallel networks for each image type, such as colour and optical flow. The devised approach performs an early fusion of image types using separable convolution filters, based on the observation that these filters separate the learning of channel-wise features and inter-feature relationships. This fusion approach outperforms late fusion of single image type networks showing that this early fusion is an effective technique for increasing recognition accuracy of fine-grained actions without the need for training multiple parallel networks.

4.1 Introduction

4.1.1 Motivation

Recognition of the fine-grained actions involved in human-object interactions, such as OSCEs, is the key research problem tackled in this thesis. Accurately identifying the appearance and motion patterns of these actions, and learning to discriminate between

these can be challenging when these motion patterns and appearances may be similar across actions. Difficulty also lies in the fact that there are a variety of valid approaches to completing tasks. Another complicating issue is that, as a result of these actions, the objects being manipulated may change shape and form. As we observed in the previous chapter, the change in topology of a cup when it became full adversely affected the ability to track the pose of the object throughout a task performance. These reasons make recognition of fine-grained actions challenging using techniques based on semantic tracking of hands and objects.

Convolutional neural networks (CNNs) have been applied to the problem of action recognition [127, 16]. Such approaches have advantages over semantic pose-based recognition approaches that may be relevant to our problem. One advantage of such CNN-based approaches is that the CNN operates directly on the image data, learning representations that can be used to discriminate between actions. As observed in the previous chapter (Section 3.4.4), errors in the distinct pose estimation stage negatively affected action recognition results. As an entire CNN is trained in an end-to-end fashion, without distinct pose estimation stages, it is directly optimised to recognise actions from images. Another advantage of a CNN-based approach is that it is better suited to situations where objects deform and transform, which occurs during human-object interactions such as OSCEs. The previous pose-based approach focussed on tracking rigid objects, and thus may not be applicable when objects change shape and form significantly. The CNN-based approach, however, can learn to recognise the varying appearances of these objects without further intervention needed. Although not a direct research goal, a further advantage of CNN-based approaches is that it enables real-time performance with the deployment of reasonable graphics hardware. In a CNN-based approach, there is no longer a requirement for the processing stages such as point cloud transformations, table removal, and iterative closest point for object tracking. Each of these stages has significant processing associated with it, making real-time processing challenging. Based on these advantages, investigation into the use of CNNs for the problem of recognition of the fine-grained actions involved in human-object interactions is warranted.

Recent methods of applying CNNs to the problem of action recognition often utilise motion information in the form of optical flow [127, 16] as an input to the neural network. This optical flow data can contain highly discriminative details for recognition of certain actions, specifically actions involving distinctive motion patterns such as sporting actions [273]. The importance of high-quality optical flow information to learning accurate action models has been shown by Varol et al. [135]. In the case of fine-grained action recognition, this motion information may be of even greater importance. In particular, it has been shown that flow information has a higher learning utility when videos are more uniform [128], which is the case for the fixed-setting activities we focus on. For these reasons, we wish to utilise flow information that captures fine-

grained motion patterns in as much detail as possible. As such, we propose the use of scene flow information, the three-dimensional analogue of two-dimensional optical flow information, for recognition of fine-grained actions using a CNN architecture. We leverage a recent method that utilises the primal-dual algorithm to calculate scene flow information from RGB-D data at real-time rates [18]. At the time of research, we had yet to observe techniques that utilised scene flow information for the task of fine-grained action recognition as part of a deep learning framework.

In this chapter, we examine the applicability of learning jointly from colour, depth and scene flow information to perform fine-grained action recognition. We propose an early fusion approach of performing this joint learning, which has certain benefits. These benefits are the removal of the need to train multiple parallel network branches, the co-locality of motion and appearance information, and the possible discovery of joint latent data structures across depth and colour, Many recent CNN techniques for action recognition [127, 16, 135] train parallel CNN branches for the colour and flow information, with outputs fused through weighting [127, 89], discriminative classifiers [39, 128], or through further neural network layers [137]. It has been reported that fusion at latter convolutional layers improves classification results, however, this approach requires expensive training of parallel branches. In our case, we have multiple image types available, including colour, depth and scene flow, making training on all image types prohibitively expensive. Another benefit of early fusion is that jointly located information in different image types may aid the learning process. For example, in the activity of preparing a meal, the actions of chopping tomatoes and chopping cucumbers may exhibit similar motion patterns. The co-locality of appearance information with this data may allow earlier disambiguation between these two similar actions. A further benefit of early fusion relates to colour and depth image modalities. Work that involved recognition across these modalities [13] observed that there exist latent structures, and by learning how to fuse these modalities a joint representation can give better classification results. For these reasons, we explore a technique for performing early fusion of image types via depth-wise separable convolution as part of a CNN architecture. In our evaluation, our early fusion technique outperforms a late fusion of outputs of networks trained on single images types.

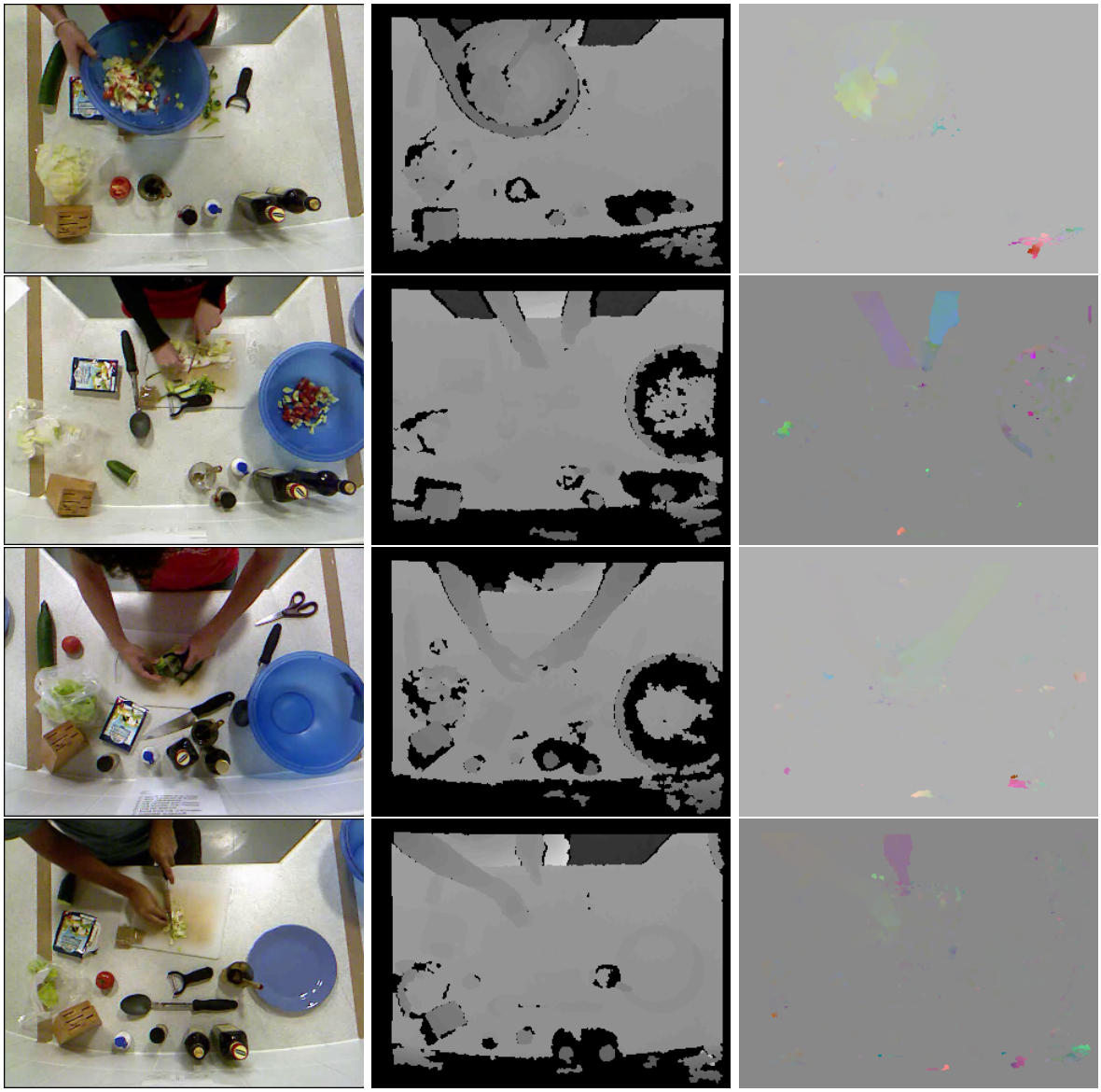
4.1.2 Contributions

The contributions made in this chapter are:

- The proposal of scene flow information for fine-grained action recognition;
- The proposal of a method to allow action recognition from multiple image types without expensive parallel branches; and
- An evaluation of such an approach in combination with recurrent neural networks for fine-grained action recognition.

4.2 Design

In this section, we describe each of the components of the fine-grained action recognition neural network architecture. This architecture takes as an input motion information in the form of scene flow. Scene flow is the 3D vector field that describes the motion of an image scene over a small time interval. We use the PD-Flow algorithm [18] to calculate this flow information from RGB-D images. A visualisation of sample outputs of this algorithm are shown in 4.1.



(a) Colour **(b)** Depth **(c)** Scene Flow

Figure 4.1: Visualisations of the different image types used to recognise actions. Each row is extracted from a single RGB-D frame from the 50 Salads dataset [232]. An unwanted artefact observed is the presence of noise in the scene flow images (c). This is due to pixels appearing and disappearing between successive depth images, causing the PD-Flow algorithm to estimate flow incorrectly for these pixels. The depth and scene flow images are scaled for visualisation purposes.

4.2.1 Deep Learning Architecture

The network architecture is composed of an early fusion component and a base convolutional network. The network takes as input a single colour frame, and its corresponding depth and scene flow frame, combined by concatenation in the channel dimension.

Data Fusion Given the availability of different image types from RGB-D information, including colour, depth and the generated scene flow, the question remains of how to combine this information in a deep network. Previous methods of fusion have included weighting the classification scores of independent networks that have been trained separately [127, 16]. This approach prevents the discovery of important features jointly located across the image types.

An alternative approach, that would allow discovery of such latent patterns, may involve forming the input image tensor by concatenation of each of the image types and training a single network. In convolutional layers [35], the network is tasked with learning convolutional features that operate across all channels of the input image tensor. Thus the feature kernels will need to jointly recognise channel-wise features and inter-channel relationships. This joint learning task is more challenging when the number of input channels is greater, as is the case when multiple image types are used.

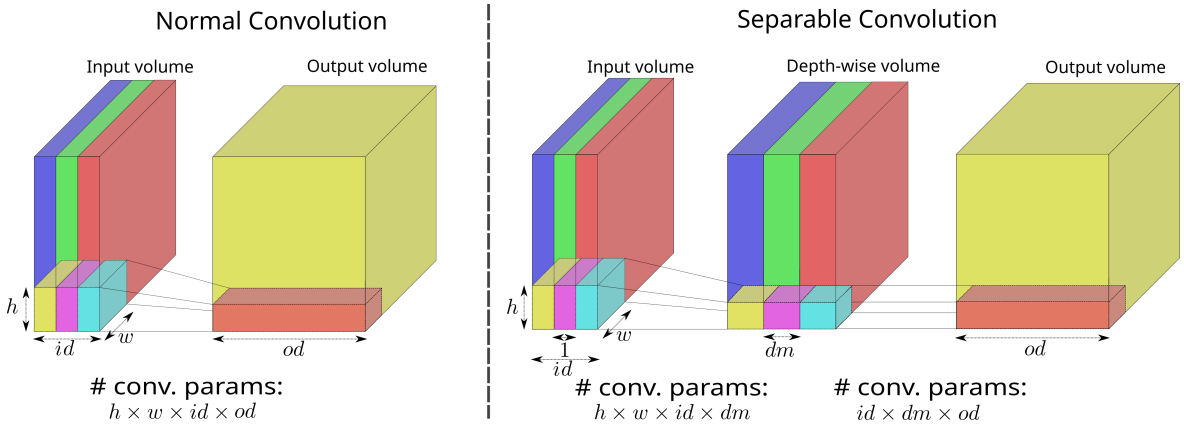


Figure 4.2: Separable convolution consists of a depth-wise convolution stage followed by a point-wise (1×1) convolution. There are two sets of parameters for separable convolution, the first set is tasked with learning spatial features in each input channel, and the second set is tasked with learning a linear relationship between these depth-wise feature responses. In the figure, the number of input channels, id, is 3, for illustrative purposes. However, separable convolution works for an arbitrary number of input channels, id.

This convolutional learning task can, however, be decomposed into a pair of convolutional tasks by utilising a technique known as separable convolution [331]. A separable convolution works by first performing a distinct convolution operation for each channel of the input image tensor, known as depth-wise convolution. The outputs of this convolutional stage are combined by a convolution operation with a receptive field of a single pixel, known as point-wise convolution. This operation effectively learns a

linear relationship between the individual pixel outputs of the depth-wise convolution operation. The separation of convolution operations permits non-linear activation to be performed prior to point-wise convolution. A consequence of separable convolution is that the task of learning features per image channel is separated from the task of learning relationships across the image channels. A further benefit of separable convolution is that the total number of parameters to be trained can be less than that of normal convolution, depending on the number of depth-wise features, controlled by the depth-wise multiplier parameter, *dm* in Figure 4.2.

In our case, separable convolution is applied at the start of the network architecture. To achieve this, a single colour frame and its corresponding depth and scene flow frames are combined by concatenation along the channel dimension producing a tensor with input channel dimension (*id* in Figure 4.2) of seven. These seven channels are composed of red, green and blue for colour, a depth value, and the three components of the motion vector in the camera reference frame for the scene flow data. As part of separable convolution, the depth-wise convolution operates on each channel separately, including the colour and scene flow channels. Two successive stages of separable convolution are used in our network to learn a fused representation of the combined image types.

Convolutional Neural Network The fusion technique, as previously discussed, can be deployed as the first stage of an architecture to learn a joint representation of the different image types. Recent neural network architectures employ many further layers to build richer hierarchies of images features for visual understanding tasks [39, 73]. In our case, we wish to train a convolutional neural network to recognise image features salient to the task of fine-grained action recognition. Thus, we need to specify the remaining architecture of the convolutional neural network used for this task.

After initial tests on different state-of-the-art architectures [73, 67], it was decided to use a variation of residual networks, known as the Wide Residual Network [76] (WRN) to perform extraction of visual features from the joint image representation. This architecture was devised based on the observation, as discussed in Section 2.1.3, that the increased depth that residual networks permitted resulted in diminishing returns with increasing depth. Instead, the authors favour increasing the number of filters in each layer (called the width) over increasing the depth of the network. This network architecture outperformed contemporaneous networks for CIFAR10 and CIFAR100 image recognition tasks. In our tests, it performed better for fine-grained action recognition than other state of the art networks [73, 67]. However, we found that the WRN architecture performed better when it was modified so that more features were dedicated to capturing discriminative information at larger spatial sizes (i.e. early in the network). The WRN architecture permits control of the number of features in the early stage of the network, via the width parameter, and thus is amenable to this approach.

To provide a baseline for single image action recognition, a full architecture is devised. In this architecture, shown in Figure 4.3, the inputs undergo two stages of

separable convolution to learn a joint feature representation. This output then undergoes a convolution stage before being passed as an input to the Wide Residual Network. The final global pooling layer of the network is reduced to a 4×4 pooling as preserving some late spatial information was found to increase classification performance. The output of this pooling layer is fed into a fully-connected layer with ReLU (rectified linear unit) non-linear activation (a 'Dense' layer), and then into a final fully-connected layer with softmax activation (a 'Softmax' layer) to produce a prediction distribution over the possible action classes.

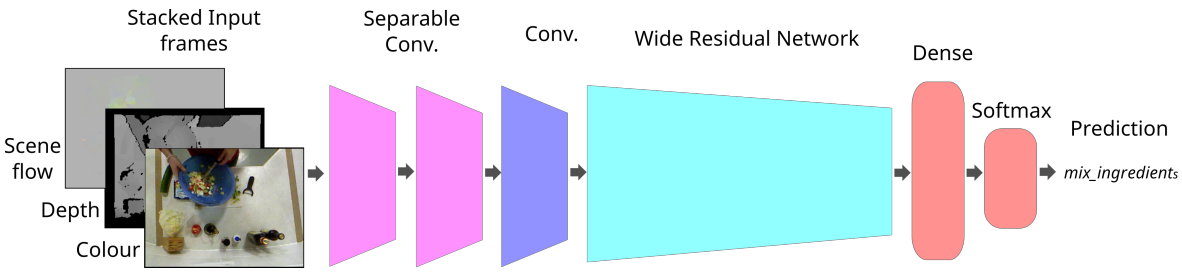


Figure 4.3: The architecture of the network involves a number of convolutional stages. The first two separable convolutional stages are responsible for the fusion of the features. Following this, there is a convolutional stage before the Wide Residual Network of depth 16.

Recurrent Neural Network Until now, our network architecture has not had access to temporal information beyond the temporal information encoded in the scene flow data. An approach utilised in the literature to include temporal information [127] is to combine a CNN with a recurrent neural network (RNN), such as a Long Short Term Memory (LSTM) cell. In such a combined architecture, the CNN functions as an image feature extractor and the recurrent network learns to recognise actions based on sequences of these image features. It is possible to jointly train such an architecture by using an individual CNN for each timestep of the input sequence sample. This leads to a large number of convolutional weights to train, with the number of weights scaling linearly with the number of timesteps considered during training. To overcome this problem, the CNNs can share a single set of convolutional weights, effectively reducing training to that of a single CNN that learns image features present across all timesteps of the input sample sequence.

In our case, we combine the previously described single-image CNN with a recurrent neural network as shown in Figure 4.4. The weights of all of the described convolutional layers are shared across the sequence timesteps. The LSTM cell (previously described in Section 3.2.4) are effective for recognising actions from image sequences [127]. We thus choose to utilise this cell design in our evaluation of the combined architecture.

Another RNN, discussed in Section 2.1.2, is the Gated Recurrent Unit (GRU) [58]. This design, shown in Figure 4.5, combines the input and forget gates, into a single update gate, resulting in a a set of update equations:

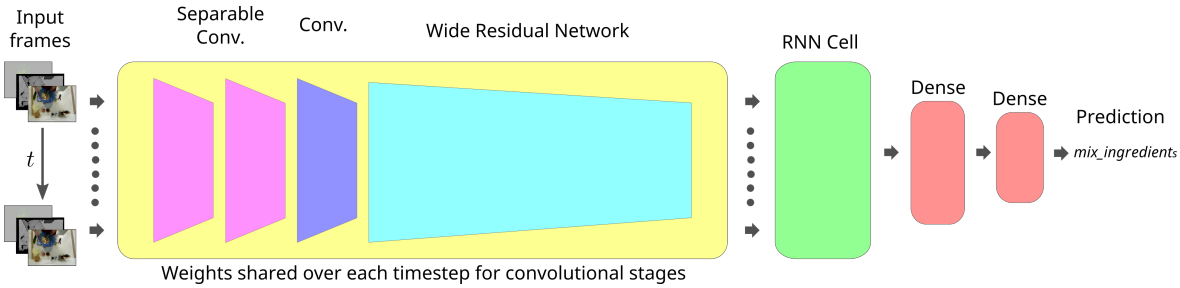


Figure 4.4: This architecture is adapted from the single image version to include multiple images as an input. The convolutional component of the network (yellow) extracts image features, as per the previous architecture in Figure 4.3. The convolutional weights are shared across all timesteps, so that, effectively, a single convolutional component is being trained. The output features of the convolutional component are fed as inputs into a subsequent RNN cell, which learns to discriminate between sequences of these inputs. Finally, two fully-connected (or "dense") layers learn a mapping from this output to a vector of length the number of action classes. Performing a softmax activation on this output produces a class prediction distribution.

$$\mathbf{z}_t = \sigma([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_z}); \tag{4.1}$$

$$\mathbf{r}_t = \sigma([\mathbf{h}_{t-1}, \mathbf{x}_t, 1] \cdot \mathbf{W_t}); \tag{4.2}$$

$$\tilde{\mathbf{h}}_t = \tanh([\mathbf{r}_t \mathbf{h}_{t-1}^T, \mathbf{x}_t, 1] \cdot \mathbf{W}); \tag{4.3}$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t)\mathbf{h}_{t-1}^T + \mathbf{z}_t \tilde{\mathbf{h}}_t^T. \tag{4.4}$$

$$\tag{4.5}$$

This combination of input and forget gates, controlled by a single parameter $\mathbf{z}_t$ results in fewer trainable parameters than an LSTM cell. In comparative studies, the GRU cell performed equal to or better than LSTMs [60, 332]. The LSTM cell, however, outperforms on language modelling tasks [60]. In these problems, there exist long-term dependencies, such as when the sentence subject occurs at the beginning of a long sentence. The better performance of LSTMs here may indicate that the division of input and forget gates may be better suited for such problems. For our sequential information, the clip length will be relatively smaller, and so the case for robustness to long-term dependencies is weaker. As such, we will evaluate the GRU cell as an alternative to the LSTM cell as part of our combined architecture.

Our combined convolutional and recurrent architecture performs late temporal fusion via the recurrent neural network cell. It is possible to use other techniques of temporal fusion at different stages of the network, such as 1D convolution (over time) or 3D convolution (over a space-time volume). However, in this case, we wish to principally study fusion techniques of different image types without the need for multiple parallel networks for each image type. Using our chosen architecture, an ablation study of the performance of different input image types and the fusion of these can inform

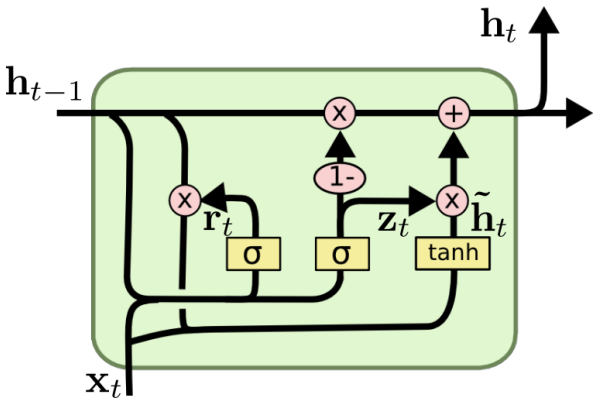


Figure 4.5: The recurrent neural network cell structure for the Gated Recurrent Unit. In this cell, learned weights, operating on the previous hidden state $\mathbf{h}_{t-1}$ and the current data $\mathbf{x}_t$, control combined input and forget gates, via $\mathbf{z}_t$ and $1 - \mathbf{z}_t$, respectively. The input gate controls the addition of an intermediate hidden state $\tilde{\mathbf{h}}_t$. This intermediate hidden state is a learned transformation of the current data and the previous hidden state, where the previous hidden state has undergone a gating controlled by the variable $\mathbf{r}_t$. This variable is itself a learned transformation of the previous hidden state and the current data. Figure is adapted from original, with permission [316].

future design decisions regarding this aspect.

We denote the neural network architecture described thus far as the 'Single Image Network', the 'GRU Temporal Fusion' and the 'LSTM Temporal Fusion', for the CNN, CNN and GRU combination, and the CNN and LSTM combination, respectively.

4.3 Implementation

4.3.1 Dataset

As discussed in Section 2.2.6, there are few existing RGB-D fine-grained action recognition datasets. This research aims to recognise actions involving fine motion and multiple deformable objects. The 50 Salads dataset [232] is an existing dataset that involves similar complex actions in a goal-directed scenario. This dataset consists of 50 RGB-D videos of people preparing a salad in a kitchen environment. It captures 25 different subjects preparing two types of salads using a single overhead RGB-D camera, resulting in over 4 hours of video. The dataset is labelled at multiple levels of granularity. The granularity at which we are attempting to classify (called 'eval' by the authors) is composed of 10 actions, such as *add oil*, *mix dressing*, *add pepper*, *peel cucumber* and *serve salad*. These actions exhibit similar subtle motions and similar appearance, making them challenging to distinguish. The dataset also contains data collected from accelerometers attached to utensils, however for our purposes, we wish to examine the potential for learning from solely RGB-D data.

To perform an evaluation on this dataset, we need to process the data for use by our neural network architecture. We prepare scene flow images for the entire dataset using the PD-Flow algorithm [18] where the flow is calculated in the forward temporal

direction. The dataset of colour and depth images are first downsampled from 640x480 to 320x240, and the scene flow images are calculated at this resolution. The depth images in the 50 Salads dataset are registered to the colour images, and so there exists a pixel-wise correspondence between the image types, including the output scene flow images. The scene flow, depth and colour images are concatenated individually for use as inputs to the network, and stored using the Hierarchical Data Format version 5 (HDF5) with compression to reduce potential disk reading performance bottlenecks when training.

We employ a common preprocessing technique to normalise image data for training [35] such that the data has zero mean and unit variance. For each experiment in our evaluation, we calculate the mean image and variance image over the training set. Due to the large size of the dataset, this calculation is susceptible to underflow and overflow errors. To avoid such issues, the algorithm of Welford [333] is used. This algorithm reformulates the calculation of the first two statistical moments (mean and variance) into an iterative calculation, thus preserving a common precision during operations. Once calculated, the mean image is subtracted from each sample image in both the training and test sets to ensure that the pixel data is centred on the origin. We do not subtract the corresponding mean from the scene flow data. If there is no movement in a scene, the scene flow should ideally already have zero mean. A case where subtraction of the mean may be beneficial is when the camera is moving. In our use case, however, we assume a static camera position. When normalising for unit pixel variance, it was found that division by the variance or standard deviation images led to worse performance than division by a fixed scaling number of the order of magnitude of the data range. This is likely due to the presence of noise in depth and flow images resulting in variances that are shifted by these outlier values. Thus, the samples are divided by a fixed scaling number equal to half of the largest value encountered in the mean image. Division by fixed scaling value has previously been applied when dealing with large datasets, such as ImageNet [69]. As a results of this processing, the pixel data should be centred on the origin and lie mostly in the range $[-1, 1]$.

A common approach to improve generalisation of a CNN is to augment the data in a plausible manner, such as by affine transformations. Performing this augmentation when feeding data to the network produces an effectively larger dataset. In our tests, such augmentation techniques negatively impacted performance. This may be due to the fixed nature of the scene layout in the dataset. Under augmentation, the network dedicates learning capacity to generalisations that do not exist in the test set.

Training against a relatively large image size, such as 320×240, greatly increases the memory requirements for training. An approach of reducing this expense is to perform an early convolution stage with a larger stride, reducing the output image size. Such an approach is employed by Residual Networks [73] when training against ImageNet images of size 224×224. However, this convolutional layer has an associated memory

cost, and under graphics memory constraints, it was found to be more beneficial to downsample and crop the data. The memory saved as a result of this can be used to increase the parameters in the WRN blocks. All image samples are downsampled to 160×120, and a manual cropping, of size 140×70, is performed for the entire dataset. This cropping excludes areas of video frames where there is little visible variation throughout the data.

To compare our technique with the approach described in the Chapter 3, we will evaluate our approach on the Cup of Tea dataset using images from the central camera only. In this case, we use the same approach to prepare the dataset, namely extracting flow information, downsampling and cropping. We train and evaluate against videos recorded using the central camera (of the three used for recording). This camera placement is most similar to the placement of the camera for the 50 Salads dataset, thus allowing us to identify approaches that may be transferable from the smaller dataset to the larger 50 Salads dataset.

4.3.2 Classifier Training

The hyperparameters values used during training were selected after exploratory training runs. The base Wide Residual Network model uses a widening factor of 8. This factor controls the multiple of trainable parameters per convolutional layer in the network. We also use a depth of 16, which controls the number of convolutional layers applied in each block, where a block refers to a contiguous part of the network where the output size is constant. As discussed in Section 4.2.1, it was found to be advantageous to trade-off widening against depth in our case. The base model foregoes further widening or depth in favour of a larger batch size which improved classification results when constrained by graphics memory requirements.

In training, we minimise the categorical cross-entropy loss, in line with Section 3.3.4. One difference, however, is that the action classes are not weighted in the loss calculation as this was found to negatively affect recognition accuracy.

Rather than use a full training epoch as the period for monitoring performance and scheduling a learning rate, we use shorter periods, which we term "mini-epochs". This allows more granular control over the learning rate scheduling. This was necessary because training for an entire epoch at a single learning rate led to overfitting. To train for mini-epochs, the training set is divided into disjoint subsets. These subsets are created by random sampling without replacement from the set of all training samples, with each sample having equal probability of being selected (i.e. uniformly). For all experiments on the 50 Salads dataset, mini-epoch training sets have a cardinality of 0.01× that of the entire training set. On average, a mini-epoch training set will have 0.3 samples per second of video, based on 30 frames per second framerate for 50 Salads. For all experiments on the Cup of Tea dataset, mini-epoch training sets have a cardinality of 0.04× that of the entire training set. On average, a mini-epoch training

set has 1.2 samples per second of video, based on 30 frames per second framerate for Cup of Tea. We treat the test sets in a similar manner to monitor convergence during training. When all disjoint subsets have been observed by the network, we generate new subsets.

A carefully managed learning rate schedule, shown in Table 4.1, was used in conjunction with the Stochastic Gradient Descent optimization technique. Nesterov momentum with a momentum value of 0.9 was used [32]. The same learning rate schedule was used across each of the single image experiments. For the experiments involving sequences of images, the Adam (Adaptive Moment Estimator) optimiser was used with initial learning rate of 0.001 and other parameters as per the paper recommendations [321]. As mentioned in Section 3.3.4, this optimisation technique precludes the need for manual scheduling of the learning rate via adapting the learning rate based on estimates of moments at the gradients.

Mini Epochs	75	100	125	150	175
Learning rate	0.1	0.02	0.004	8e-4	1.6e-4

Table 4.1: The learning rate schedule.

A batch size of 46 was used when training networks that operate on single images. When training on sequences of images, a batch size of 6 was used with each sample composed of a sequence of 8 images. It was found that a lower batch size led to better performance when training on sequences of images. A temporal stride of a single frame was used. Tests with longer sequence lengths, or longer stride, did not lead to improvements in performance. All weights in the network are initialised with the 'He' initialisation method as per the Wide Residual Network [73]. Following recommendations for the Wide Residual Network [76], no biases are used in the convolutional filters. Sparse network weights during training (where many are close to zero) can lead to overfitting, and so we impose an L_2 regularisation loss during training, with scale, λ, of 0.005, to counter this effect. Dropout was further used to regularize the training of the network [36]. Specific neurons were dropped with a probability of 0.4 between each of the fully connected layers and with a probability of 0.5 after the Relu non-linearity in the residual blocks. The Tensorflow framework was used for all training for its ability to define a network using a simple declarative syntax [322]. In each experiment, a pair of Nvidia 1080 graphics cards are used for training.

4.4 Evaluation

4.4.1 Feature Fusion

In this section, we wish to evaluate the performance of the proposed early fusion technique. The base convolutional network as detailed in Section 4.2.1, and as shown in Figure 4.3, does not include any temporal fusion and hence allows us to characterise

the image feature fusion in isolation. The network is trained to classify single images into one of the 10 'eval' action classes for the 50 Salads dataset. The dataset is trained using the first four splits and tested on the final split. The test split thus represents 10 performances of preparing a salad, with 5 subjects each performing the task twice. This experiment is repeated for each image type and for the combination of all image types. The network is identical for each experiment, including the separable convolution stages, to control for any performance increase due to an increase in network parameters. We also utilise the same training method for each image type (including optimisation, learning rates, and regularisation), as detailed in Section 4.3.2.

Image Data Type	*Accuracy*	*Precision*	*Recall*	F_1
Colour	62.76	40.14	40.66	39.40
Depth	60.35	41.68	31.59	31.54
Scene Flow	46.13	20.73	18.82	16.71
Colour + Depth	69.24	48.10	41.42	41.86
Colour + Scene Flow	65.65	44.57	39.10	38.77
Depth + Scene Flow	60.84	44.92	28.97	28.99
Colour + Depth + Scene Flow	68.84	50.55	39.20	40.36
Combined	**71.84**	**50.56**	**46.73**	**46.46**

Table 4.2: Classification results for different image types for a single test split of the 50 Salads dataset, as percentages. '+' denotes ensembling by averaging the output class distributions of networks trained on single image types. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported. 'Combined' refers to the combination of all the image types being used as an input to the network.

The evaluation metrics are shown for each of the image types in Table 4.2. The accuracy reported is the number of correct samples divided by the total number of samples in the test set. The remaining evaluation metrics are as per Section 3.4.4, namely: precision is the ratio of true positives to the total number of predicted positives; recall is the ratio of true positives to the total number of positives; and F_1 is the weighted harmonic mean of the precision and recall. For each of these metrics, the value is reported as the unweighted mean across all action classes.

On inspection of the results in Table 4.2, we note that training on a combination of all the image types ('Combined') performs best. The accuracy and F_1 scores are over 9% and 7% higher, respectively, than the next best results for single image types, which are for the colour image type. This indicates that the fusion technique is effective in discovering latent relationships between colour, depth and scene flow information.

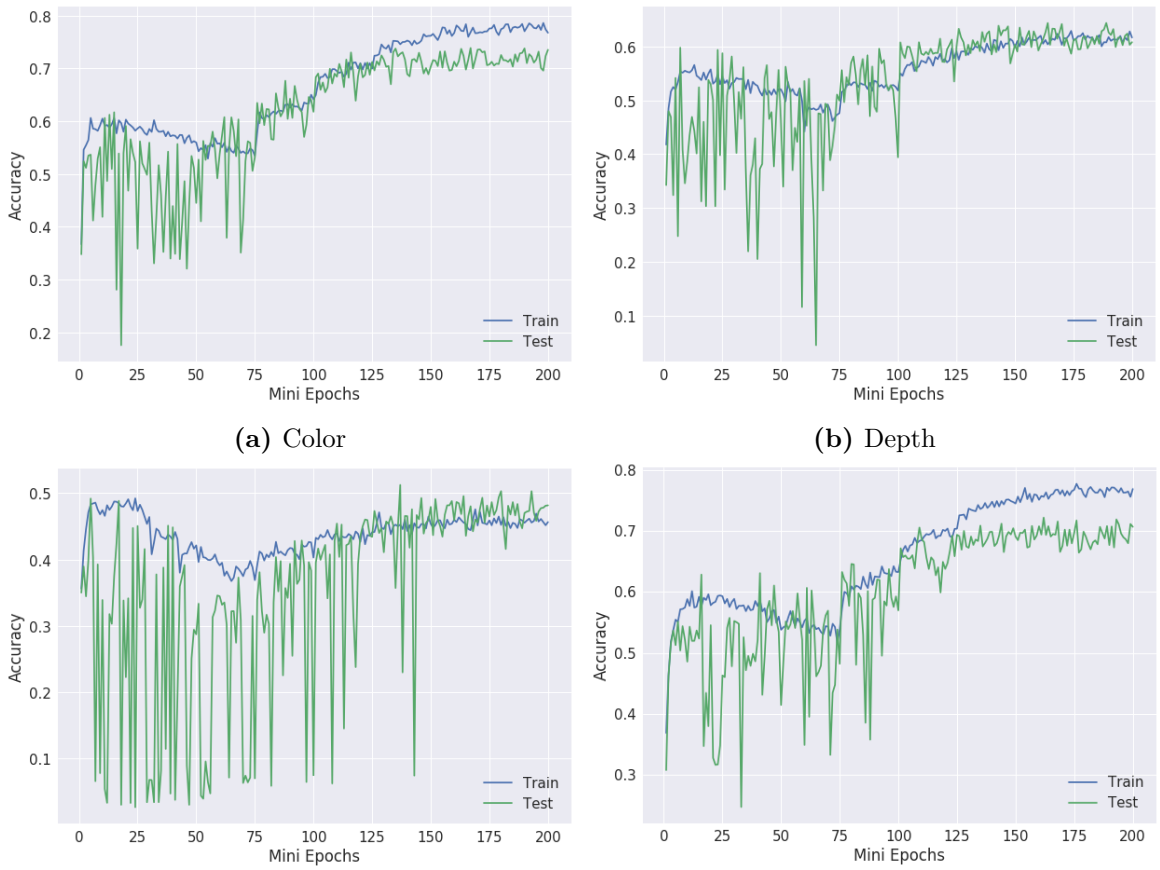
We note that the scene flow image type performs poorly upon inspection of the metrics for the individual image types. The scene flow information of a single frame, devoid of any contextual information from colour and depth, thus may contain little learning utility.

We compare our early fusion method to an alternative late fusion method. This

alternative method produces an ensemble of networks by averaging the networks' output class distributions. These networks are trained to classify based on single image types alone. We note that the ensemble of colour and depth networks (denoted 'Colour + Depth' in Table 4.2) improves performance significantly over single image networks, indicating that these image types are complementary for the classification task. A similar, albeit smaller, improvement in performance, is also noted in the ensemble of colour and scene flow networks (denoted 'Colour + Scene Flow' in Table 4.2). However, each of these ensembles has lower performance that the single network trained on a combination of all of the image types (denoted 'Combined' in Table 4.2). This indicates that our early fusion method is an effective technique of learning from multiple image types for this classification task. The alternative late fusion method also has the disadvantage of increased computational cost during inference due to the need to perform forward propagation through multiple networks. By avoiding the need to train networks for each image type, the early fusion technique is also an effective approach when training capacity is limited or when the number of image types is large.

The training profiles for each of the feature are shown in Figure 4.6. The profiles are consistent with similar reported training profiles when using residual connections [73]. In the training and testing profiles for the depth image type (Figure 4.6 (b)), there is little difference between the training and testing accuracy during the latter stages of training. There are two possible reasons for this. The network may have learned to classify solely on information that generalises well, and is thus well-regularised. The network may also be underfitting, and performance may be increased via modifications in the optimisation method and regularisation. Given that these optimisation and regularisation decisions perform well for the colour and combined image types, and that we are primarily interested in fusion of image types, we keep these decisions fixed for these experiments and leave further investigations to possible future work that utilises a separate networks for this image type.

Given that fusion of the combined image types outperforms the other image types, we wish to further characterise the technique by evaluating the method using cross-validation over the five folds of the 50 Salads dataset. To calculate evaluation metrics over the five folds, the predicted and ground truth labels for each of the test splits are combined, and the evaluation metrics are calculated for each action class and combined by averaging, as described in Section 3.3.4. This follows the recommended methodology for 50 Salads [232]. Inspecting the results, as listed in Table 4.3, the frame-wise accuracy across the entire dataset (denoted 'Accuracy') is 64.47%. This is less than the value obtained for the single split and warrants further investigation of the results across the individual splits. In Table 4.3, the evaluation metrics, precision, recall and F_1, are reported. The standard deviation across the action classes is also reported, and in each case is significant. Thus, we also investigate the classification performance across each of the action classes to understand this better.



(a) Color **(b)** Depth

(c) Scene Flow **(d)** Combined

Figure 4.6: The accuracy during training across the different features and early fusion methods. The profile for the colour (a), depth (b) and the combined (d) image types are coherent with training profiles of residual networks [73]. The training profile for scene flow (c) shows a wide variation in test accuracy during training, indicating that the network struggles to learn generalisable features from this data alone. A property of all training profiles is that the test accuracy varies significantly when the learning rate has been held constant for a large number of updates, which has also been observed during training of Wide Residual Networks [76].

Classifier	*Accuracy*	*Precision*	*Recall*	F_1 *score*
Single Image Network	64.47	49.90 ± 21.13	41.68 ± 28.61	43.10 ± 25.10

Table 4.3: The classification metrics for the cross validation evaluation using the 50 Salads dataset. All figures are reported as percentages.

The evaluation metrics across the 50 Salads test splits display a large variation, as shown in Table 4.4. Split 3 and 5 show reduced accuracy when compared to the others, indicating the network training may be suboptimal for these splits. In Chapter 3, we used a grid search technique based on the cross-validated F_1 score to ensure balanced performance across test splits. Training time is significantly longer for a CNN approach, than the approach in Chapter 3, and so such a grid search approach is infeasible.

In Table 4.5, some action classes shown significant variation in evaluation metrics. The support shows that some classes are heavily over-represented, in particular *cut ingredients*, while others are under-represented, such as *add pepper* and *mix dressing*.

Split	*Accuracy*	*Precision*	*Recall*	F_1 *score*
1	71.84	50.56	46.73	46.46
2	64.76	49.58	44.36	44.96
3	58.14	48.02	37.74	37.49
4	67.41	54.91	41.63	43.67
5	60.72	53.98	39.59	41.84

Table 4.4: The classification metrics across each of the test splits of the 50 Salads dataset.

Action class	*Precision*	*Recall*	F_1 *score*	*Support*
Background	54	80	64	137754
Add oil	56	34	42	25990
Add pepper	0	0	0	11752
Add dressing	38	21	27	20223
Mix dressing	36	5	8	18367
Mix ingredients	44	37	40	21200
Peel cucumber	74	63	68	59286
Serve salad onto plate	61	68	64	33283
Place into bowl	60	27	37	48129
Cut ingredients	77	83	80	201888

Table 4.5: The classification metrics calculated for each action class across all five splits. Support refers to the number of instances of the action class in the dataset. All figures, apart from support, are reported as percentages.

A positive correlation is observed between the metrics and the level of presentation. In particular, the *add pepper* class, which has the smallest support, is not correctly classified. This CNN approach thus struggles with under-represented classes, and techniques of overcoming such issues will be required in the later work in this thesis.

4.4.2 Temporal Fusion

Thus far, we have evaluated fine-grained action recognition methods using single images. Using sequences of images, a network can access further temporal information than is contained in the scene flow images. To determine the effectiveness of a combined CNN-RNN model, as detailed in Section 4.2.1, we perform a full cross-validated evaluation over all five test splits of the 50 Salads dataset.

For the combined model, there are different approaches to training. Given the previously trained CNN, it is possible to re-use these weights in the combined model. To test this, we devise two strategies of re-using these weights. In the first ('fixed CNN'), we fix the CNN weights during training to be equal to the weights of the network trained on a single image. Effectively, this approach reduces the CNN portion of the network to a fixed-feature extractor. In the second approach ('fine-tune'), we use the previously trained CNN weights and allow the optimisation to change these weights during training. The results of these tests are presented in Table 4.6. In these cases, the two strategies of re-use resulted in performance regressions when compared

to the CNN model ('Single image'). The strategy of optimising the network jointly ('fine-tune') outperforms the strategy of using fixed CNN parameters. These results indicate that the features learned during training on single images may not be optimal for the task of recognition of features across sequences of images.

Model	*Accuracy*	*Precision*	*Recall*	F_1
Single image	64.5	49.9	41.7	43.1
LSTM (fixed CNN)	46.1	20.7	18.8	16.7
LSTM (fine-tune)	60.4	41.7	31.6	31.5

Table 4.6: Classification results for the different CNN-RNN training combinations when cross-validated on all five splits of the 50 Salads dataset. In the above results, the precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported.

To better optimise CNN parameters for the combined CNN-RNN architecture, we train the network 'end-to-end' from initial values (He initialisation for CNN weights [73], Glorot initialisation for the recurrent cell [40]). This approach needs a longer training time, but could potentially overcome the issues detailed above where the CNN parameters are fitted to the single image approach. As detailed in Section 4.2.1, we train on two alternative RNN cells, namely the LSTM and GRU, performing temporal fusion across the output image features of the CNN. In each case, we use two RNN layers of 128 units each, and dropout inputs with a probability of 0.3 [36]. We train each model for 200 mini-epochs (where a mini-epoch corresponds to 0.001 of the full training set). The results of these two experiments are presented in Table 4.7.

Model	*Accuracy*	*Precision*	*Recall*	F_1
Single Image Network	**64.5**	49.9 ± 21.1	41.7 ± 28.6	43.1 ± 25.1
LSTM Temporal Fusion	51.4	44.7 ± 21.9	50.5 ± 14.8	43.5 ± 13.5
GRU Temporal Fusion	60.1	**52.8** ± 18.9	**56.2** ± 15.4	**50.9** ± 12.6

Table 4.7: Classification results for different temporal fusion approaches tested, as percentages, when tested using cross-validation on all five splits of the 50 Salads dataset. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported. The Single Image Network is that of Section 4.4.1. The LSTM Temporal Fusion and GRU Temporal Fusion utilise LSTM and GRU cells respectively as part of the combined CNN-RNN architecture detailed in Section 4.2.1.

We observe from these tests that the combined CNN-RNN leads to a reduction in the total number of correctly classified samples, i.e. accuracy, when evaluated using cross-validation over five test splits. We note from Table 4.7 that the GRU Temporal Fusion approach leads to improved precision, recall and F_1 scores over the Single Image Network. For the GRU Temporal Fusion approach, the standard deviation of these metrics is also less. Compared to the LSTM Temporal Fusion approach, the GRU Temporal Fusion approach outperforms it across all metrics. This may be due to the

rationale discussed in Section 4.2.1 that the LSTM is better suited to modelling long-term dependencies that do not exist in the short clips that we use in training. To further elucidate the performance advantages and disadvantages of the GRU Temporal Fusion approach, we examine the metrics for this approach across individual action classes.

Action class	*Precision*	*Recall*	F_1 *score*	*Support*
Background	69	59	64	137754
Add oil	30	82	44	25990
Add pepper	24	65	35	11752
Add dressing	34	46	39	20223
Mix dressing	72	27	39	18367
Mix ingredients	50	38	43	21200
Peel cucumber	62	59	61	59286
Serve salad onto plate	57	70	63	33283
Place into bowl	44	51	47	48129
Cut ingredients	86	66	74	201888

Table 4.8: The classification metrics for the GRU Temporal Fusion approach calculated for each action class across all five splits. Support refers to the number of instances of the action class in the dataset. All figures, apart from support, are reported as percentages.

In Table 4.8, we see the performance of GRU Temporal Fusion across the action classes. We note significant differences of individual class metrics between the Single Image Networks results of Table 4.5 and the GRU Temporal Fusion results. This indicates that the GRU Temporal Fusion approach learns a different representation of the classes. In particular, the class 'add pepper' which is never correctly classified by the Single Image Network shows improved classification metrics. This indicates that the presence of further image data via the combined CNN-RNN approach allows these under-represented actions to be better classified.

A further evaluation of the GRU Temporal Fusion results across datasets splits, reveals performance disparity across the splits. This indicates that performance may be improved via further investigation of hyperparameters, as in the Single Image Network case. Further details of this analysis are available in Appendix H.

4.4.3 Method Comparison

In the previous chapter, we detailed an approach to recognising the actions involved in the preparation of a cup of tea. This involved using pose estimates of objects and arms as inputs to a neural network architecture (involving LSTMs). The approach outlined in this chapter uses a CNN to produce image features to use as input to a recurrent neural network, rather than pose information. These two approaches should be compared for applicability as part of pervasive computing applications for human-object interactions.

To compare the two approaches, we train the three neural network architectures outlined in Section 4.2.1 against the Cup of Tea dataset. In each case, the networks and parameters are identical to those utilised in Sections 4.4.1 and 4.4.2. The optimisation method used in each case is Adam [321], with a learning rate of 0.001. This method is used in place of stochastic gradient optimisation, to avoid the need of identifying an optimal learning rate schedule for this smaller dataset. From the dataset, we extract the colour and depth images from the central camera and extract the corresponding scene flow information. In the analysis below, we shall refer to the approach of Chapter 3 as 'PoseLSTM' for brevity.

Model	*Accuracy*	*Precision*	*Recall*	F_1
PoseLSTM	82.90	**81.46** ± 8.65	**82.20** ± 7.62	**81.72** ± 7.49
Single Image Network	81.28	79.47 ± 9.55	78.49 ± 11.15	78.92 ± 10.13
LSTM Fusion Network	**83.01**	80.47 ± 8.30	82.01 ± 12.26	80.81 ± 8.86
GRU Fusion Network	82.47	81.13 ± 7.39	80.85 ± 13.10	80.53 ± 8.79

Table 4.9: Classification results (in percentages) for the different approaches when tested on the 'Cup of Tea' using cross validation over all eight folds. In the above results, the precision and recall are calculated for each action class, and the unweighted mean across classes is reported.

The results of the cross-validation analysis are presented in Table 4.9. We observe that the PoseLSTM approach is outperformed in terms of overall accuracy by the LSTM Fusion Network. The performance across the other evaluation metrics are less for the CNN-based networks; however only slightly.

Besides the performance differences of the approaches, there are other concerns relevant to this method comparison. An advantage of the CNN-based approach is that the entire network is trained in an end-to-end fashion, without distinct pose estimation stages. Pose estimation was challenging in cases of changing topology of objects (3.4.3) and affected classification results. The arm pose estimation method (3.2.3) is not guaranteed to capture fine-grained hand movements, such as those involved in OSCEs. Another point to note is that the pose-based approach required the deployment of three cameras to ensure that objects are reliably tracked across the entire performance of a task. The performance of the CNN-based approaches indicates that a single camera approach is feasible for achieving high recognition accuracy. This removes the logistical and computational requirements associated with capturing data from multiple cameras.

4.5 Discussion

In this work, we demonstrate an early fusion approach as an appropriate alternative to training parallel CNN branches when performing action recognition tasks across multiple image types. Furthermore, we have proposed the use of scene flow information

for the recognition of actions from RGB-D data using a CNN-based approach. As fine-grained actions have jointly distinctive colour, depth and motion patterns, we argue that early fusion is reasonable, and demonstrate performance increases over training on single image types and over an alternative late fusion approach. Another advantage of this early fusion approach over training parallel branches is that there is a reduction in network parameters, and hence training cost.

In this chapter, we have demonstrated the ability to use the early fusion method as part of a combined CNN-RNN architecture. This approach improved recognition performance across certain metrics; however, it resulted in a reduction in overall accuracy. Given the extra information available when learning from sequences of images, it is plausible that performance can be improved by treating the temporality of the data differently. The early fusion approach outlined in this chapter is efficient compared to a parallel approach, in that it utilises a similar number of parameters as a single image recognition network. This represents an approach that may be preferable when computing power is constrained. However, there may exist opportunities for improvement by performing fusion efficiently at additional locations in the network. In the following chapter, we shall focus on these aspects of improvement and demonstrate the method on a collection of recordings of OSCE performances.

Chapter 5

WeaveNet – Fine-Grained Action Recognition by Dense Fusion of Image Types

In this chapter, we investigate spatio-temporal neural network architectures for fine-grained action recognition. In the previous chapter, feature fusion early in the network, via separable convolution, provided a method of learning from latent multi-modal features, while removing the expensive need of training multiple parallel branches for each image type. However, fusion at other stages of the network may improve performance further. Accordingly, we propose *WeaveNet*, a new architecture that permits fusion of the image types throughout the network. To achieve this, a thin shallow parallel network branch maintains separation of features by image type. These image features are combined with the main network branch in a staged fashion, so the network learns to fuse these features multiple times. This approach maintains efficient scaling with multiple image types. A method of performing fusion over temporal information is also proposed and shown to improve results. The WeaveNet architecture outperforms published works across multiple action granularities on the 50 Salads dataset, including a 9% accuracy increase (from 73.4% to 82.7%) for the 'evaluation' granularity. Finally, we discuss the collection of a significant dataset of OSCE performances and evaluate our devised architecture against this new dataset for a single camera viewpoint.

5.1 Introduction

5.1.1 Motivation

In this chapter, we investigate techniques of fine-grained action recognition using multi-modal image features as inputs to a neural network and specifically test against the fine-grained actions involved in OSCEs.

In the previous chapter, a fusion technique for input image types was devised,

specifically an early fusion using depth-wise separable convolution. The reasoning behind early fusion is based on observations of latent multi-modal structures across depth and colour images [13]. In our case, we have more than two image types available, and thus we wish to examine fusion techniques that work across an arbitrary number of image types. In a recent work [138], methods of fusion of parallel colour and optical flow networks for action recognition were studied. Here, the skip connection pattern of residual networks [73] was used to fuse network branches, and more optimal fusion locations and directions were identified. There is less work performed on fusion methods for multi-modal data obtained from RGB-D videos using CNNs, with works training separate networks and fusing by averaging [226, 14]. With the availability of depth information, there is an increase in the available image types to train on (e.g. normal maps, depth maps), and so a method of fusion that works over an arbitrary number of image types in an efficient manner is desirable. Furthermore, fusion techniques for motion information in the form of scene flow have yet to be explored in the literature. We reason that the dense connectivity pattern [22] is an appropriate technique for this problem and incorporate it into our WeaveNet architecture for multi-modal action recognition.

In this work, OSCE performances are recorded with fixed camera placements and labelled with multiple actions that exhibit high similarity. The problem of recognition of these fixed-setting fine-grained actions differs from the more general problem of action recognition. As observed in the previous chapter, motion information provides less utility for recognition of fine-grained actions than for coarse action classes such as sports [127]. For fine-grained action recognition, subtle contextual details, such as having put on gloves in an OSCE performance, could provide information to allow discrimination of fine-grained actions. Lea et al. [19] tackle this problem using a VGG-like CNN [39], trained on colour and motion history images [334], to extract image features from videos of human-object interaction tasks. They propose performing action segmentation distinctly once these image features have been generated for the entire sequence. Such segmentation approaches are of orthogonal interest to this work, due to the motivation of use of recognition as part of an online pervasive computing application. We focus on optimising the first stage of such an approach, recognition of fine-grained actions using CNNs trained on spatio-temporal information. The use of a segmentation stage in combination with our fine-grained action recognition approach is not precluded by this approach.

OSCEs are particularly suitable to study using fine-grained action recognition techniques given their fixed and controlled setting and the limited set of possible actions that can be performed. To validate our devised techniques for use in this domain, the collection of a significant OSCE dataset is required. In section 2.2.6, we reviewed datasets used for the problem of fine-grained action recognition and noted that there does not exist one of OSCE performances. The contribution of an OSCE dataset may

allow building systems to solve real-world problems regarding the training of medical professionals. Furthermore, the fixed nature of the OSCE setting affords flexibility of arrangement, and as such, a dataset that allows for a wide range of approaches would be useful to future research.

5.1.2 Contributions

The following contributions are made in this chapter:

- **WeaveNet**, a CNN architecture for fine-grained action recognition from multiple image types;
- **Densely-Fused Action Images**, a method of learning a spatio-temporal representation for use in the WeaveNet architecture; and
- **OSCE-V**, a significant dataset of venepuncture OSCE performances recorded using multiple RGB-D cameras.

In this chapter, we contribute a novel CNN architecture, **WeaveNet**, for fine-grained action recognition in a fixed setting. This architecture is demonstrated to learn effectively from multiple image types. It scales efficiently with increasing numbers of input image types, without requiring expensive parallel branches for each image type [137]. To learn effectively from multiple image types, it leverages the dense connectivity pattern of the Densely-Connected Convolutional Networks (DenseNets) [22]. This architecture was initially proposed for image recognition problems and showed state-of-the-art performance on benchmark datasets. In the WeaveNet architecture, efficient fusion is facilitated by the inclusion of a single thin parallel network branch that maintains separation of features of different image types (e.g. colour, depth, and scene flow). These features are combined with the main network branch in a staged fashion so that the network learns to fuse features of different image types multiple times. This removes the problem of having to identify the most strategic location of fusion of features of multiple image types. The WeaveNet architecture is shown to be a valid contribution based on experiments showing that it improves recognition accuracy over baseline models and outperforms other published techniques on a benchmark fine-grained action recognition dataset [232].

In this chapter, we propose a method, **Densely-Fused Action Images**, for temporal fusion as an integral part of the WeaveNet architecture. The deep learning approach of Chapter 4 tacked the spatial and temporal aspects of the information distinctly, with CNNs and recurrent neural networks, respectively. This approach required sharing the weights of the convolutional layers across the temporal domain, increasing the computational cost linearly with increasing number of timesteps. In this work, we propose an efficient alternative method for temporal fusion. This approach involves learning a fusion of motion and colour information, applicable across the temporal dimension,

to produce an image-like representation of spatio-temporal information. The novelty of this approach is that dense connectivity is used to fuse and transform image types, such as colour and scene flow, for use in the generation of this representation. Including spatio-temporal information in this way increases performance when used in combination with the WeaveNet architecture to perform fine-grained action recognition in a fixed setting.

In this chapter, we present a new dataset, **OSCE-V**, of performances of the OSCE of venepuncture. This OSCE requires the performance of a goal-directed human-object interaction task that involves drawing blood from a prosthetic arm according to a strict set of instructions. This dataset is recorded with RGB-D cameras from three vantage points, using the methodology detailed in Chapter 3. A dataset for the recognition of the fine-grained actions involved in an OSCE performance does not currently exist in the literature. Furthermore, there does not exist another dataset that incorporates multi-modal recording, multiple vantage points and fine-grained action labelling of a human-object interaction task. This dataset could form the basis of future research into pervasive computing applications for OSCEs. This dataset represents a contribution due to the range of recognition approaches afforded by it, the novelty of the application domain and its ability to form the foundation of future systems.

5.2 Design

5.2.1 Base Architecture

In Chapter 4, we used a combination of different image types to recognise fine-grained actions. We observed that fusion of colour, depth and scene flow early in the network improved recognition results. However, these image types may contain differing amounts of discriminative information. Hence, fusion of these image types at a single early location in the network may not be ideal. For example, flow information may be more informative once the network has recognised an object in a certain location. Also, there is the possibility that one image type dominates after the early fusion and so discriminative information present in the other image types may not be fully utilised. Furthermore, the differing complexities of these image types may mean that fusion of these image types at the same level of a convolutional architecture may also not be ideal. The level of abstraction, and hence the depth of the network, needed for processing each of the different image types may vary. We wish to identify a network architecture that is amenable to being used to tackle these problems.

Similar to the approach of Chapter 4, and to other neural network approaches to action recognition [16, 138], we use an existing CNN architecture to bootstrap our action recognition approach. The architecture that we select is the Densely-Connected Convolutional Network (DenseNet) [22]. The architecture addresses the problem of diminishing performance increases with increasing depth of residual networks by rein-

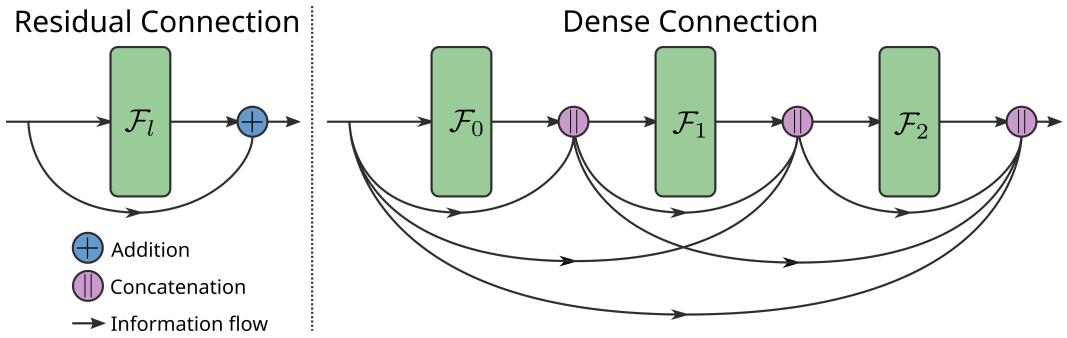


Figure 5.1: Residual connections differ from dense connections in the number of incoming connections to a layer and in the method of combination (addition and concatenation respectively).

troducing intermediate outputs as inputs to later layers. This strengthens feature propagation and encourages feature re-use, which are aspects that may be beneficial to the problem of learning across multiple image types.

To clearly describe the architecture of our later networks, an explanation of dense connections used in DenseNets is provided. As described in the original work [22], an input image to a network is denoted as $\mathbf{x}_0$, and the independent non-linear transformation of a network layer l as $\mathcal{F}_l(\cdot)$. In a feed-forward CNN architecture, such as the VGG network [39], an intermediate output, $\mathbf{x}_l$, is the result of the application of the transformation to the previous input data, $\mathbf{x}_{l-1}$,

$$\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1}). \tag{5.1}$$

The residual network approach re-framed the non-linear transformation as a residual function that alters the original data,

$$\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1}. \tag{5.2}$$

Graphically, this adds a connection that bypasses the transformation, connecting with the output of the transformation by addition, as per the leftmost image of Figure 5.1. Data flows unchanged along this introduced connection, and so such a connection is termed an identity *skip connection*. This connection allows gradient to flow to earlier layers during backpropagation, facilitating deeper architectures.

Dense connections differ in that multiple preceding layers are connected to the current layer,

$$\mathbf{x}_l = \mathcal{F}_l([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}]). \tag{5.3}$$

Here, the previous outputs $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}$ are combined by concatenation. This contrasts with the combination by addition of residual networks, as can be seen in Figure 5.1. The additional connections of the DenseNet approach offer further paths along which gradient can flow during backpropagation.

The full DenseNet architecture also includes other details that are relevant to our

future architecture descriptions:

- For the nonlinear transformation, $\mathcal{F}_l$, the network uses batch normalisation [37], rectified linear unit (ReLU) activation and a convolution with a 3×3 filter, in succession. This is collectively referred to as a *dense layer*.
- As dense connections must connect feature maps of equal spatial resolution, the DenseNet architecture is broken up into blocks of densely-connected layers, called *dense blocks*, to facilitate progressive downsampling of the data.
- To downsample, batch normalisation followed by 1×1 convolution and 2×2 average pooling is performed in collectively what is called the *transition layer*.
- In DenseNets, the concatenation operation allows layers to output varying numbers of output features. This number is controlled by the *growth rate* parameter.
- A *bottleneck layer* applies batch normalisation, ReLU and 1×1 convolution to reduce the number of input features to each dense layer $\mathcal{F}_l$.
- A 1×1 convolution is also used to reduce output features of a transition layer, and is controlled by the *compression rate*.
- Dropout [36] can be used to regularise the network by dropping certain outputs of each dense layer $\mathcal{F}_l$.

For recognition across multiple image types, the DenseNet architecture has the following desirable properties:

1. As mentioned in Section 2.2.2, Feichtenhofer et al. tested methods of combining intermediate outputs of appearance and motion streams in a two-stream architecture and found that concatenation followed by convolution performed best [137]. Such a pattern of operations is characteristic of the DenseNet architecture, as illustrated in the rightmost image of Figure 5.1. This property will be used to combine features of different image types within a network to take advantage of this increased performance. More recent research has explored other methods of performing fusion across parallel CNN branches [151, 141] as discussed in Section 2.2.2. SlowFast networks [141], which achieves state-of-the-art performance on datasets such as AVA [241], uses a novel fusion method to fuse Fast branch features into the Slow branch. The best performing fusion technique for this 3D architecture is found to be fusion by concatenation, with Fast branch features undergoing a strided temporal convolution prior to concatenation. Striding is used to reduce the temporal dimension size of features to allow concatenation between the branches. This convolution operation is pointwise in the height and width dimensions (i.e. a $5 \times 1 \times 1$ filter for time, height, width respectively). Such fusion techniques could feasibly be explored in future work to improve performance of our WeaveNet architecture.

2. An advantage of the DenseNet architecture is that it promotes strong gradient flow during backpropagation. The dense connections provide deep layers greater access to the gradients from the loss function, increasing supervision of these layers. This deep supervision can **promote more effective fusion of image types in early layers**.

3. The DenseNet architecture **overfits less** compared to residual networks when data augmentation is not used [22]. This property is desirable in our case, as data augmentation is less applicable due to fixed camera placement.

4. As a result of the concatenation of output features, the **features of earlier layers are re-used** in later layers. This property may allow unfused or partially-fused image features to be re-used later in the network, potentially at a more beneficial location for fusion purposes.

5. Due to the dense connections, **earlier low-complexity features are combined with later high-complexity features**. Combining image types with dense connections may help overcome the problem of identifying the ideal fusion locations by fusing images types at different levels of processing.

6. The DenseNet architecture is more **parameter and computationally efficient** than residual networks. For training on multiple image types, this has the advantage of reducing the hardware and time requirements for training.

These advantages make the DenseNet architecture particularly attractive for the problem of action recognition by fusion of image types.

The particular arrangement of DenseNet, used as the base architecture, requires specification. In initial experiments it was found that three dense blocks performed better than four dense blocks. The initial convolution layer, which precedes the first dense block, uses a stride of 1 to maintain the same feature dimensions and a filter size of 3×3. Preceding each convolutional layer in the dense blocks, a bottleneck layer is used to reduce the channel-wise dimensions. Furthermore, the compression rate in the transition layers is set to reduce total input features to each block. The base DenseNet architecture, shown in Figure 5.2, remains fixed throughout the remainder of the chapter.

5.2.2 WeaveNet

The goal for the WeaveNet architecture is to permit fusion of features of distinct image types to occur throughout the network. Previous methods have selected, on an ad-hoc basis, locations at which to perform fusion of features of different image types [16, 137, 138]. We have identified dense connections as a useful pattern for this purpose, as these connections can concatenate unfused input features to the fused

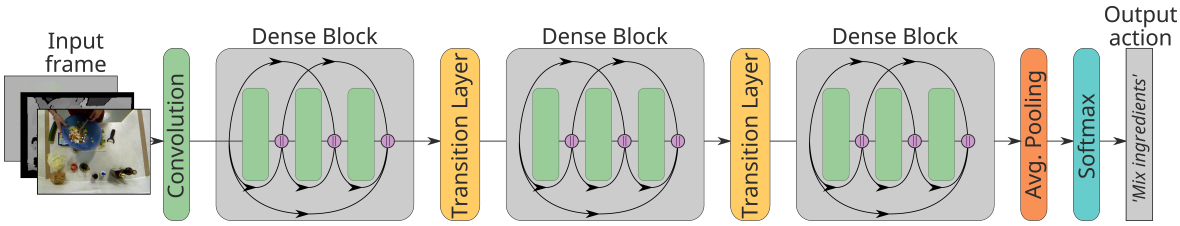


Figure 5.2: An illustration of the base DenseNet architecture used in our experiments. In the figure, each dense block is composed of only three nonlinear transformation layers (in green) for illustrative purposes. Six layers are used in each block in our actual base architecture. Each nonlinear transformation layer is composed of a bottleneck layer followed by a dense layer. A final linear transformation layer with softmax activation (Softmax Layer) is used to output action class probabilities.

output features of a dense layer. The DenseNet architecture utilises these connections, and so has properties that are advantageous for fusion, however certain properties of the architecture present issues for our problem. The issues relate to the initial convolutional layer and the transition layer (as shown in Figure 5.2). Our WeaveNet architecture directly tackles these issues to permit fusion of different image types throughout the network. In the following sections, we outline these issues and detail our solutions.

Initial Convolution

In the DenseNet architecture, the initial convolutional layer takes a single image as input. To train on the base architecture, image types are concatenated to form a single image that is passed as input. The convolution operation of this layer operates over the input, and so the layer outputs a fusion of the image types. This early fusion may lead to certain image types dominating the feature learning in the remainder of the network. Early fusion also prevents maintaining separation of features by image type within dense blocks.

A possible solution to this fusion problem is to entirely remove the initial convolution layer. However, this layer increases the receptive field (the effective area of an input image that a specific convolution operation has access to) for the convolution operations of the first dense block. Another possible solution is to modify the convolution operation itself so it does not fuse the image types.

Previously, in Section 4.2.1, we used a depthwise convolution operation as the first stage of the two-stage separable convolution operation. A depthwise convolution operation maintains separation of channel-wise features. This depthwise convolution can be modified so that instead of operating over single image planes (i.e. channels) it operates over groups of image planes (i.e. image types). How this convolution operation, which we refer to as *groupwise convolution*, relates to the previously used depthwise convolution is described in Figure 5.3. A requirement of this operation, is that each image type is encoded with an equivalent number of channels, shown as *id* in the Groupwise Convolution illustration in Figure 5.3. For example, if we use colour,

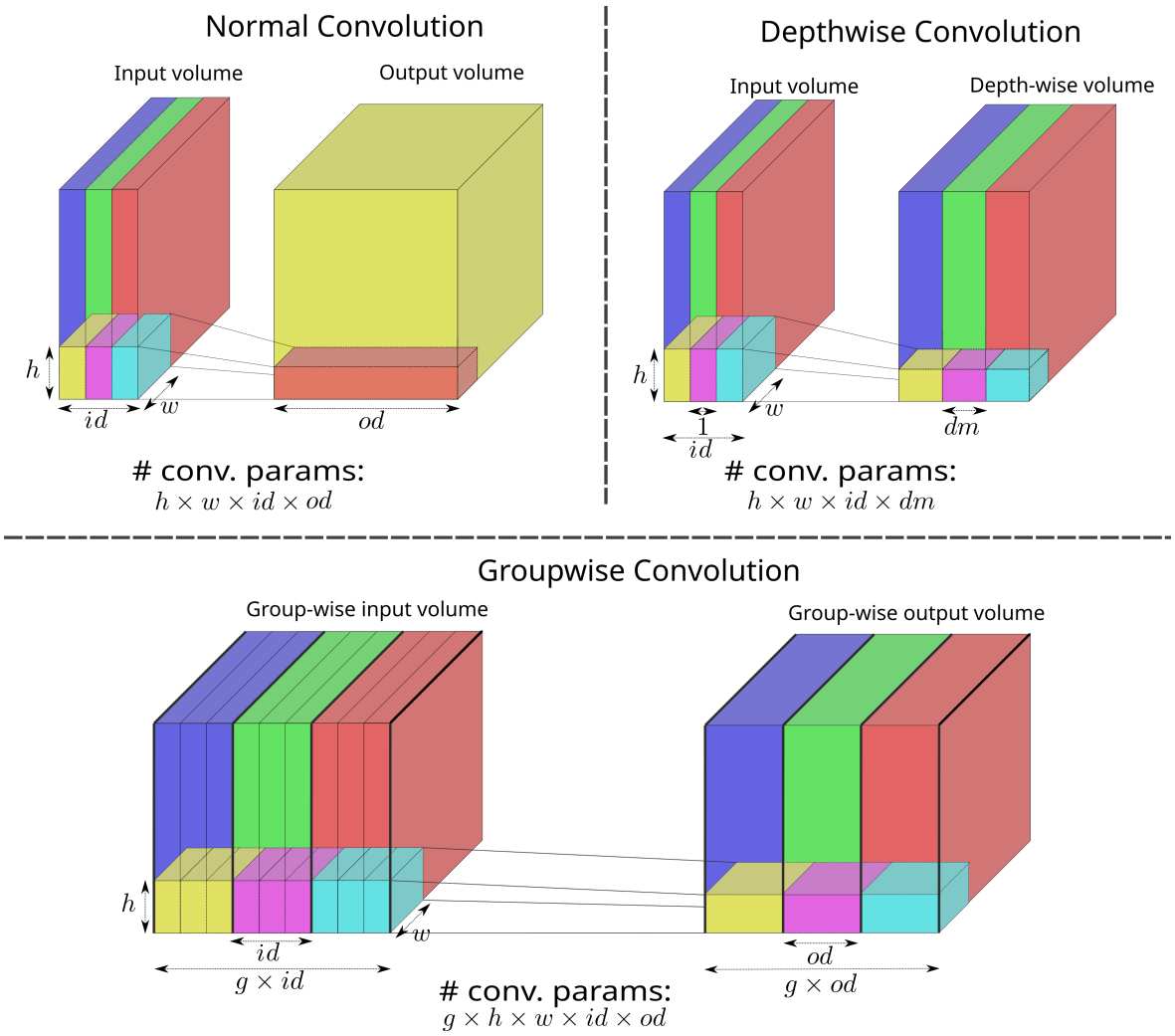


Figure 5.3: An illustration of the differences between normal convolution, depthwise convolution and groupwise convolution. In normal convolution, a single convolution operation, with input size id and output size od, is applied across all channels of the input. In depthwise convolution, convolution operations, with input size 1 and output size dm, are applied to each of the respective channels of the input. Depthwise convolution works for an arbitrary number of input channels, id (shown as 3 in the figure). In groupwise convolution, convolution operations, with input size id and output size od, are applied to the respective groups of input channels of size id. The number of groups, g, can correspond to the number of image types used, and id can correspond to the number of channels per image type. Depthwise convolution is very similar to groupwise convolution in that distinct convolution operations are applied to the input. However, in depthwise convolution, operations are applied per channel, whereas in groupwise convolution, they are applied per group (or image type as in our case).

depth and scene flow as image types, these image types would need to be encoded using an equal number of channels. By using this convolution operation, and assigning a group to each image type, the output convolutional features are separated by image type, and so the input to the first dense block is in an unfused state. Thus, the dense connections in the first block pass image features separated by type to each dense layer in the block. This allows the dense layers to learn a fusion of different image types. This property is also desirable in the subsequent dense blocks, however, another property of DenseNets prevents this.

Transition Layer

The presence of the transition layers in the DenseNet architecture impedes the fusion of distinct image types in all dense blocks. Two properties of the transition layer prevent features of different image types being forwarded to the subsequent dense blocks. Firstly, the transition layer contains a convolution operation. This has the same undesirable effect of fusing the features of different image types as the previously-described initial convolution had. Secondly, the transition layer prohibits fusion by reducing the feature spatial size via an average pooling operation. This change in feature dimension restricts fusion-promoting dense connections to features within a single dense block, as equivalent spatial sizes are required for concatenation. The complexity of image features increases as CNN depth increases. Hence, the transition layer limits dense connections between a fixed range of feature complexity. Ideally, we want to combine a range of complexity of features of different image types throughout the network. For example, image types, such as flow information, may be less detailed, and may need to undergo fewer transformations than other image types, such as colour information, prior to fusion. Our *WeaveNet* architecture directly tackles these issues to permit fusion of a larger range of complexities of distinct image types throughout the network.

Removing the transition layer is a possible method of overcoming the prevention of fusion by transition layers. However, reduction of image feature spatial size with increasing depth is characteristic of CNNs, and so the transition layer plays an essential role. Alternatively, we seek a method of maintaining image features in an unfused state at decreasing spatial resolutions. By introducing a parallel network branch with this property, we can use this branch to concatenate unfused image features at the start of each dense block. In this network branch, we can maintain a limited number of features (i.e. thin) and limited number of convolutional operations (i.e. shallow) to minimise the additional computational cost. We specify an appropriate architecture for this branch.

Ideally, we desire a range of complexity of features of all image types to be concatenated at the beginning of each dense block in the main branch. We have already identified the dense connectivity pattern as a method of achieving this range of complexity, and so use this pattern in the parallel branch. The dense blocks used in the DenseNet architecture are amenable to use in a thinner and shallower configuration. By splitting features by image type, we pass image features through individual branches composed of a single thin dense block. Each thin dense block outputs features of a range of complexity for each image type. Collectively, we refer to a group of thin dense blocks for the image types as a *Strand Block*, and we refer to these individual branches as *Strands* to emphasise the fact that these branches are metaphorically thin (as shown in Figure 5.4).

A method of decreasing image feature spatial size, to allow for concatenation at the

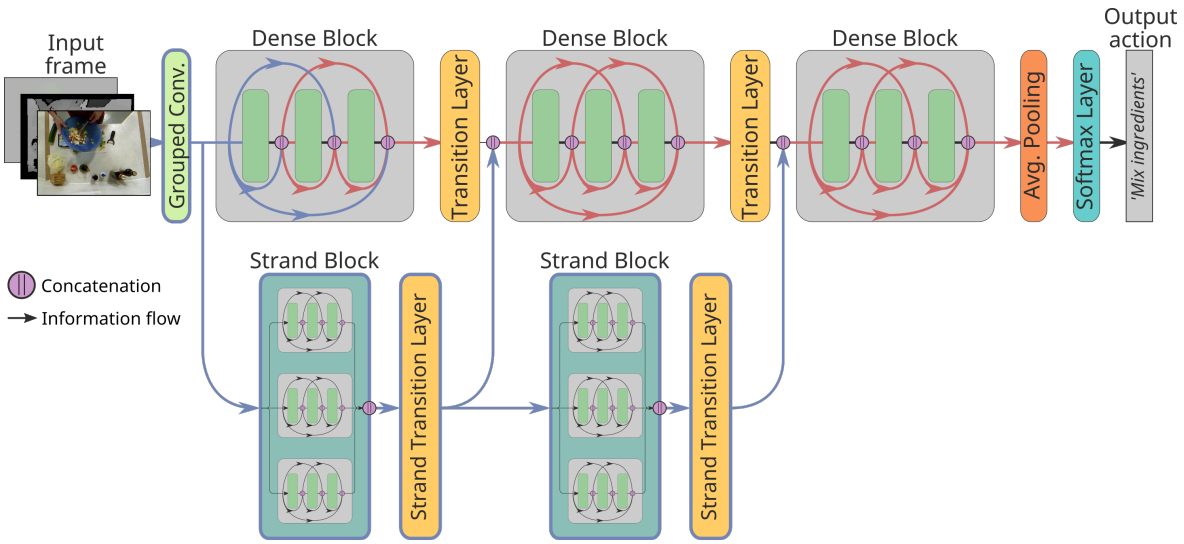


Figure 5.4: The WeaveNet architecture introduces a parallel network branch to maintain separation of the image types. In each Strand Block, image features are separated by image type and passed through a thin dense block. In the Stranded Transition Layer, groupwise convolution is used to maintain separation of the different image types. Blue lines represent paths along which *all* features are separated by image type. Red lines represent paths along which *some* features are separated by image type. As both these types of paths are present throughout the network, fusion of features of distinct image types occurs across the entire network.

beginning of the dense blocks in the main branch, remains to be specified. We propose the use of what we refer to as a *Strand Transition Layer*. Its design mirrors the DenseNet transition layer, being composed of batch normalisation, ReLU, convolution and average pooling. It differs from a transition layer, however, by using a groupwise convolution in place of the convolution operation. This solution maintains separation of image features, similar to how it did in our solution to the initial convolution problem.

The strands are connected to the main branch via concatenation of the outputs of the Strand Transition Layer to the outputs of the Transition Layer of the main branch, as shown in Figure 5.4. This concatenation ensures that simpler unfused image features of stand blocks are combined with more complex features present in the outputs of the transition layers. Fusion of these unfused image features thus occurs throughout each dense block via the dense connections. This overall network architecture is called *WeaveNet*, based on the metaphor of thin strands being weaved into the main branch of the network.

The number of possible paths of information through the network is increased as a result of the addition of the parallel branch. In Figure 5.4, the forward propagation direction of image data is shown, with coloured paths highlighting paths along which unfused data flows. The paths through the network also allow multiple possible paths for gradient to flow to each dense layer during backward propagation. The variety of paths along which the gradient can flow adds greater learning ability to the Strand layers. This should encourage the strands to learn features for each image type that can be effectively fused in the main branch.

We have developed a new architecture that permits fusion of features of different image types throughout a CNN. Similar to how SpatioTemporal Residual Networks [138] leverage the residual network architecture [73] to perform action recognition, we leverage the DenseNet architecture [22] to perform fusion of multiple image types for fine-grained action recognition. The use of thin parallel strands avoids the expense of full parallel branches for each image type. The dense connectivity pattern used also removes the requirement of deciding where to fuse individual branches of each image type. Fusion of distinct image features, of varying complexity, now occurs throughout the network.

5.2.3 Stranded Late Fusion

The parallel network branch present in the WeaveNet architecture is used to inject unfused image features into each dense block to promote fusion of image types. However, recent works utilising parallel networks for action recognition improved performance by performing late fusion of two parallel networks (i.e. fusion at the end of the network) [138, 137]. Specifically, for SpatioTemporal Residual Networks [138], late fusion of a residual network branch for appearance data (colour) and a residual network branch for motion data (flow) increased recognition accuracy. In our architecture, we perform fusion of these (and other) image types throughout our main network branch. However, performance may be improved by additionally performing a late fusion of unfused features of different image types. We identify further architectural changes to perform this late fusion efficiently and effectively.

When separate network branches are maintained for multiple images types [138, 137, 16], late fusion can be performed by combining the outputs of the separate streams using further neural network layers or discriminative classifiers such as support vector machines [16]. In our architecture, the strands maintain separation of the different image types. The purpose of these parallel strands is to inject image features in an unfused state into each dense block to ensure that fusion of image features at differing complexities occurred throughout the network. However, as image types are separated between strands, it may also be possible to use these strands to perform late fusion.

In the WeaveNet architecture, the features output by the final Strand Transition Layer are concatenated to the features of the main branch before the final Dense Block. These strand features are likely to be less complex than the features in the main branch that they are being combined with, due to undergoing fewer convolutional operations in the Strand Block path. This is not an issue for the WeaveNet architecture as the features will be used to learn more complex features as part of the final Dense Block. Combining high-complexity features in a late fusion would be more beneficial, as evidenced by the results of late fusion of two-stream networks [137]. To produce higher complexity features for each image type, the depth of the strand paths (i.e. the number of convolutional operations on the features) should be increased. To achieve

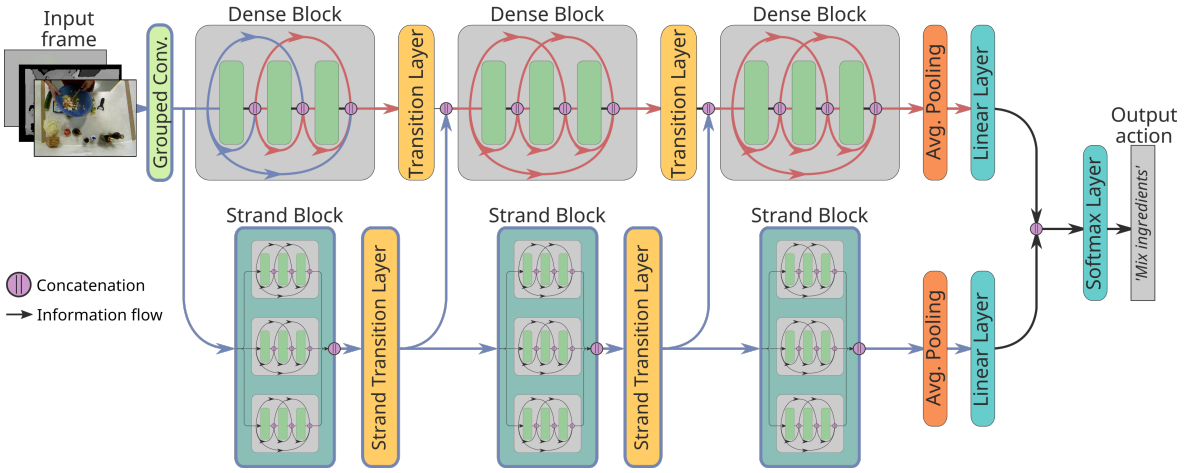


Figure 5.5: The WeaveNet-LF architecture includes an additional Strand Block compared to the WeaveNet architecture. The outputs of this Strand Block undergo average pooling, batch normalisation, and rectified linear (ReLU) activation before being used as inputs to a linear layer. The outputs of this linear transformation are combined by concatenation with similarly pooled, activated and transformed outputs of the main network branch. The combined outputs undergo batch normalisation, ReLU activation before being fused in a final linear layer with softmax activation (Softmax Layer). Blue lines represent paths along which *all* features are separated by image type. Red lines represent paths along which *some* features are separated by image type. The additional Stand Block, with input and output blue paths, thus maintains separation of features by image type prior to late fusion.

this, the WeaveNet architecture is modified to include an additional Strand Block. This will increase feature complexity while maintaining separation of these features by image type.

The precise method of late fusion of these more complex features needs to be specified. One possibility is to concatenate these features to the output features of the final Dense Block in the main branch of the network (see Figure 5.4). This would significantly increase the number of input features for the final Softmax Layer (i.e. a Linear Layer with Softmax activation). As a result, this layer would be highly parametrised, which can lead to overfitting, and hence underperformance, without careful regularisation. Alternatively, we could use further linear layers to reduce the number of features before the final softmax layer, as shown in Figure 5.5. Further linear layers divide the late fusion stage into two stages: fusion of the strands' features; and fusion of these late-fused outputs with the main branch features. To prevent overfitting, we limit the number of outputs, and hence parameters, of the linear layers.

The late fusion method described here enables complex features of each image type to be combined effectively. The dense connectivity pattern within the additional Strand Block enables complex features to be produced efficiently. Fully-connected linear layers are used in a staged fashion, to eliminate over-parametrisation, when combining late-fused features with the main branch features. We shall refer to this architecture, shown in Figure 5.5, as the *WeaveNet-LF*.

5.2.4 Densely-Fused Action Images

Thus far in this chapter, we have mainly focussed on methods of improving recognition performance based on the spatial component of fine-grained actions (i.e. the relative locations of features across image types), however, it is also important to consider the temporal component of actions (i.e. how these features change over time).

Training on inputs of representations of motion information is a method of incorporating temporality in an action recognition approach. Such representations can include hand-designed features such as Histograms of Optical Flow [98] and Motion Boundary Histograms [96]. As CNNs can train on entire images, optical flow images are a commonly used motion representation for CNN action recognition approaches [137, 127, 16]. The quality of motion information used greatly affects the ability to learn efficient representations for recognition purposes [135]. Scene flow information, which we shall use, offers a richer representation of motion compared to optical flow, representing motion in 3D Cartesian coordinates, compared to the 2D screen space coordinates of optical flow.

Motion representations, such as optical flow images, are more informative than colour images for recognition of certain actions [127]. For example, recognition from single optical flow images outperforms recognition from single colour images for actions such as 'push-ups' and 'jump rope', due to specific motion patterns but ambiguous spatial contexts present in optical flow images [127]. Fine-grained actions, such as those involved in an OSCE, however, may exhibit more subtle differences in motion patterns across actions classes. Furthermore, the motions may be sparse, both spatially (i.e. occupy a small area of images) and temporally (i.e. occur in a small subset of a set of sequential frames labelled with the same action), and thus it may be difficult to determine an action from a single flow image.

To add further temporal information to our training, it is possible to alter our approach to deal with sequences of images rather than a single image. In Chapter 4, we explored using recurrent neural networks for this purpose, with varying results (4.4.2) in line with findings by other authors [127, 19]. An alternative approach is to replace 2D operations (such as convolutional and pooling) with 3D analogues. The resulting increased dimensionality of features may make training more challenging. Distinct 3D filters would be needed to recognise various frequencies of spatio-temporal patterns associated with a single action, e.g. fast and slow 'chopping' actions in a kitchen activity. This leads to redundancy across 3D filters as each filter learns to recognise similar spatial patterns. This issue is further compounded when there are multiple, equally valid, ways of performing actions, which may be the case for fine-grained actions (e.g. using different kitchen utensils for a 'peeling' action).

To minimise such training issues, approaches have sought to compress the temporal dimension of spatio-temporal volumes to allow for well-studied 2D CNN architectures to be leveraged. Motion history images [335] were employed by Lea et al. [19] for

recognition of fine-grained actions based on observations of sparsity of optical flow information for fine-grained action datasets. A single-image compressed video representation, called a 'dynamic image' [140] was used as an input to a 2D CNN for action recognition. These dynamic images were used to encode spatio-temporal volumes of scene flow data to perform RGB-D gesture recognition [227]. These dynamic image approaches focus on single image types, however it may be beneficial to use multiple image types (such as colour and flow) for a compressed representation to allow identification of the objects in motion. For instance, distinct actions involving picking and placing different items in a kitchen activity may exhibit similar motion patterns (in the flow data), however the appearance of these items (in the colour data) would allow easier disambiguation of the actions. We modify our approach to train on compressed spatio-temporal representations involving multiple image types.

An inherent rank of spatio-temporal features is given by the temporal order of the features. Fernando et al. [139] used this property to extract a compressed representation of such features, called a rank pooled representation. Bilen et al. [140] extended this technique to work for raw video frames in a CNN context, which is how we shall employ this technique. This technique produces a ranking image that, when multiplied by a frame in a sequence, produces an estimate of its order in the sequence. As this image representation implicitly encodes the entire spatio-temporal volume, it can be used as a compressed representation of the volume. A method of approximating the image using a single gradient descent step was devised to allow this image to be used as part of a CNN [140]. This approximate method can be used as an intermediate layer in a CNN, referred to as a *Rank Pooling Layer*. The output images of such a layer are referred to as *Dynamic Images*.

Bilen et al. [140] showed that static images, like the inputs to the WeaveNet-LF, and dynamic images are complementary when combined for training. To increase temporal information available to our network, we intend to create a compressed representation of colour and scene flow frame sequences, and use this as an additional image type. This decision is based on identifying objects in colour data and associated movements of these objects from scene flow data. We modify WeaveNet-LF training by removing scene flow as an input image type, due to its sparsity and lower performance when training on this image type as found in Section 4.4.1.

In a recent work [336], Dynamic Images were generated from scene flow information to recognise human gestures. Recognition performance was improved when scene flow data was transformed (by a series of 1×1 convolutional kernels) prior to use in Dynamic Image generation. Using scene flow alone in such a way may not be ideal for our problem. The motion patterns of gestures are less temporally and spatially sparse than those in our fine-grained action problem, and this sparsity may result in a Dynamic Image with little informative detail. Furthermore, the state of the surrounding context is arguably of much greater utility for fine-grained fixed-setting actions than for ges-

tures. As such, we seek an approach that can both transform scene flow information prior to dynamic image generation, as well as include spatial information to allow the scene context to be used.

In this chapter, we have utilised the dense connectivity pattern to learn fused representations of multiple image types. Here, we wish to fuse scene flow and colour images in a form that can be used to generate a compressed spatio-temporal representation. The dense connectivity pattern is effective at combining images of differing complexity. Colour images are more complex than scene flow images, and hence, we use a Dense Layer here to combine these two image types. The Dense Layer transforms scene flow data prior to a rank pooling layer, as was shown to be effective for gesture recognition tasks [227]. In Figure 5.6, we show how we shall employ a Rank Pooling Layer as part of an early fusion stage. Each timestep of an action clip will require processing by a Dense Layer prior to the Rank Pooling Layer. To reduce the number of trainable parameters, the weights of the Dense Layer are shared across time. The Dense Layer can thus be viewed as a function for fusion of colour and flow, operating equivalently across time. The Rank Pooling Layer compresses the temporal extents of the input sequence of colour and scene flow image pairs to a single sample. This sample is a compressed representation of the changing context state from the colour images and the fine motion information from the scene flow images, which we refer to as a *Densely-Fused Action Image* (DFAI).

The DFAI representation is used in the larger WeaveNet-LF network by replacing the scene flow images with DFAI images as an input image type. By pre-pending the WeaveNet-LF with the DFAI generating operations, as shown in Figure 5.6, a new architecture is created which we refer to as WeaveNet-LF-DFAI. This network is trainable end-to-end, as Rank Pooling Layers can be used as an intermediate layer in a CNN. The main branch of the network now learns a complementary fusion [140] of the DFAI and single images of different types (colour, depth, etc.).

5.3 Implementation

5.3.1 Datasets

Dataset Selection

We evaluate our devised techniques on a collected dataset of OSCE performances, which we shall refer to as the *OSCE-V* dataset (Objective Structured Clinical Examinations - Venepuncture). The design of this dataset is fully described in Appendix I. In the dataset, 20 experimental subjects are recorded performing the task of venepuncture three times, resulting in 60 complete performances (example images shown in Figure 5.7). The dataset is labelled by an independent labeller with one of the 25 action labels shown in Table 5.1. The number of labelled frames is similar in scale to other datasets

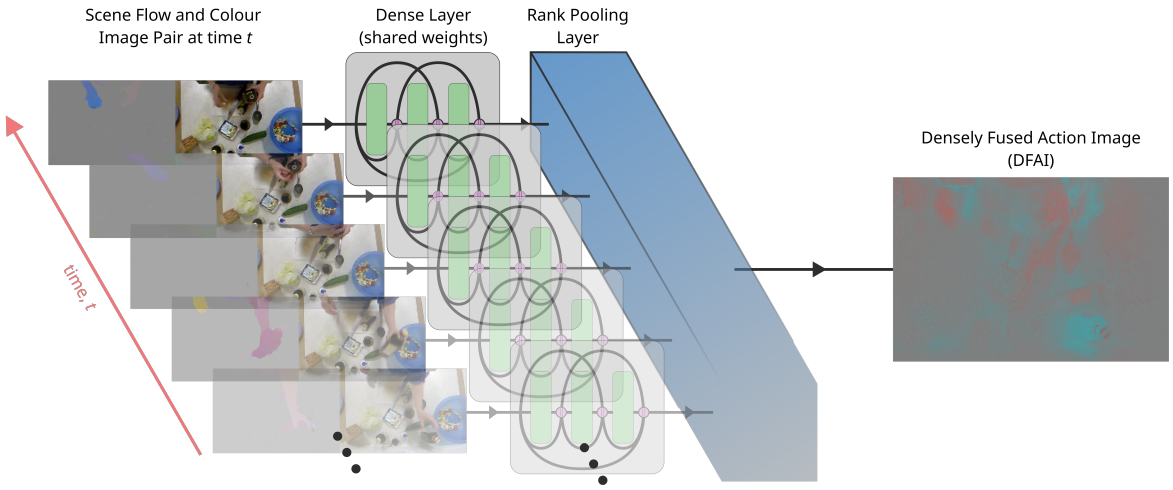


Figure 5.6: Densely-Fused Action Images are produced by a fusion of scene flow and colour image pairs using a Dense Layer. This Dense Layer shares weights across time, so that it is applied identically to each image pair in a sequence. The Rank Pooling Layer combines the fused outputs of the Dense Layer to produce the output Densely-Fused Action Image.

[232, 93]. For the case of OSCE-V, however, the presence of recordings from three viewpoints at a resolution of 640x480 in RGB-D makes this dataset significantly larger in terms of raw data compared to the 50 Salads dataset [232]. In our experiments, we train on the single central viewpoint camera for this dataset.

In Chapter 4, we used this 50 Salads dataset [232] to evaluate our devised techniques of feature fusion for fine-grained action recognition. This dataset is selected once again as a benchmark. This dataset is comprised of 50 performances of the task of preparing two types of salad using various kitchen implements in a fixed camera setting. Thus, as it is similar in scale to a single camera of the OSCE-V dataset, it can serve as a proxy for evaluating WeaveNet architecture variations. The dataset contains multiple levels of granularity: 'coarse'; 'mid'; 'eval'; and 'fine'. Previously, we evaluated our techniques on the 'eval' granularity which includes 10 actions. To demonstrate the applicability of our techniques to fine-grained action recognition, we shall also evaluate on the finer granularities, 'mid' and 'fine', which are composed of 18 and 52 actions, respectively. The difference between the 'mid' and 'eval' granularities is that the 'eval' granularity combines actions where the same utensil is used. The 'fine' granularity splits up the 'mid' action segments into three stages: 'pre'; 'core'; and 'post'. The 'pre' and 'post' phases "include grabbing, moving and placing utensils and ingredients" while the 'core' phase captures the essential actions [232].

Evaluation Method

To compare the techniques described in this chapter, we shall use the metrics: accuracy; precision; recall; and F_1. We use the approach explained in Section 3.4.4 of calculating these metrics over multiple splits and multiple actions, i.e. we take the unweighted mean of the metrics of each of the action classes.

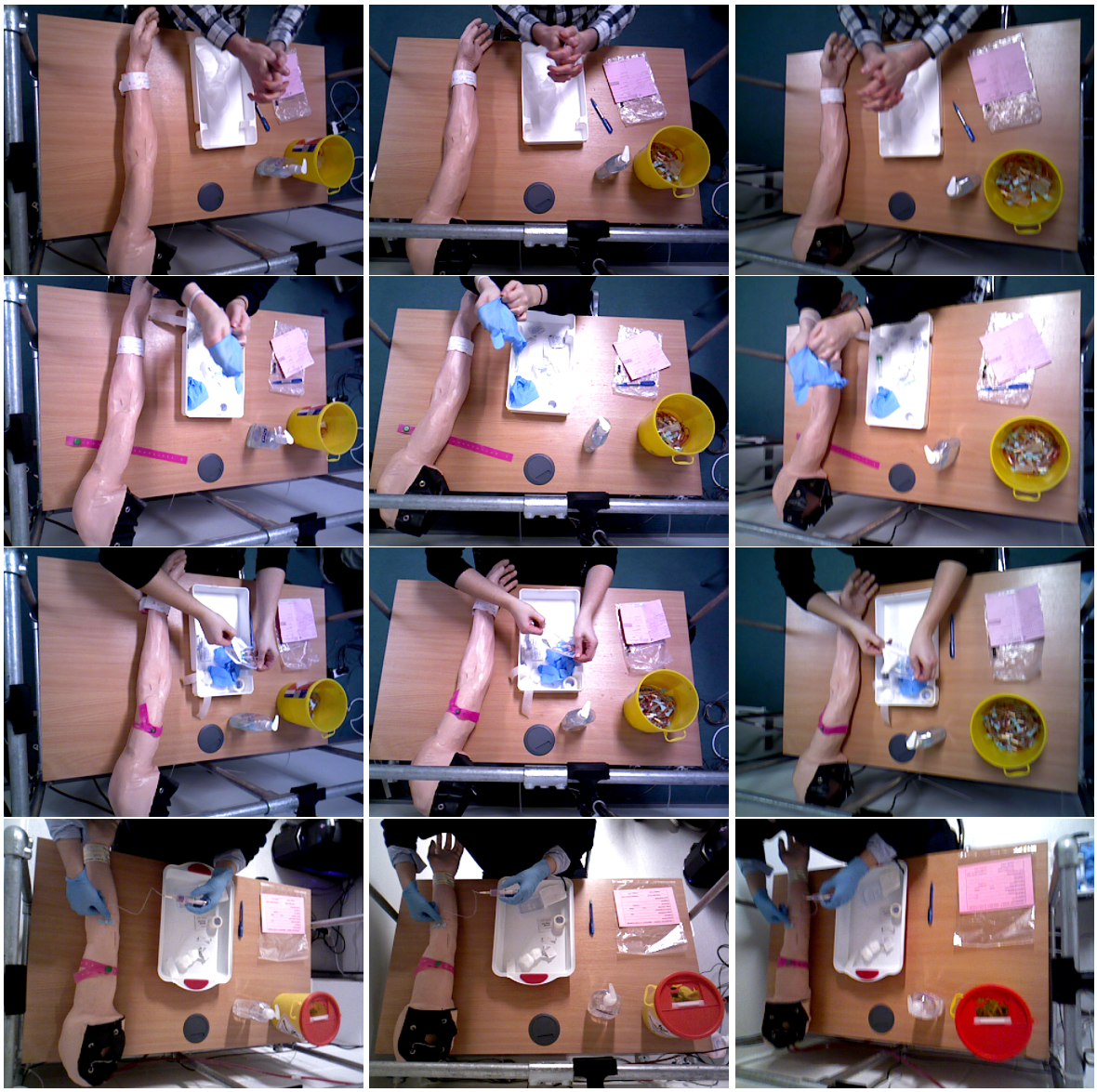


Figure 5.7: Sample frames recorded of performances of the OSCE task of venepuncture, with the left, centre, and right column representing the views from the corresponding camera view respectively. The actions from top to bottom row are as follows: wash hands, don gloves, prepare equipment, draw blood.

Action label	*No. Frames*
Background	103,244
Wash hands	51,145
Check ID	5,148
Clean procedure tray	10,933
Gather equipment	7,937
Move arm into position	4,290
Apply tourniquet	34,760
Identify vein	7,997
Put on gloves	36,037
Prepare equipment	65,390
Swab the site	16,208
Ready needle	38,335
Insert needle	18,886
Draw blood into bottle	39,289
Jostle bottle to mix	6,520
Open tourniquet	10,303
Apply cotton wool to site	12,306
Extract needle	11,688
Dispose of needle hazard	9,389
Tape cotton wool to site	16,701
Document sample bottle	11,172
Document sample questionnaire	18,018
Seal sample in bag	22,054
Dispose of waste	7,586
Remove gloves	6,947
Total	572,283
Total time	5:17:56

Table 5.1: The distribution of labels for the OSCE-V dataset. In total, over 5 hours of footage is collected for each of the three cameras. For comparison, the 50 Salads dataset [232] contains a total of 577,994 frames which is similarly sized to our total for a single camera. The 50 Salads dataset also contains 18 distinct mid level actions compared to 25 in our case.

As OSCE-V includes 60 performances across 20 subjects, to best characterise the generalisability of approaches, a cross-validation method should be used despite the consequent elongated training time (discussed in Appendix I.7). For the dataset, we propose five test folds, with each fold composed of 12 performances across four subjects.

Preprocessing

For training the WeaveNet architecture (Section 5.3.1), we must select the image types to use. In Chapter 4, we used colour, depth and scene flow image types. As one of the purported benefits of our WeaveNet architecture is that it effectively trains on multiple image types, we choose to additionally train on surface normals. Surface normals are defined as the vector perpendicular to the tangent plane of the surface at a point. The surface normal for a point can be calculated as the third eigen-vector of the covariance matrix formed from all points within a local neighbourhood. We use the Point Cloud Library [337] for this purpose and set the spherical local neighbourhood to have a radius of 1 centimetre. The direction of the resulting surface normals are fixed to point in the direction of the camera origin. To form an image, the normal vector components (x, y, z) are mapped to distinct colour channels (akin to red, green, blue) and encoded using 8 bits per channel. These surface normal images differ from depth in appearance in that edges and surfaces with high curvature are more conspicuous, while flat regions (such as tables) are relatively constant. We use the jet colour mapping to encode the depth information. This representation has been shown to improve results in the context of RGB-D object recognition [338]. This also ensures that depth is encoded using an equivalent number of channels to the other image types, a requirement for the groupwise convolution of Section 5.2.2. Finally, we map the vector components of the scene flow data to individual colour channels, similarly to surface normals. We encode the scene flow using 8 bits per channel, with the minimum and maximum values clamped at positive and negative five centimetres, based on inspection of the data. Encoding each image type using 8 bits per channel significantly improved our training times by reducing the data loading burden which was a bottleneck even when using fast solid state disks and performing data caching. We observed negligible deterioration in training accuracy as a result of using this reduced representation.

Normalisation of input image features has been shown to improve results for CNN training and as such we perform it here [35]. In Section 4.3.1, we calculated mean and variance for the colour channels for each split of the datasets used. We found during this work, however, that for colour images, centring on the midpoint of the channel range, and scaling by half of the range, gave similar results. This technique has been employed for training on ImageNet [69]. For depth and surface normal images, we use a single mean and variance value, calculated across the training splits, to normalise the data. For scene flow images, we use the channel value corresponding to the null vector (i.e. no movement) to centre the data, and we scale the data by half the data range.

As in Section 4.3.1, we do not use any data augmentation techniques based on the reasoning that the scene arrangements and lighting remain fixed throughout the datasets. Also similar to before (Section 4.3.1), we downsample all images to be 160×120 and take a manual cropping (of size 144×96 for 50 Salads and of size 145×100 for OSCE-V). This reduces the graphics memory cost associated with early layers in the network, as well as further reducing the data loading burden during training.

5.3.2 Classifier Training

In our evaluation, four models are trained for comparison: DenseNet; WeaveNet; WeaveNet-LF; and WeaveNet-LF-DFAI. The training of these models requires many hyperparameter choices. In Table 5.2, we list the hyperparameters used for the different network architectures. The growth rate, compression rate, and bottleneck size (explained in 5.2.1), are kept fixed throughout all experiments. These hyperparameters were found to work well within constrained graphics memory boundaries. In particular, overfitting was observed when the number of Dense Blocks (as per Figure 5.2) was increased. In the Strand Blocks (as shown in Figure 5.4), we use a smaller number of layers per block and a smaller growth rate than in the Dense Blocks based on similar observations.

For the WeaveNet-LF network, we use differing number of units in the linear layers of the main branch and strands branch, as well as differing dropout rates prior to these layers. This differing treatment is because the strands branch is thinner than the main branch. The number of units in these layers is kept small to limit the number of model parameters, otherwise overfitting was observed.

For the WeaveNet-LF-DFAI network, we also use different hyperparameters for the Dense Block which fuses the scene flow and colour information, as listed in the final column of Table 5.2. When training this network, we use a sequence length of 16 frames, which is found to work well under available graphics memory. To increase the length of the temporal window that the network can observe, we use a temporal stride of eight frames to generate these sequences, in line with other works tested on 50 Salads [19]. This gives a temporal window of over four seconds per sample for recordings of 30 frames per seconds (such as 50 Salads). For fair comparison across all experiments, we fix the batch size and the number of units in the initial convolution layer (or grouped convolution layer for WeaveNet architectures).

To train the networks, we minimise the cross-entropy loss (as discussed in Section 4.3.2). We apply weight decay, as listed in Table 5.2, for regularisation purposes. As the datasets we train on are significant in size, and there are lots of similar frames throughout, it was found that a small number of epochs is required for convergence. When training a full epoch at a high learning rate, overfitting was observed. To better monitor convergence, and allow more granular scheduling of the learning rates, we instead train on disjoint subsets of the training set for what we call “mini-epochs”.

Network	*DenseNet*	*WeaveNet*	*WeaveNet-LF*	*WeaveNet-LF-DFAI*
Batch size	40	40	40	40
Initial convolution units	36	36	36	36
Dense Block layers	(6, 6, 6)	(6, 6, 6)	(6, 6, 6)	(6, 6, 6)
Growth rate	12	12	12	12
Bottleneck size	4	4	4	4
Compression rate	0.5	0.5	0.5	0.5
Dropout	0.1	0.1	0.1	0.1
Weight decay	5.0e-4	5.0e-4	5.0e-4	5.0e-4
Strand Block layers	N/A	(3, 3)	(3, 3, 3)	(3, 3, 3)
Growth rate (strands)	N/A	6	6	6
Compression rate (strands)	N/A	0.5	0.5	0.5
Dropout (strands)	N/A	0.1	0.1	0.1
Dropout (linear - main)	N/A	N/A	0.1	0.1
Dropout (linear - strands)	N/A	N/A	0.2	0.2
Units (linear - main)	N/A	N/A	128	128
Units (linear - strands)	N/A	N/A	64	64
Dense Block layers (DFAI)	N/A	N/A	N/A	(6)
Growth rate (DFAI)	N/A	N/A	N/A	6
Bottleneck size (DFAI)	N/A	N/A	N/A	2
Dropout (DFAI)	N/A	N/A	N/A	0.1
Sequence length	N/A	N/A	N/A	16
Frame stride	N/A	N/A	N/A	8

Table 5.2: The hyperparameters selected for each of the tested architectures.

These subsets are constructed by sampling without replacement from the entire set of samples, with each sample having equal probability (i.e. uniformly). Each mini-epoch training set has cardinality of 0.02× that of the entire training set. This equates to 0.6 samples for each second of video on average (assuming 30 frames per second). When all disjoint subsets have been observed by the network, we generate new subsets.

When training the WeaveNet-LF-DFAI architecture, we train on sequences of images as opposed to single images. Training the architecture end-to-end, we found that the network quickly overfits. This is because the Dense Block used to generate the DFAIs observes a higher number of frames than the remainder of the network. To counteract this, we do not backpropagate through this block on every update. Instead we use an update frequency equivalent to the sequence length (i.e. we update this block's weights once every 16 forward passes). This update frequency ensures that each layer in the network is trained based on observation of an equivalent number of frames, and was found to be effective at eliminating the observed overfitting.

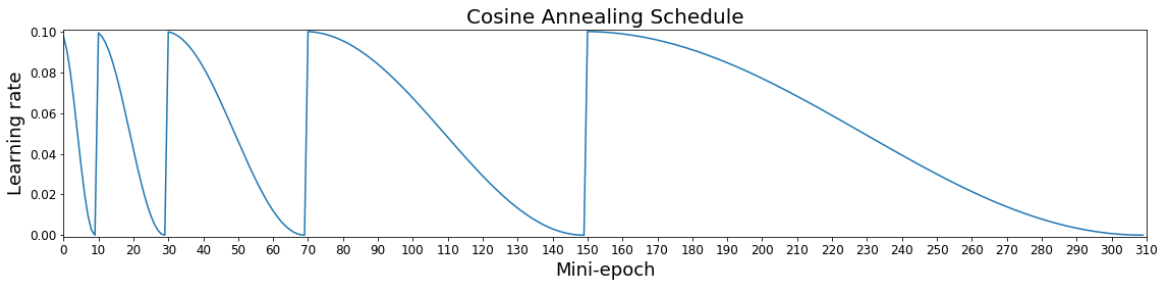


Figure 5.8: The learning rate schedule following the cosine annealing method. We take model snapshots at mini-epochs 70, 150 and 310.

For effective training, it is necessary to select an appropriate optimisation method. Previously (in Section 4.3.2), we used the Stochastic Gradient Descent (SGD) optimiser along with a carefully managed learning rate schedule. We again use the SGD optimiser given its use in training the DenseNet architecture [22]. We also use Nesterov momentum with a momentum value of 0.9 [32]. In contrast to before, we employ a different learning rate scheduling strategy. Utilising a schedule where the learning rate for each epoch is determined by a cosine function can achieve empirically better results [339]. This cosine function produces an annealing schedule where a high learning rate is used for a long period, with a sharp decay in learning rate towards the end of the schedule. The key difference with this method is that once this schedule is completed it restarts at the original learning rate and repeats the schedule again. A lower frequency cosine function can be used to produce a longer schedule in these subsequent schedules. This process of "warm restarts" can disrupt convergence in local minima and cause the network weights to diverge from their previous values, potentially finding more optimal values. An advantage of this method is that it is possible to take snapshots of the model weights at these convergence points to produce a set of models on completion of training. As the warm restarts cause the model weights to diverge, these model snapshots are distinctive enough to allow ensembles to be produced with them. Based on experimentation, we use values of $\eta_{max} = 0.1$, $\eta_{min} = 10^{-5}$, $T_0 = 10$, and $T_{mult} = 2$, as defined in the original work [339], producing a final schedule shown in Figure 5.8. To evaluate our models, we inspect the results of the final model (after 310 mini-epochs) and compare it to those produced by taking an ensemble of the model snapshots at mini-epochs 70, 150 and 310.

The networks are trained using the PyTorch deep learning framework. This framework was used for its expressibility and object-oriented design, which facilitated easier re-use of network components. All networks are trained on an Nvidia Titan V card.

5.4 Evaluation

5.4.1 Model Ablation

We summarise the quantitative improvements of the WeaveNet architecture, and variants, over the baseline DenseNet model in Table 5.3. Models were trained on the 'eval' granularity of the 50 Salads dataset, which has 10 action classes. The WeaveNet model improves F_1 score by more than 3% over the DenseNet model when trained on all image types (colour, depth, scene flow, normals). The WeaveNet-LF and WeaveNet-LF-DFAI models each improve performance further across all metrics. We also note that the cosine annealing ensembling approach effectively improves classification performance in each case. A parameter study for the WeaveNet-LF architecture is included in Appendix J.

Model Type	*Accuracy*	*Precision*	*Recall*	F_1
DenseNet	77.73	73.70	69.90	70.80
DenseNet (ensemble)	79.28	75.15	71.82	72.34
WeaveNet	81.07	76.37	73.36	73.81
WeaveNet (ensemble)	82.37	78.97	75.58	76.20
WeaveNet-LF	82.02	77.98	74.74	75.46
WeaveNet-LF (ensemble)	83.96	79.50	77.52	77.34
WeaveNet-LF-DFAI	82.73	79.54	76.55	77.38
WeaveNet-LF-DFAI (ensemble)	**84.51**	**81.86**	**79.08**	**79.76**

Table 5.3: Classification results for the different model types described in this chapter across the five splits of the 50 Salads dataset for the 'eval' granularity, as percentages. In each case, the models are trained on all image types. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported.

5.4.2 Granularity Evaluation

The 50 Salads dataset contains finer granularities, 'mid' and 'fine', than the 'eval' granularity used for previous evaluation. The 'mid' granularity differs from the 'eval' granularity in that actions using the same utensil have not been combined into a single class, resulting in 18 actions classes. The 'fine' granularity splits each 'mid' labelling into a 'pre', 'core' and 'post' phase, resulting in 52 action classes. To demonstrate the feasibility of recognition of fine-grained actions using our architecture, we evaluate the WeaveNet-LF-DFAI on these granularities.

In Table 5.4, we see that model performance falls as the granularity becomes finer. This is expected as each action class has a decreased number of associated labelled frames. Furthermore, the increased number of classes requires an increase in complexity to be modelled.

Granularity	*Classes*	*Accuracy*	*Precision*	*Recall*	F_1
'Eval'	10	82.73	79.54	76.55	77.38
'Eval' (ensemble)	10	84.51	81.86	79.08	79.76
'Mid'	18	75.58	71.87	66.67	68.39
'Mid' (ensemble)	18	78.96	76.53	70.22	72.27
'Fine'	52	67.49	55.57	50.47	52.19
'Fine' (ensemble)	52	70.04	59.07	52.47	54.63

Table 5.4: Classification results for DenseFuseNetLF-DFAI model for different levels of granularities across the five splits of the 50 Salads dataset, as percentages. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported.

5.4.3 Comparison to Published Works

To understand the performance of our WeaveNet-LF-DFAI architecture (the highest performing architecture in Section 5.4.1) for fine-grained action recognition, we compare to other published works. In Table 5.5, we see that our approach outperforms these approaches for all granularities. We point out that the Spatial CNN and Spatio-Temporal CNN [19] are of closest comparison in terms of purpose due to their focus on action recognition rather than action segmentation. On very fine-grained actions, we outperform this approach significantly, **improving accuracy from 44.1% to 67.5% for the 'fine' granularity**. This confirms that our approach is well-suited for fine-grained recognition. On more coarse-grained actions, we also outperform other works, **improving accuracy from 73.4% to 82.7% for the 'eval' granularity**.

The Encoder-Decoder Temporal Convolutional Network (ED-TCN) [20] uses the outputs of the Spatio-Temporal CNN to perform a distinct action segmentation stage. Such a secondary segmentation stage could similarly be applied to the outputs of the WeaveNet-LF-DFAI model, and may improve classification accuracy further. The comparison with Coupled Generative Adversarial Networks [340] is particularly relevant, as this approach also attempts to combine multiple image types to perform action segmentation. Given that WeaveNet-LF-DFAI outperforms these approaches in terms of accuracy, it could form the basis of future segmental approaches.

It may be possible to use architectures such as I3D [17], SlowFast [141] or techniques such as Temporal Relation Networks [144], to determine action recognition performance using fewer image modalities than used here. The WeaveNet architecture required careful consideration of architectural parameters to prevent overfitting. In particular, the number of Dense Blocks was reduced for this reason, as discussed in Section 5.3.2. The large-scale models of I3D and SlowFast may also require such consideration when attempting to classify these datasets. Such explorations of the performance of these models should be explored as part of future work.

50 Salads ('eval')	*Accuracy*
Spatial CNN [19]	68.0
Spatio-Temporal CNN [19]	71.3
Encoder-Decoder Temporal Convolutional Network [20]	73.4
WeaveNet-LF-DFAI	**82.7**
50 Salads ('mid')	*Accuracy*
Spatial CNN [19]	54.9
Spatio-Temporal CNN [19]	59.4
Improved Dense Trajectories + Language Model [239]	48.7
Encoder-Decoder Temporal Convolutional Network [20]	64.7
Temporal Deformable Residual Networks [341]	68.1
Coupled Generative Adversarial Networks [340]	74.5
WeaveNet-LF-DFAI	**75.6**
50 Salads ('fine')	*Accuracy*
Spatio-Temporal CNN [19]	44.1
WeaveNet-LF-DFAI	**67.5**

Table 5.5: Comparison with state of the art techniques that perform cross-validation on the 50 Salads dataset. We report the frame-wise accuracy over all five splits of the dataset.

5.4.4 Computational Analysis

In Table 5.6, we report the results of a computational profiling of tested architectures. We see that the computations (i.e. floating point operations, FLOPs) increase only modestly with the WeaveNet architectures. The WeaveNet-LF-DFAI model has increased computations due to the processing of the colour and scene flow images for each frame in a sequence. We note that the WeaveNet architectures introduce additional parameters; however, overall the model size remains relatively small. For reference, the SqueezeNet architecture [342], which was specifically designed for parameter efficiency, has 1.2 million parameters. Had we used separate branches for each image type, as performed in previous works [137, 16], the computations and number of parameters would have increased 100% with each additional image type. Based on the recognition improvements we achieve from the additional image types, our model scales efficiently to process multiple image types.

5.4.5 WeaveNet Fusion Analysis

To quantify the benefits of the fusion method introduced by WeaveNet, the performance of the baseline DenseNet architecture must be established for the various combinations of image types used for classification. We train DenseNet on individual image types to reveal the relative saliency of single images of each type for action recognition purposes. We evaluate the performance across the first test split of the 50 Salads dataset.

Model Type	*FLOPs* ($\times 10^9$)	*Parameters* ($\times 10^3$)
DenseNet (colour only)	1.13	184.8
DenseNet	1.17	187.7
WeaveNet	1.99	366.2
WeaveNet-LF	2.04	477.0
WeaveNet-LF-DFAI	3.53	483.0

Table 5.6: The profiling results for each of the tested model types. FLOPs refers to the floating point operations required for a forward propagation of a single image sample of size used in the experiments. Parameters refer to the number of model parameters for the tested arrangements. Apart from DenseNet (colour only), input samples include all image types.

Image Type	*Accuracy*	*Precision*	*Recall*	F_1
Colour	83.88	**84.00**	69.67	71.30
Depth	78.05	72.47	67.69	68.54
Scene Flow	59.26	47.31	41.79	41.30
Normals	75.74	67.14	62.14	62.22
DenseNet	**84.83**	83.59	**75.70**	**77.21**

Table 5.7: Classification results as percentages for the different data types for a single test split of the 50 Salads dataset using the 'eval' granularity, as percentages. The reported results are from the cosine annealing ensemble of three networks. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported. 'DenseNet' image type refers to the combination of all image types used as a concatenated input to the DenseNet network.

In results for the base DenseNet architecture, shown in Table 5.7, we observe that the network performs best on the colour image type compared to the other individual image types. We note that the scene flow image type performs poorest. This reflects our arguments that scene flow information is sparse, temporally and spatially, for fine-grained action recognition in a fixed setting. We note that for recall and the F_1 metric, the combined type significantly outperforms the colour image type. This indicates that the inclusion of other image types allows disambiguation of the action classes that are difficult to classify with colour information alone. However, in terms of accuracy and precision, the colour image type achieves performance close to the combination of all image types. This indicates that action classes that are overrepresented in the dataset are more accurately classified using colour information alone. This may be due to the network dedicating more parameters to recognition of salient features of these classes when trained on only colour information.

We compare recognition results when using the WeaveNet architecture to perform fusion. The effect of additional image types is examined by evaluating the performance across the first test split of the 50 Salads dataset. We chose to examine the performance when: depth is used in addition to colour; depth and scene flow are used in addition to colour; and depth, scene flow and surface normals are used in used in addition to colour. These combinations were chosen as colour, depth and scene flow represent the

most distinct image types that we train on, while surface normals, being derived from depth, are less distinct.

Image types	*Accuracy*	*Precision*	*Recall*	F_1
Colour	81.66	77.33	68.30	70.14
Colour, Depth	86.27	84.87	77.04	78.80
Colour, Depth, Scene Flow	86.07	**85.32**	76.77	78.99
Colour, Depth, Scene Flow, Normals	**86.61**	84.39	**77.44**	**79.08**

Table 5.8: Classification results as percentages for the WeaveNet architecture trained on different combinations of image types for a single test split of the 50 Salads dataset. The reported results are from the cosine annealing ensemble of three networks. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported.

In Table 5.8, we see that each additional image type improves performance for the balanced F_1 metric. We note that there is a significant improvement from the addition of depth information. Scene flow and normals information increase F_1 performance somewhat, but to a much smaller extent. Previously, the DenseNet architecture showed worse precision when trained on image types in addition to colour as opposed to colour alone. We observe that the WeaveNet architecture improves all metrics, including precision, when trained on all image types. When trained on all image types, the WeaveNet architecture outperforms the DenseNet architecture (listed as DenseNet in Table 5.7). This indicates that the WeaveNet architecture is performing fusion of the image types more effectively. We note, however, that the DenseNet architecture outperforms the WeaveNet architecture for the colour image type. This validates the effect of the architectural differences in the WeaveNet architecture compared to the DenseNet architecture. As the DenseNet architecture was designed and optimised for recognition from a single image type, it performs well in these scenarios. The WeaveNet architecture, designed for recognition from multiple image types, performs better under scenarios of multiple image types.

5.4.6 Densely-Fused Action Images Analysis

The WeaveNet architecture achieves high levels of action recognition accuracy from single frames, indicating the presence of significant salient information in single frames of the 50 Salads dataset. We investigate the effectiveness of the devised DFAI method of combining temporal information with these single frames. To do so, the appearance of generated DFAI images is inspected.

In Figure 5.9, we visualise the DFAI output channels for the action of 'peel cucumber'. In this representation, we observe that motion patterns of the subject's hands are captured. We note that of the three channels in the output, two are almost identical. As the Dense Block used to encode the colour and flow images is shallow, it learns a simple representation from these image types. This representation appears to highlight

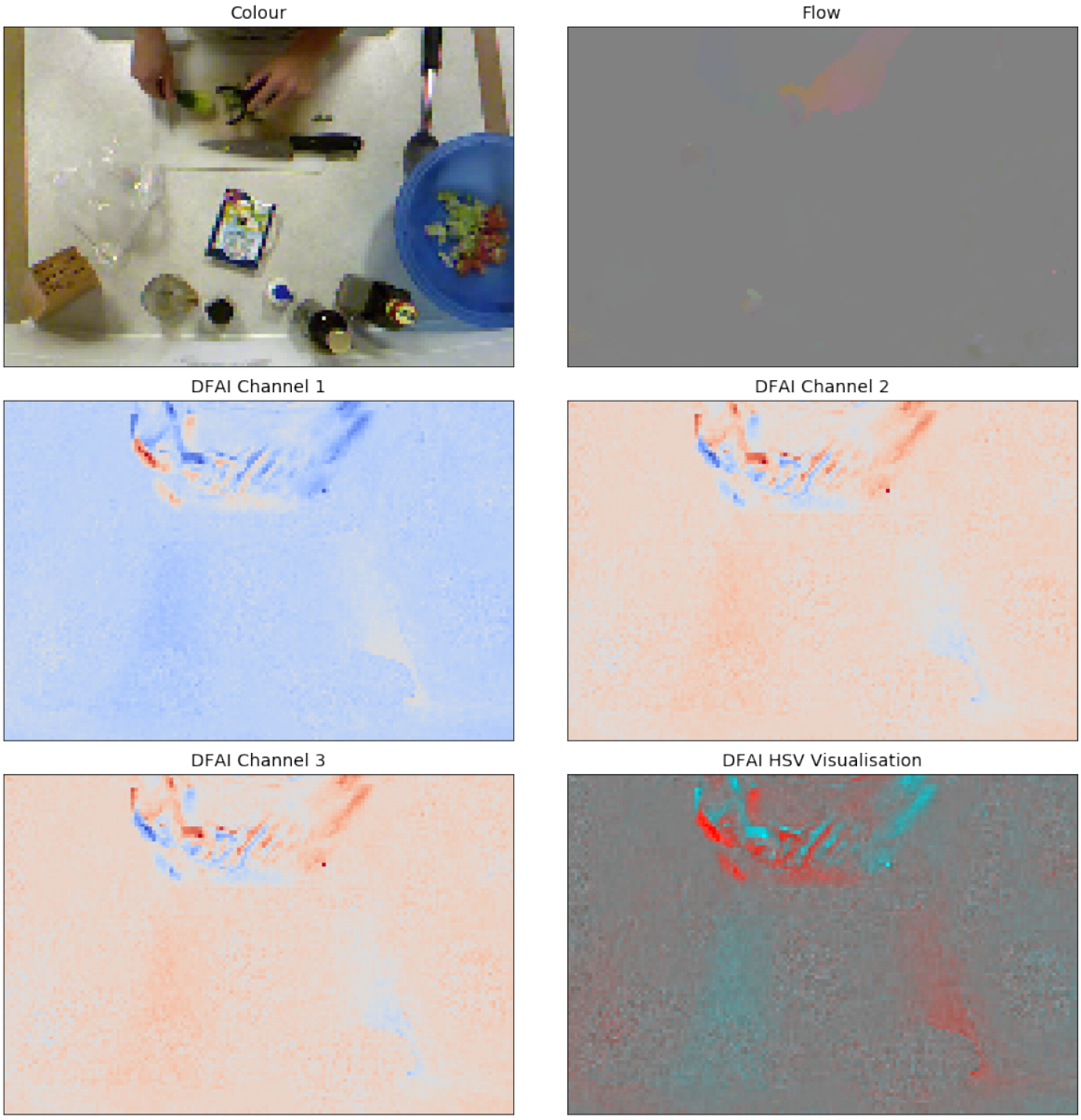


Figure 5.9: The output DFAI for a frame from the 50 Salads dataset with the 'peel cucumber' action. The DFAI HSV visualisation encodes the length of the 2D vector formed by DFAI Channel 1 and DFAI Channel 2 as the value channel, the arctangent of this vector as the hue channel and the DFAI Channel 3 as the saturation channel. The DFAI representation captures a pattern indicating movement, as the subject's hand moves laterally when they peel the cucumber.

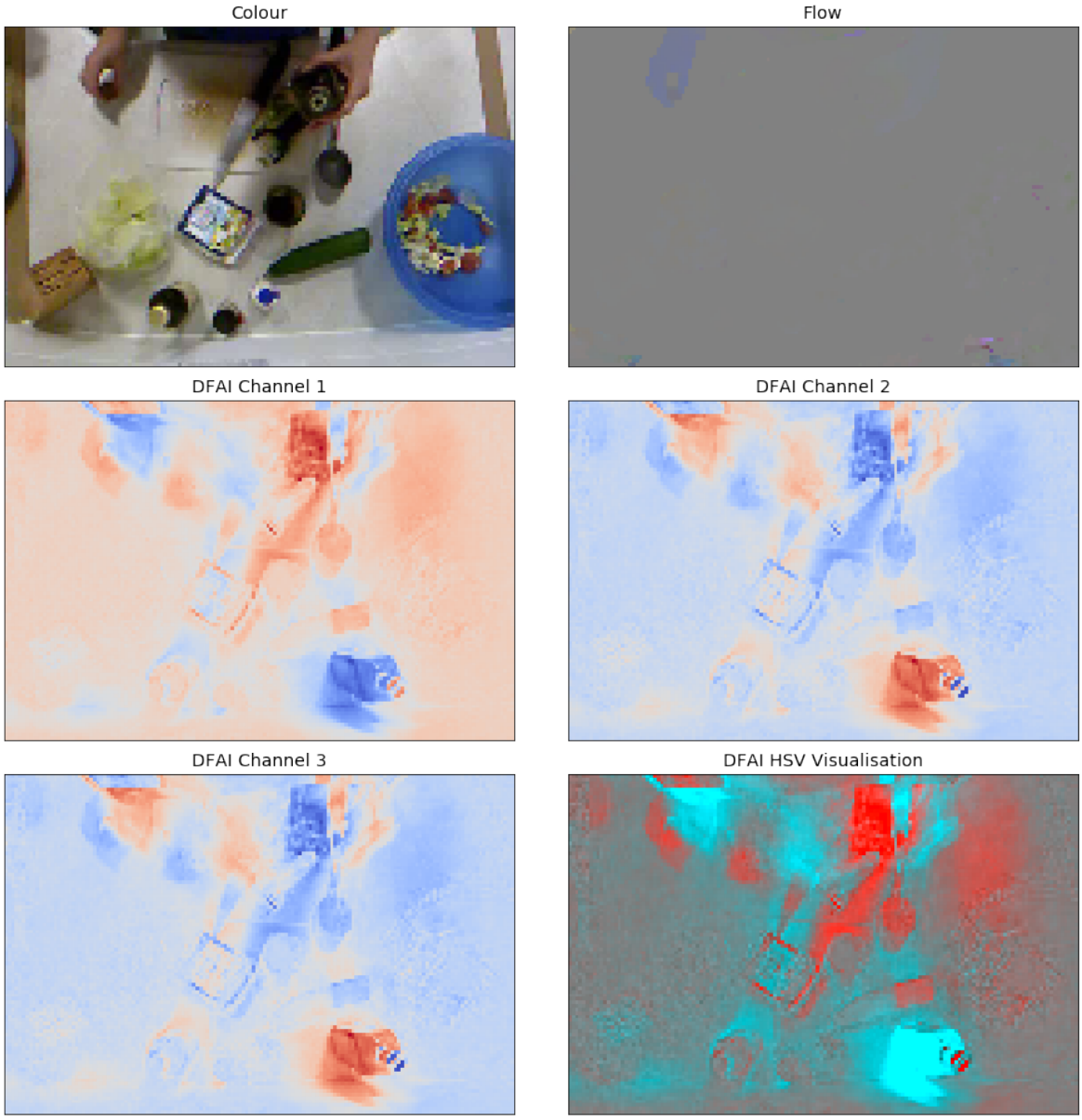


Figure 5.10: The output DFAI for a frame from the 50 Salads dataset with the 'add oil' action. The DFAI representation used is as per Figure 5.9. In the visualisation, the DFAI has captured the previous location of the oil bottle in the representation, indicating that it has been moved from this location. It shows that the DFAI has captured the changing contextual information, which may aid recognition.

objects and motions by firing when there is either positive or negative deviation from a background value. When combined over time using a Rank Pooling Layer, the DFAI representation highlights the motions of the subject and objects. In Figure 5.10, we visualise the DFAI representation for the action of 'add oil', and note that it effectively highlights the changed locations of the oil bottle. Thus, this DFAI representation can be effective at capturing changing context, as well as motion. For the scenario of fine-grained action recognition, in a fixed context, contextual changes are informative. In this regard, this DFAI representation bears similarity to difference images or motion history images.

To determine the quantitative effect of the inclusion of this representation, we evaluate the WeaveNet-LF-DFAI architecture over all five splits of the 50 Salads dataset on the 'eval' granularity. We examine the effects of inclusion of the DFAI representation, in place of scene flow information, by comparing recognition performance against WeaveNet-LF trained on all image types.

Action class	*Precision*	*Recall*	*F_1 score*	*Support*
Add dressing	86.5 (+6.4)	73.7 (+7.7)	79.6 (+7.2)	52,675
Add oil	77.8 (+9.5)	86.4 (+2.7)	81.9 (+6.6)	25,990
Add pepper	85.0 (+5.6)	83.7 (+1.6)	84.3 (+3.6)	11,866
Background	72.8 (-2.3)	87.9 (+1.1)	79.6 (-0.9)	105,252
Cut into pieces	91.1 (+0.2)	91.5 (-1.5)	91.3 (-0.6)	201,994
Mix dressing	72.0 (-6.7)	49.7 (+9.4)	58.8 (+5.5)	18,416
Mix ingredients	63.5 (-0.2)	62.1 (-2.1)	62.8 (-1.2)	21,119
Peel cucumber	89.6 (+3.6)	84.8 (+2.4)	87.1 (+2.9)	59,286
Place into bowl	84.0 (+7.0)	58.0 (-4.4)	68.6 (-0.3)	48,024
Serve salad onto plate	73.0 (-7.5)	87.7 (+1.2)	79.6 (-3.7)	33,283
Class mean	79.5 (+1.6)	76.5 (+1.8)	77.4 (+1.9)	

Table 5.9: The classification metrics calculated for each action class across all five splits, for the WeaveNet-LF-DFAI model trained for 310 mini-epochs. Support refers to the number of instances of the action class in the dataset. All figures, apart from support, are reported as percentages. In parenthesis, we report the performance increase or decrease compared to the equivalent metric for the WeaveNet-LF model (best viewed in colour).

The WeaveNet-LF-DFAI model F_1 over the five test splits of 50 Salads is 77.4%, an improvement of 1.9% over the WeaveNet-LF model. Inspecting the results in Table 5.9, we note large improvements in metrics for certain actions ('add dressing', 'add oil') and disimprovements for other actions ('serve salad onto plate', 'mix ingredients'). These varying results could be explained by the effect of the chosen temporal window. As some actions involve broad movements over longer periods (e.g. picking up the oil bottle for 'add oil'), the large temporal window can capture the changing location of objects (as shown in Figure 5.10) to facilitate recognition. Conversely, when the length of an action is short or where the motion is brief (such as in 'serve salad onto plate'), the temporal window may contain a significant proportion of frames that relate to previous actions, making recognition more challenging. To examine this situation further, we

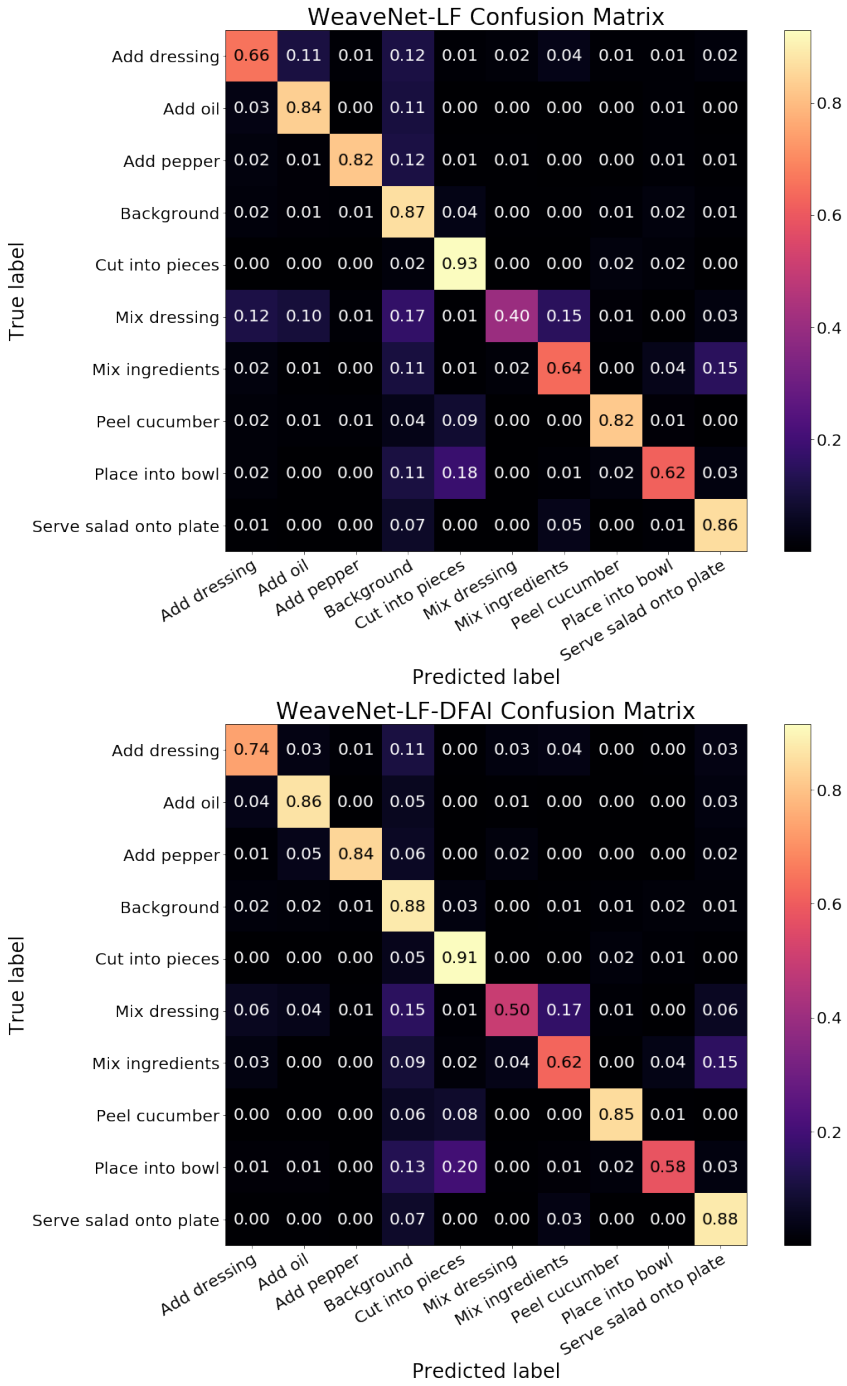


Figure 5.11: Confusion matrices for the WeaveNet-LF and WeaveNet-LF-DFAI models across all five splits of the 50 Salads dataset for the 'eval' granularity. The false negatives of the WeaveNet-LF-DFAI model are concentrated on fewer classes than those of the WeaveNet-LF model, particularly for the 'mix dressing' action class.

inspect the confusion matrices for the two models.

In the confusion matrices shown in Figure 5.11, we note that the WeaveNet-LF-DFAI model predicts the action 'serve salad onto plate' incorrectly as the actions 'add oil', 'add dressing' and 'mix dressing' to a greater extent than the WeaveNet-LF model. These actions precede 'serve salad onto plate' in a certain proportion of the 50 Salads task orderings. This observation corresponds with the explanation of the effect of the temporal window size. We note that the 'mix dressing' action is predicted less as 'add dressing' or 'add oil', indicating that the DFAI representation (with the chosen temporal window) helps to disambiguate the similar scene contexts of these actions. Similar observations are made for the action 'add dressing' which is predicted less as 'add oil'. Based on these observations, our choice of temporal window size, which was in line with previous work [19], has trade-offs in terms of model performance for different actions. Future research may be required to identify methods of using an adaptive window size to improve performance across all action classes.

5.4.7 OSCE-V Performance

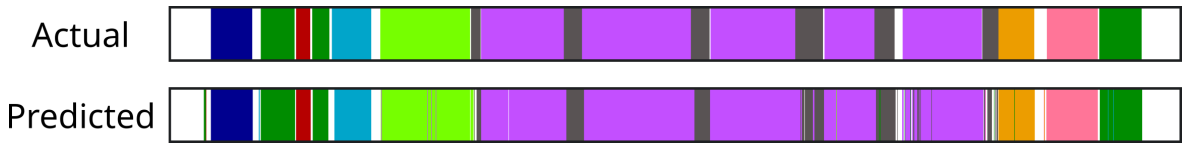
To validate OSCE-V dataset as a benchmark for fine-grained action recognition, we train the WeaveNet-LF-DFAI model for the central viewpoint camera only. The model was trained on a combination of colour, depth and DFAI images. The procedure used for training was identical to that used for 50 Salads. The recognition results, across all splits, are presented in Table 5.10. An accuracy of 72.1%, for a single model, on this OSCE-V dataset is in line with our accuracy on the 'mid' granularity of 50 Salads of 75.6%. These two scores indicate that our WeaveNet-LF-DFAI approach generalises well to other datasets, specifically including OSCE-V, given the similar number of action classes (25 for OSCE-V, 18 for 50 Salads 'mid' granularity). An accuracy level of 72.1% indicates that there is potential for performance improvements, possibly with the inclusion of multi-camera information.

Model	*Accuracy*	*Precision*	*Recall*	F_1
WeaveNet-LF-DFAI	75.07	75.16	70.24	72.09
WeaveNet-LF-DFAI (ensemble)	77.22	77.99	72.84	74.83

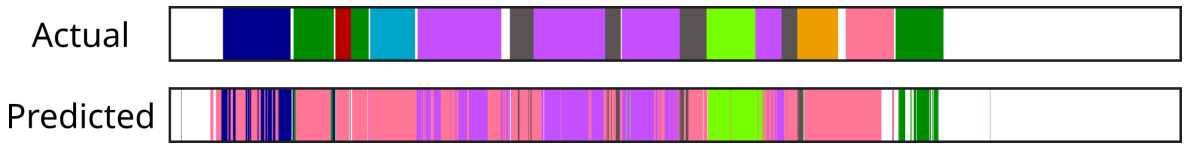
Table 5.10: Classification results for the WeaveNet-LF-DFAI model across all splits of the OSCE-V dataset, as percentages. Training was performed across all image types. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the 25 classes, and the unweighted mean across classes is reported.

5.4.8 Qualitative Evaluation

To better understand how well the WeaveNet-LF-DFAI model performs, we compare model predictions against ground truth labels for entire video sequences.



(a) 50 Salads, 'eval' granularity, 93.9% accuracy.



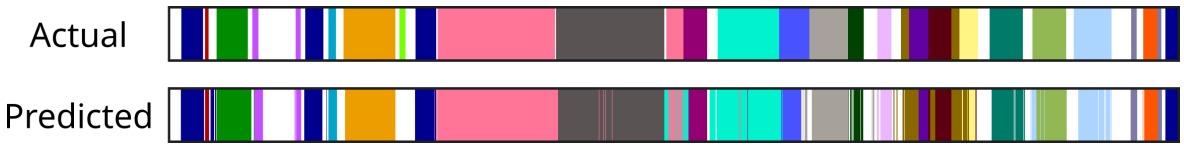
(b) 50 Salads, 'eval' granularity, 61.1% accuracy.

Figure 5.12: Visualisations of ground truth labels against WeaveNet-LF-DFAI predictions for single videos, with each colour representing a distinct action class. Visualisations of the videos with highest and lowest accuracy for 50 Salads 'eval' granularity are shown in (a) and (b), respectively.

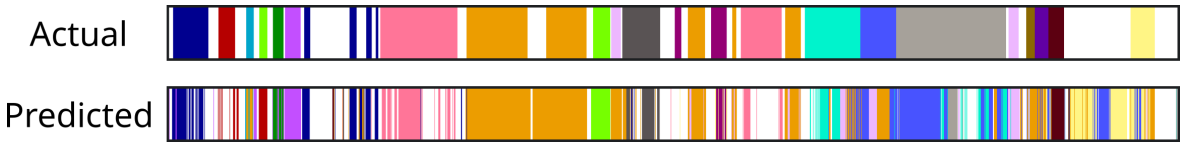
In Figure 5.12 (a), the predicted 'eval' granularity labels for the 50 salads video with the highest classification accuracy are shown. These predictions closely match the ground truth labels, however, there are sporadic errors throughout the sequence. Errors are observable at the beginning of action segments, in particular the dark gray segments, corresponding to 'place into bowl' actions. The WeaveNet-LF-DFAI model has difficulty recognising the beginning of these action segments due to the visual similarity to the preceding actions. This is in line with the confusion matrices of Figure 5.11) where the 'place into bowl' action is shown to be confused with 'cut into pieces'. Furthermore, the lower recall of WeaveNet-LF-DFAI for 'place into bowl.' compared to WeaveNet-LF (as per Table 5.9) shows that the addition of preceding frames as inputs compounds this confusion with the preceding action.

In Figure 5.12 (b), the predicted 'eval' granularity labels for the 50 salads video with the lowest classification accuracy are shown. In this case, predictions are strongly biased toward a single action, 'serve salad onto plate'. For much of this video sequence, differently from other 50 salads video sequences, the salad serving plate is centrally located in the task area. This shows that the model has learned that the presence of the plate indicates the 'serve salad onto plate' action. Although this is sensible, it is plausible, as in this instance, that the presence of the plate is not relevant to the current action being performed. One method of improving model performance to deal with such issues would be to collect more data, including more instances of such realistic scenarios.

In Figure 5.13 (a), the predicted labels for the OSCE-V video with the highest classification accuracy are shown. Sporadic errors, similar to Figure 5.12 (a), are observable. In the latter half of the sequence, there is confusion between the olive, violet, maroon and yellow coloured segments, corresponding to 'apply cotton wool', 'extract needle', 'dispose of needle' and 'tape cotton wool' actions, respectively. The similar contextual appearance of these actions makes recognition challenging, however



(a) OSCE-V, 87.9% accuracy.



(b) OSCE-V, 53.3% accuracy.

Figure 5.13: Visualisations of ground truth labels against WeaveNet-LF-DFAI predictions for single videos, with each colour representing a distinct action class. Visualisations of the videos with highest and lowest accuracy for OSCE-V are shown in (a) and (b), respectively.

the model has managed to correctly predict a subset of frames within each of the ground truth segments.

In Figure 5.13 (b), the predicted labels for the OSCE-V video with the lowest classification accuracy are shown. Despite 53% classification accuracy, many actions are incorrectly predicted for their entire duration. Some of the errors can be explained by the fact that there are portions of this video sequence in which the action being performed is only partially visible. For example, the participant moves in and out of shot while washing their hands. Other errors may be as a result of the participant performing actions in a different manner to other participants. For example, during the 'draw blood' action (light gray segment in Figure 5.13 (b)) the participant adjusts the needle causing confusion with the 'insert needle' action (in blue). Recognition of such action variations are thus more challenging for the WeaveNet-LF-DFAI model without access to further temporal information.

Classification performance for challenging sequences, and indeed for each of the previously discussed sequences, could be improved with the use of a segmental model in addition to WeaveNet-LF-DFAI. Such a model could be trained via supervised learning on predicted sequences of the WeaveNet-LF-DFAI model. The availability of temporal contextual clues could help eliminate the observed sporadic errors. In the sequence of Figure 5.12 (b), the temporal information available would reduce the bias towards the 'serve salad onto plate' action which typically occurs towards the end of a sequence. Finally, data augmentation of sequences (e.g. temporal jittering) during training may improve segmental model performance on challenging sequences such as that of Figure 5.13 (b).

5.5 Discussion

In this work, we have devised a novel CNN architecture, WeaveNet, that performs action recognition via fusion of multiple image types. We have detailed the decisions be-

hind the architecture and validated these choices empirically on the benchmark dataset of 50 Salads. We devised a novel approach of including temporality as part of the network in an efficient manner, through combining densely-fused scene flow and colour information across time to create Densely-Fused Action Images. Experimentally, we show that each architecture variant is efficient in terms of computations and parameter use. Our approach outperforms published works on three 50 Salads label granularities. Specifically, on the 'fine' granularity, we improve accuracy from 44.1% to 67.5%, and on the 'eval' granularity, we improve accuracy from 73.4% to 82.7%. These results demonstrate the effectiveness of our WeaveNet architectures at learning to recognise fine-grained actions from multiple image types.

The goal of this thesis is to recognise the fine-grained actions involved in goal-directed activities, such as an OSCE. In this chapter, we detailed the collection and arrangement of a new dataset of performances of the OSCE of venepuncture. We showed that the WeaveNet architecture is capable of recognising the fine-grained actions with which this dataset is labelled. As such, we have demonstrated a fine-grained action recognition method that could be used in an online manner as part of a pervasive computing application for OSCEs.

This work presents multiple avenues for future research. The inclusion of multi-camera recordings as part of the dataset allows for exploration of methods that learn from multiple viewpoints. One possibility is to use the data to construct volumetric representations and use these representations as inputs to a classification scheme. The dataset presents opportunities for further labelling, such as subjective assessments of the 'quality' or 'skill level' of actions. Such a labelling could form the basis of a ranking method to determine perceived skill. Our recognition approach, WeaveNet, works in an online manner; however, by combining it with an action segmentation approach, a breakdown of an OSCE performance by steps could be produced. This breakdown could be used to determine that the correct steps were taken during an OSCE, and to calculate other information such as the timings of actions. This information can be used in an OSCE assessment or for debriefing the student for training purposes.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Our research intended to address the problem of how best to **recognise the fine-grained actions of a fixed-setting goal-directed activity, such as an OSCE, from video data**. In reviewing research related to this problem, we identified particular gaps related to this research goal. Further research was found to be warranted into methods of using multiple image types for the problem of fixed-setting fine-grained action recognition. For this problem, we also noted that further investigation was required into the use of temporal information (i.e. motion data and sequence data). It was also noted that a dataset of fine-grained actions performed as part of a goal-directed task in a fixed setting that includes depth and colour information from multiple viewpoints did not exist. The contributions of this thesis tackle these problems. A system was devised to capture human-object interaction tasks involving fine-grained actions from multiple viewpoints with RGB-D cameras. Techniques of using multiple image types to recognise fine-grained actions in a fixed setting were also devised. Along with these techniques, methods were contributed of incorporating spatio-temporal information into the action recognition process, improving recognition performance. A dataset was collected of performances of the OSCE of venepuncture, including 60 performances, totalling over 15 hours of footage from three cameras. This dataset is a valuable contribution for future research into pervasive computing applications in this domain. Cumulatively, these contributions directly tackle the motivating problem of recognising the fine-grained actions of a fixed-setting goal-directed activity, such as an OSCE.

In Chapter 3, a system was designed, implemented and evaluated for the recording of table-based human-object interaction tasks from multiple viewpoints. A pipeline was devised for the tracking of 3D poses of arms and objects from point cloud data received from multiple cameras. A method was presented of using grid search to identify a neural network architecture, involving Long Short Term Memory cells (LSTM), that maximises action recognition from pose estimates. A multi-camera RGB-D dataset

of performances of the task of preparing a cup of tea was collected, and the devised recognition approach was evaluated on this dataset. A code framework is contributed, allowing other researchers to feasibly re-implement our system and to devise efficient data processing pipelines using the framework. This system is also validated for use as the capture method of OSCE performances. As such, it can be used for capturing multiple possible human-object interaction tasks, as well as for pervasive computing applications.

Chapter 4 addressed a problem of convolutional neural network (CNN) approaches to action recognition, whereby it is necessary to train parallel networks for each image type, such as colour and optical flow. The devised early-fusion approach uses separable convolution, separating the learning of channel-wise and inter-feature relationships, to learn latent cross-modality features when training on multiple image types. It was shown that this approach improves performance over approaches on single image types for fine-grained action recognition (improving accuracy from 63% to 72%). In addition to using scene flow information, methods of including further temporal information, via the use of recurrent neural networks in combination with a CNN, were studied. A comparison was made between such CNN-based approaches and pose-based approaches for the use case of recognition of OSCE actions. For OSCEs, object pose tracking may become challenging when objects change shape and form, affecting recognition. As CNN approaches can be trained directly from raw image data, it is not sensitive to errors in separate pose estimate stages. Based on these findings, a CNN-based approach to fixed-setting fine-grained action recognition was validated.

Chapter 5 builds on the findings of Chapter 4 to further investigate neural network architectures for recognition of fine-grained actions, such as those involved in an OSCE. Specifically, spatio-temporal neural network architectures were investigated for this purpose. *WeaveNet*, a new architecture that permits fusion of multiple image types throughout a network, was proposed. A thin shallow parallel network branch maintains separation of the image types to achieve this fusion. Fusion throughout the network removes the need to extensively study permutations of possible fusion locations for combinations of image types. A method of performing fusion over temporal information, *Densely-Fused Action Images*, is also proposed and shown to improve results. The WeaveNet architecture outperformed existing techniques across multiple action granularities on the 50 Salads dataset, including an increase in accuracy from 73.4% to 82.7% for the 'eval' granularity. A significant dataset of OSCE performances, OSCE-V, was collected and labelled, and the WeaveNet architecture was evaluated against this new dataset.

In each of Chapters 4 and 5, we train our fine-grained action recognition networks on single camera viewpoints, achieving high levels of recognition. An optimal camera set-up for possible applications involving these recognition methods would involve such a single multi-modal camera arrangement, due to reduced complexity and processing

requirements. However, using the capturing framework described in Chapter 3 to capture datasets, such as it was for OSCE-V, allows research opportunities that may not be feasible with a single camera set-up. The availability of multiple viewpoints permits research into generalising over camera position, removing the requirement of fixed camera placement. The RGB-D data from multiple cameras can be used to generate synthetic novel views for training or testing. Another research opportunity is to learn to select the best viewpoint for action recognition when an action is obscured or occluded in other viewpoints. One other research opportunity is to generalise over the arrangement of the scene. Using the RGB-D data from three cameras, a more complete and accurate representation (such as a mesh or volumetric representation) of the scene can be made and subsequently used in augmentation strategies. For example, 3D objects can be removed or placed in the scene using this representation and thus used to strengthen action recognition under varying scene arrangements. Should these research opportunities be beneficial for a target application domain, the multi-camera recording framework of Chapter 3 is considered more optimal for research purposes.

6.2 Future Work

Overall, in this thesis, the contributions have focussed on fine-grained action recognition through combinations of colour and other image types, such as depth. Through estimation of semantic features, such as pose information, or through deeply-learned image features, high levels of recognition were achieved for this problem. This recognition level shows that combining image types in these ways is a worthwhile research direction, evidenced by the performance of the WeaveNet architecture. In future work, combining other image types may lead to further improvements. Specifically, infra-red images may capture temporal information due to the heat of touched objects. Combining this image type with colour and depth could improve fine-grained action recognition performance further. Combining pose information with deeply-learned image features could also improve performance. As part of an OSCE, the rigid objects can be tracked using the rigid object tracking approach of Chapter 3. State-of-the-art approaches for hand pose estimation could also be used to estimate human pose features. Combining these two distinct approaches, pose-based and deeply-learned image features, as part of a wider architecture could lead to recognition performance improvements. Other possibilities for improved recognition performance include characterising the performance of state-of-the-art action recognition models, such as SlowFast [141], on the recognition task of OSCE-V.

There are other possible research directions enabled by the contributions contained in this thesis. The availability of multi-camera information in the OSCE-V dataset permits research opportunities into generalising camera position, generalising scene arrangement using data augmentation, and combining network predictions for each view.

The colour and depth from multiple viewpoints can be used to generate new viewpoints, and the point cloud segmentation methods of Chapter 3 could be used to augment the background scenery. Research into the problem of action segmentation using the output image features of WeaveNet can also be explored in future work. An existing problem in action segmentation, which is applicable to OSCE-V, is that datasets are composed of a limited number of complete sequences. The OSCE-V dataset can be used to further investigate methods of overcoming this limited data problem when performing action segmentation. A possible approach would be to synthesise new sequences based on combinations of existing sequences to artificially increase the number of sequences. Research could focus on how to do this effectively, i.e. which sequences to join and where to join. These research directions are all made feasible by the contributions made in this thesis.

The action recognition approaches in this thesis and OSCE-V dataset can be used as the basis of research into pervasive computing applications for OSCE expert supervision. Such an application may have many possible benefits. It could be capable of providing timely feedback in an *online* manner, i.e. during, rather than after an OSCE. This aspect could be facilitated by the action recognition performance of WeaveNet. This application would also provide a scalable way to increase the level of students' practice under supervision. It could also provide a thorough debriefing to students, leading to better performance [343], and allowing a learner to develop self-directed learning skills. An automated system could provide a thorough debriefing to students after OSCE examinations. Such a system may also eliminate possible subjectivity biases [344, 345, 346], resulting in more objective and fair assessments of students. By recording OSCEs, students may also benefit from observing and reflecting on their performances [347]. The development of such an automated OSCE expert supervisor application is beyond the scope of this thesis, however we hope that the work contained in this thesis could be useful to research into such an application.

Appendix A

Recording Framework Candidate Camera Arrangements

To reach a final specification of the physical design of the recording framework of Chapter 3, a number of different candidate designs were considered. Each of these candidate designs are assessed under the proposed pipeline of Section 3.2 and the motivating factors of Section 3.1.1.

Single Colour Camera

An initial physical design involved the use of a single colour camera. This camera is situated opposite the performer of the task, looking down at the table upon which the task will be performed. Positioning the camera in this location allows for the observable area of the task area to be maximised. To assess the feasibility of this set-up, an exploratory experiment was devised that involved tracking a person's performance of a simple task.

Videos were recorded of a person moving coloured chips around a task table. The goal of the system was to count the number of chips of each colour that the performer moved from one side of the table to another.

Under the devised initial pipeline, we sought to track the positions of the hands of the performer and the chips as they moved across the table. To track the chips, the

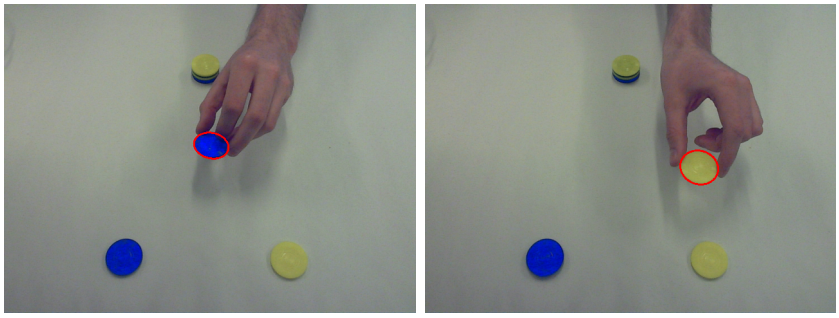


Figure A.1: Tracking of coloured chips using the mean shift algorithm

established technique of Continuously Adaptive Mean Shift was used [348], see Figure A.1. Images of the different coloured chips were collected from various angles in various lighting conditions and used for initial training of the tracker. To track the hands of the person, a Random Forest [326] binary skin classifier was trained using the hue and saturation values of a training set of skin and non-skin pixels in Hue-Saturation-Value (HSV) colour space. Morphological operations were then used to segment the hand regions in each frame, and a distance transform operation was used to identify a median point for each hand.

In these initial tests, it was found that some factors made the reliable tracking of chips difficult. A primary difficulty was occlusions. These were dependent upon the particular motion path that a subject used to perform the task. The occlusions were also dependent on the positioning of the person relative to the camera. This observation demonstrates the reasoning behind human assessors moving about the OSCE performance station to gain a better vantage point. Another difficulty encountered here was that reliable tracking of the performer's hands was challenging if objects of similar colours are used. Due to these difficulties in this constrained setting, in a more complicated scenario of multiple objects of less distinct colouring, reliable tracking would be more challenging. It may be possible to control such aspects, however, such controls would make it difficult to claim that the system is feasible to re-implement and would place overly strict restrictions on aspects such as lighting.

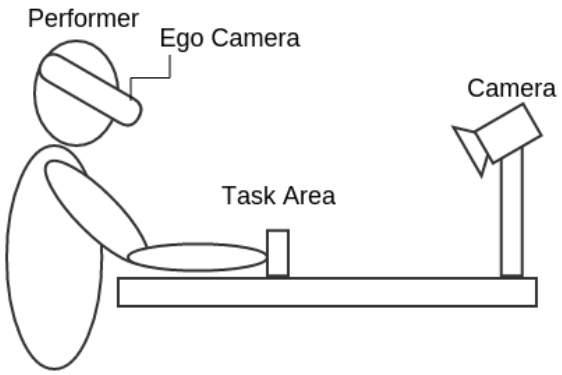


Figure A.2: A multi-camera set-up including an egocentric camera.

Ego Camera

To tackle the effect of occlusions, another potential set-up was devised, shown in Figure A.2. This set-up makes use of an ego-centric camera. By attaching a camera to the performer (an ego-camera), facing into the task area, this would eliminate a large number of these occlusions by effectively providing image data akin to what the performer is seeing.

An advantage of the ego-centric set-up is that a larger number of occlusions will be eliminated by providing images from the viewpoint of the performer. However, this

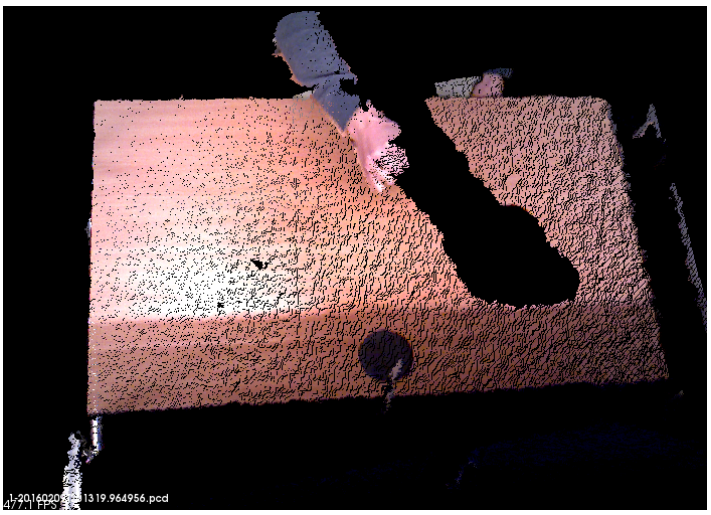


Figure A.3: A rendering of a coloured depth image illustrating the effects of occlusions. There is a jug in the subject's hand and a cup on the table. However due to occlusions, neither of these objects have valid depth data due to occlusions.

set-up also has distinct disadvantages regarding the requirements of the design of this system. One disadvantage is that the system would require the performer to wear an ego-camera. Although these cameras may have become more ergonomic in recent years, it would have some impact on the person's ability to perform the task by potentially restricting their movement. Furthermore, ego-cameras present a particular challenge in that it is possible that the performer may move their head quickly away from the task area, affecting the performance of any tracking algorithms.

Single Depth Camera

Another possible set-up configuration was to use a depth camera, such as the Microsoft Kinect, to observe the task area. Depth cameras have been shown to be particularly advantageous when used for tasks such as human pose estimation [8]. There are also numerous techniques developed in the area of 3D vision that allow for the pose of rigid objects to be estimated in six degrees of freedom [349, 350, 351]. Given the assumption that the objects are known in advance, object pose estimation techniques can be applied. Furthermore, an estimation of the pose of an object in a scene is more informative than typical bounding box estimates resulting from 2D object detection techniques. This additional information should allow for better classification rates by allowing for an object's pose to be used as an input to the action classification stage of the pipeline.

Utilising the property that the task area is composed of a large planar object (i.e. a desk), depth cameras also allow for easier segmentation of the table (and hence objects resting on the table) than is achievable with a comparable colour camera.

This arrangement, however, shares similar challenges to that of a single colour camera, in that occlusions of regions salient to the problem of fine-grained action classification may occur, as shown in Figure A.3

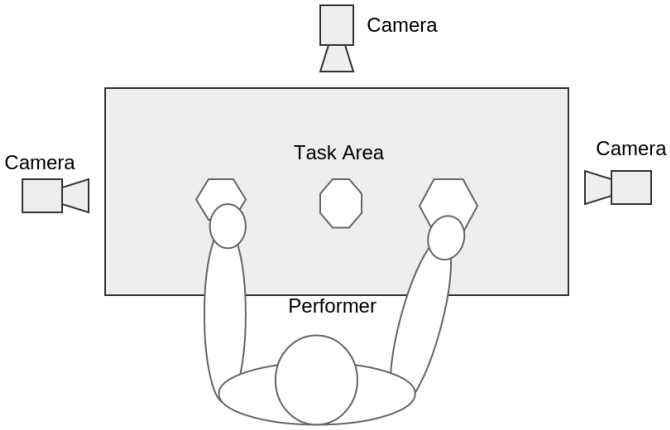


Figure A.4: A multi-camera set-up including a number of fixed viewpoints.

Multiple Depth Cameras

The multi-camera set-up has a disadvantage in that it adds to the complexity of the system to have a number of cameras observing the scene. However, by having different viewpoints, potential occlusions of actions are reduced. The cameras can be arranged in such a way that they maximise the view of the task area, such as in Figure A.4. This set-up also satisfies the motivating factor that the system should not impede the ability of the performer to complete the task. By having multiple sensors arranged this way, the classification accuracy has the potential to be increased by allowing object detection and tracking algorithms to be performed using the colour and depth images from each camera.

Furthermore, it has also been shown that utilising a small number of depth cameras can be as effective as a large number of colour cameras in generating a voxelisation of a subject [352].

Appendix B

Multiple RGB-D Camera Calibration

In Chapter 3, a multiple RGB-D camera arrangement was used to capture task performances. To reliably fuse the information from the sensors, and to provide as accurate data as possible, it is necessary to perform a thorough calibration of these cameras, intrinsically and extrinsically. The process used to perform this calibration is outlined below.

Firstly, we need to calibrate the cameras for their respective intrinsic parameters. This is performed using the method of Zhang [353]. This technique involves observing a known planar pattern from multiple positions. Image processing techniques are used to locate these known patterns in the collected images so that correspondences can be found across these images. Given these known correspondences, a closed-form solution is calculated and is further refined by non-convex optimisation of a maximum-likelihood cost function. This method is performed for each camera's colour and infra-red sensors, with tools built using the OpenCV library [354].

To describe the extrinsic calibration technique, it is necessary to first define some terms. A *point cloud* is a set $\mathcal{P}$ of points $\mathbf{p} \in \mathbb{R}^n$. This vector, $\mathbf{p}$, representing a point, contains elements for 3D position, denoted p_x, p_y and p_z. It may also include elements for properties such as colour, surface normal and surface curvature.

Using a homogeneous representation, the pinhole camera model for a depth camera can be expressed with the following relation,

$$\begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} d \cdot u \\ d \cdot v \\ d \end{bmatrix}. \tag{B.1}$$

Here, the normalised focal lengths in the x and y directions of the image plane, f_x and f_y, respectively, and the camera projection centre, (c_x, c_y), have been previously estimated during the intrinsic calibration. The matrix $\mathbf{R} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ represents a

rigid rotation, and the vector $\mathbf{t} \in \mathbb{R}^3$ represents a translation. We can assume that the camera is placed at the origin, $\mathbf{t} = \mathbf{0}$, aligned with the canonical reference frame, $\mathbf{R} = \mathbf{I}$. Thus, the depth value, d, at pixel location (u, v) in the depth image, allows us to calculate the three-dimensional position, (p_x, p_y, p_z), of this projected world point in closed-form.

The point clouds, $\mathcal{P}$ and $\mathcal{Q}$, for two calibrated depth cameras, are calculated with respect to different coordinate reference frames. Thus, to merge the two point clouds, it is necessary to estimate the Euclidean transformation, $\mathbf{T} : \mathbb{R}^4 \to \mathbb{R}^4$, that maps the coordinate frame of the source cloud, $\mathcal{Q}$, to the coordinate frame of the target cloud, $\mathcal{P}$.

To estimate this transformation between two point clouds, there must exist a point $\mathbf{p}_i \in \mathcal{P}$, such that $\tilde{\mathbf{p}}_i = \mathbf{T}\tilde{\mathbf{q}}_j$ for some $\mathbf{q}_j \in \mathcal{Q}$, where $\tilde{\mathbf{p}} = [p_x, p_y, p_z, 1]^T$ is the homogeneous representation of $\mathbf{p}$. This is assumable in our case as the regions that the cameras observe overlap to a large proportion. This transformation matrix can be parametrised by six degrees of freedom (three for rotation and three for translation), and hence to estimate the transformation, a minimum of three such correspondences are required.

To estimate $\mathbf{T}$, given n correspondences, we can use the sum of squared differences as a cost function,

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \left\{ \sum_{i=1}^{n} \left(\tilde{\mathbf{p}}_i - \mathbf{T}\tilde{\mathbf{q}}_i \right)^T \left(\tilde{\mathbf{p}}_i - \mathbf{T}\tilde{\mathbf{q}}_i \right) \right\}. \tag{B.2}$$

This least squares representation can be estimated using a nonlinear least squares estimation method such as Levenberg Marquardt [355]. However, given some of the strong constraints that can be made on the matrix $\mathbf{T}$, it is possible to calculate this transformation in closed form, analogously to similar problems in the 2D domain [356].

The transformation matrix $\mathbf{T}$ is composed of a rigid rotation matrix $\mathbf{R} : \mathbb{R}^3 \to \mathbb{R}^3$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, as per Equation B.1. Thus, we equivalently seek to find estimates for $\mathbf{R}$ and $\mathbf{t}$,

$$\hat{\mathbf{R}}, \hat{\mathbf{t}} = \arg\min_{\mathbf{R},\mathbf{t}} \left\{ \sum_{i=1}^{n} \left(\mathbf{p}_i - \mathbf{R}\mathbf{q}_i + \mathbf{t} \right)^T \left(\mathbf{p}_i - \mathbf{R}\mathbf{q}_i + \mathbf{t} \right) \right\}. \tag{B.3}$$

An expression for $\hat{\mathbf{t}}$ can be found by solving for the extrema with respect to $\mathbf{t}$,

$$\hat{\mathbf{t}} = \frac{\sum_{i=1}^{n} \mathbf{p}_i - \mathbf{R}\mathbf{q}_i}{n}, \tag{B.4}$$

which we abbreviate to $\hat{\mathbf{t}} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}$, where $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$ are the means of the respective correspondence points. Substitution into the cost function gives

$$\hat{\mathbf{R}} = \arg\min_{\mathbf{R}} \left\{ \sum_{i=1}^{n} \left(\left(\mathbf{p}_i - \bar{\mathbf{p}} \right) - \mathbf{R} \left(\mathbf{q}_i - \bar{\mathbf{q}} \right) \right)^T \left(\left(\mathbf{p}_i - \bar{\mathbf{p}} \right) - \mathbf{R} \left(\mathbf{q}_i - \bar{\mathbf{q}} \right) \right) \right\}. \tag{B.5}$$

If we define the matrices $\mathbf{P} = [\mathbf{p}_1 - \bar{\mathbf{p}}, \mathbf{p}_2 - \bar{\mathbf{p}}, \ldots, \mathbf{p}_n - \bar{\mathbf{p}}]$ and $\mathbf{Q} = [\mathbf{q}_1 - \bar{\mathbf{q}}, \mathbf{q}_2 - \bar{\mathbf{q}}, \ldots, \mathbf{q}_n - \bar{\mathbf{q}}]$, then we can rewrite the equation as

$$\hat{\mathbf{R}} = \arg\min_{\mathbf{R}} \{||\mathbf{P} - \mathbf{R}\mathbf{Q}||_F\} \tag{B.6}$$

where, the rigid rotation is subject to the constraints that $\mathbf{R}\mathbf{R}^T = \mathbf{I}, |\mathbf{R}| = 1$, and $||\cdot||_F$ denotes the Frobenius norm. The problem has thus been reduced to an orthogonal Procrustes problem, and a closed form solution can be computed via Singular Value Decomposition, whereby $\mathbf{P}\mathbf{Q}^T = \mathbf{U}\mathbf{L}\mathbf{V}^T$ and $\hat{\mathbf{R}} = \mathbf{V}\mathbf{U}^T$.

The problem remains of how to find the correspondences. In the domain of 3D object pose estimation, specific pipelines have been developed for the registration of point clouds. These pipelines typically involve an initial stage of keypoint estimation, whereby keypoints are found in the source and target point clouds that exhibit distinctive properties. At these keypoints, feature descriptors are determined that can then be matched in a higher dimensional space using an efficient data structure such as a kd-tree. Based on the analysis of Tombari et al. [357], Intrinsic Shape Signature (ISS) keypoints were determined for the point clouds. Signatures of Histograms of Orientations (SHOT) [358] were found and matched across the source and target point clouds. Given that in our case, the point clouds are dominated by a task table, with a homogeneous surface, it was found to be difficult to find valid correspondences between the two point clouds.

However, the fixed setting of the recording rig allows us to control aspects that may aid in the detection of features. Hence to find the correspondences, we utilise image processing techniques similar to the method for colour camera calibration [353], to detect distinctive landmarks. Known planar targets were used in the form of black squares on a white background. Harris corner detection was used to detect corners in the colour images of each camera, and contour detection was used to validate the corners as corners of the landmarks. The depth image can be registered to the colour images via the factory calibration of the cameras and hence the 3D position at the corners is observable. In some instances, the points were unobservable at these specific points due to holes in the depth maps. To overcome this, we calculate the point as the mean value of the points observed in a small circular region of the colour image. Due to the fixed nature of the landmarks, the correspondences can be assigned across the respective point clouds.

Performing Singular Value Decomposition gives us an estimate of the transformation, however, due to noise at the individual points as a result of the sensor characteristics, and uncertainty in the estimation of the landmark corners, the registration may exhibit errors. Thus a refinement stage is necessary to achieve a better quality registration. A technique for such a refinement is the Iterative Closest Point algorithm [308]. This greedy algorithm requires the determination of the closest points in $\mathcal{P}$ to the set of points $\mathcal{Q}$ transformed by T. A cost function is formulated similarly to Equation

B.2,

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \left[\sum_{i=1}^{n} \mathbf{1}_{\mathcal{D}} \left(||\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i|| \right) \left(\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i \right)^T \left(\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i \right) \right], \tag{B.7}$$

where each component of the sum is weighted by an indicator function on the closed set $\mathcal{D} = \{d \in \mathbb{R} \text{ where } ||d|| \leq d_{max}\}$ and the vector $\tilde{\mathbf{m}}_i$ represents the closest point in $\mathcal{P}$ to the transformed point $\mathbf{T}\tilde{\mathbf{q}}_i$. The algorithm begins with an initial estimate of $\mathbf{T}$ and iteratively updates it by determining any updated matches once the cost function has been minimised. The algorithm terminates when a pre-determined criterion is met.

A variation of the above cost function is to measure the distance from a source point to the plane defined by the matching point and the normal at that point in the target cloud [359].

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \left[\sum_{i=1}^{n} \mathbf{1}_{\mathcal{D}} \left(||\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i|| \right) \left(\tilde{\mathbf{n}}_i^T \left(\tilde{\mathbf{m}}_i - \mathbf{T}\tilde{\mathbf{q}}_i \right) \right)^2 \right]. \tag{B.8}$$

where $\tilde{\mathbf{n}}_i$ is the normal at point i. The normals are calculated for each point in the target point cloud based on the local region of neighbouring points, and assuming that normals are directed towards the origin. The transformation can be calculated in closed form using the Singular Value Decomposition method, similar to Equation B.2.

To ensure a consistent density across the source and target point clouds, we perform a voxelisation filter, so that each cubic voxel of width 1cm contains only a single point, calculated as the centroid of the points present in the voxel. This also allows sensible control over the indicator function in Equation B.8. The stopping criterion used for the algorithm is to terminate once the delta in the mean square error between successive iterations falls below a certain level.

Since the task table dominates the point clouds to be registered, the algorithm is susceptible to convergence to local minima. To overcome this, a number of 3D calibration targets were constructed, shown in Figure B.1. These targets take the form of Platonic solids, which all have prominent vertices and sharp changes in surface normals. These surfaces add necessary discriminating correspondences between the point clouds to allow the ICP algorithm to converge robustly.

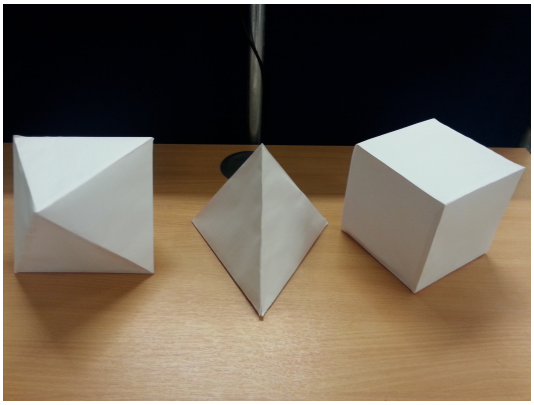


Figure B.1: Constructed platonic solid models used as calibration targets.

Appendix C

Point Cloud Processing Software Pipeline

To implement a framework for the recognition of human-object interactions, it is necessary to test numerous different techniques, especially for pre-processing stages, to validate their use as part of a pipeline. When iterating on pipelines techniques, maintaining separate pipelines for each of these combinations becomes unwieldy. Furthermore, due to the significant amount of processing involved when working with three-dimensional point cloud data, testing an individual pipeline can be time-consuming. Thus, to iterate quickly, a software framework that allows the combination of different techniques, and the re-use of previous pipeline stages, was developed.

Requirements were devised to ensure that a developed framework meets the needs of constructing multi-camera point cloud pipelines:

1. The framework needs to allow combinations of numerous different techniques to be specified easily and succinctly.

2. A pipeline should be easily adaptable to different input types as well as to inputs received from live camera feeds or from recorded point cloud files.

3. Certain stages of a pipeline may be very computationally intensive, and thus potential pipelines could suffer from low performance if implemented naively. To allow for fast iteration, the framework should permit a higher level of performance by default for a user combining pipeline processes.

4. The framework should include pre-processing components that can be used as part of a pipeline.

5. The framework should be extensible to allow a user to develop custom pipeline stages for use as part of a pipeline.

An object-oriented design approach was adopted to achieve the required modularity within the framework. This approach allows us to define objects representing specific

stages of the pipeline. As such, we define a node as a stage of a pipeline responsible for a single function. Nodes form the basis of a pipeline architecture constructed with the framework. Depending on their location in the pipeline, nodes can be classified into three different types: producers; processors; and consumers. A producer is a node that acts as a source, producing inputs for a pipeline, such as a file reader or a live camera feed. A processor is a node that takes input data and performs a specified processing technique and outputs the results. A consumer is a node that acts as a sink, consuming the output of a pipeline, such as a file writer or display window. By separating the responsibility of the nodes, it will allow for re-use and extensibility of such nodes as part of multiple potential pipelines.

Under the framework, a pipeline can be constructed as a chain of nodes, and as such, it is necessary for the nodes to be able to receive inputs from previous nodes and to send outputs to subsequent nodes. To achieve this, nodes communicate via shared buffer resources. This approach permits the nodes to operate in parallel, with synchronisation only necessary during access to connected shared buffers. Each data element is represented by a wrapper around an underlying data type (such as a point cloud) and an associated timing to allow for sorting of buffers upon insertion of a element.

The framework was built using C++, chosen for its performance and compatibility with software libraries for interfacing with depth cameras. The Boost libraries were used for multi-threading capabilities and the Point Cloud Library [337] for low-level point cloud utilities and methods. In the implementation, each node is allocated a native hardware thread allowing it to operate in parallel to other nodes. The code makes judicious use of class templates throughout, facilitating variability in the types of inputs that can be used as part of the pipeline without any modification needed to the nodes, e.g. point clouds including colours or surface normals.

To construct a pipeline, a user of the framework can create the required nodes and buffers and connect them with exposed methods. This ease of specifying a pipeline satisfies requirement 1. The generic implementation of the nodes allows specialisation to different data types without modification of the underlying node, satisfying requirement number 2. The parallel nature of the processing ensures that a pipeline will see speed improvements when running on multi-threaded systems, as compared to a sequential pipeline implementation (requirement 3). As part of the framework, several common pre-processing nodes are included (requirement 4). The object-oriented design also allows for these processing nodes to be easily extended by overriding a single function in the subclass (requirement 5).

The framework thus satisfies the requirements of our use case. It may also benefit researchers in other application areas that involve the processing of point cloud data, such as autonomous driving research and robot navigation. The code framework is located at https://github.com/leaveitout/pcltools.git.

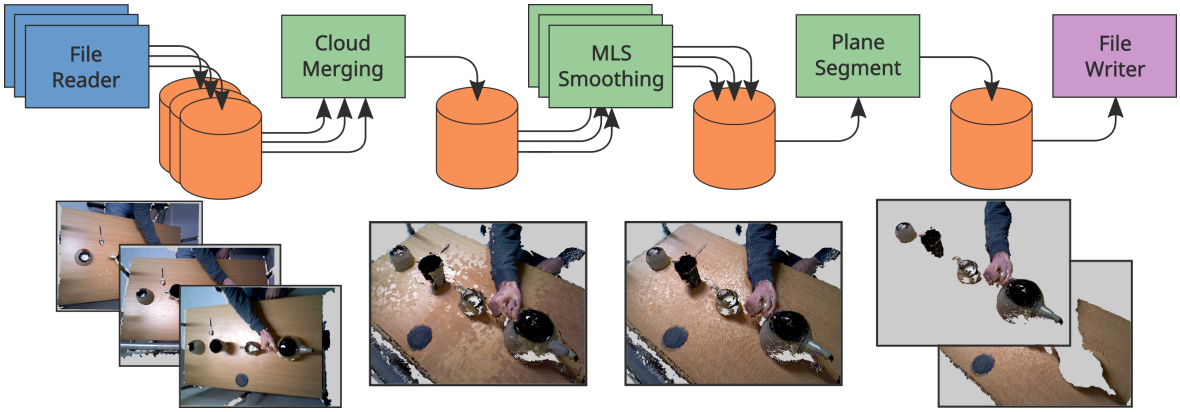


Figure C.1: An illustration of an example pipeline implemented using the devised software framework, included in Section 3.3.2 and included here for completeness. The pipeline is responsible for reading three sets of camera recordings from disk, merging, smoothing, segmenting and saving the outputs to disk. Each of the coloured rectangles represents nodes in the pipeline (blue for producing, green for processing and purple for consuming) and the orange cylinders represent buffers linking the nodes. In this specific pipeline, the Moving Least Squares (MLS) smoothing operation is particularly processor intensive and so has multiple processor nodes dedicated to this step to balance the pipeline load evenly.

Appendix D

Cup Of Tea: Dataset Labelling Rules

Below are the rules used to label the Cup Of Tea dataset, discussed in Section 3.3.4.

Action	*Begins*	*Ends*	*Frames*
Pour tea	Subject picks up teapot.	Subject releases teapot.	8205
Pour milk	Subject picks up milk jug.	Subject releases milk jug.	4540
Place sugar	Subject puts spoon into sugar bowl.	Subject has placed sugar in cup.	3587
Stir	Spoon enters tea cup.	Spoon is removed from teacup.	3172
Background	Otherwise.	Otherwise.	6139

Table D.1: The criteria for labelling the actions for each video frame, as well as the composition of action classes of the entire dataset.

Appendix E

Tree Parzen Estimator Hyperparameter Selection

Expected Improvement [325] is used as a test to identify a point in hyperparameter space, x^*, to sample next during hyperparameter selection. It can be defined [324] as the expectation, under some model M of a fitness function $f : \mathcal{X} \to \mathbb{R}$, that $f(x)$ will negatively exceed some threshold y^*,

$$\mathbf{EI}_{y^*}(x) := \int_{-\infty}^{\infty} \max(y^* - y, 0) p_M(y|x) dy, \tag{E.1}$$

where $\mathcal{X}$ is the space of the hyperparameters. In the case of Tree Parzen Estimators, $p(x|y)$ and $p(y)$ are modelled, with $p(x|y)$ defined using two densities formed from the observations,

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \tag{E.2}$$

where y^* is chosen so that the respective densities contain a specific proportion of the observations.

It has been shown [324], that the Expected Improvement is inversely proportional to $g(x)/\ell(x)$, and thus, by maximising this value, the algorithm suggests successive points that have high probability of being in the density $\ell(x)$ and low probability of being in $g(x)$.

Appendix F

Cup Of Tea: Action Classifier Hyperparameter Searches

In Tables F.1 and F.2, the prior hyperparameter distributions for the action classification models are shown, as discussed in Section 3.3.4. These hyperparameters were chosen to search over by inspection of their effect upon performance of initial model runs.

Parameter	*Prior*
Stride	1, 2 or 4
Sequence Length	8, 16, 32 or 64
Recurrent Units	64 or 128
Number LSTM layers	1, 2 or 3
Bidirectional	True or False
Dense Kernel $\mathcal{L}2$	$U(0.0001, 0.01)$
LSTM input dropout rate	$U(0.0, 0.3)$
LSTM recurrent dropout rate	$U(0.0, 0.3)$
Softmax dropout rate	$U(0.0, 0.5)$
Initial learning rate	$U(0.0001, 0.01)$

Table F.1: The prior distributions used for the hyperparameter selection. $U(x, y)$ denotes the uniform distribution between x and y. Uniform distributions are used in all of the other discrete cases. These distributions are chosen based on a number of initial tests runs.

To understand the effects of hyperparameter values, we plot the metric obtained for pairs of hyperparameters. In the cases of Random Forest and Gradient Boosted Decision Trees, we observe that there are clear regions of hyperparameter space that result in better performance, as visible in Figure F.1. In particular, we observe that the maximum depth of the tree estimators has a significant effect upon the results. There also appears to be relationships between dimensions of the hyperparameter space, as can be seen for sequence stride and maximum depth for Gradient Boosted Decision Trees. For the case of the LSTM classifier, there are similar clearly optimal regions

Parameter	*Prior*
Stride	1, 2 or 4
Sequence Length	8, 16 or 32
Number Estimators	$U(10, 50)$
Maximum Tree Depth	$U(5, 25)$

Table F.2: The prior distributions used for both the Random Forest and for the Gradient Boosted Decision Trees hyperparameter selections. $U(x, y)$ denotes the uniform distribution over $[x, y]$. These distributions are chosen based on a number of initial tests runs.

for hyperparameters, such as LSTM input dropout, as can be seen in Figure F.2. It is difficult to determine further relationships visually between hyperparameters. This may be explained by the effect of the larger number of hyperparameters to be searched over, in addition to the stochastic training of the classifier.

In the hyperparameter values of the best performing LSTM classifier, as shown in Figure F.3, we note that a small learning rate in conjunction with relatively high LSTM recurrent dropout values are used. The dropout rate of the softmax layer is also relatively high. This indicates that the training required significant regularization to prevent overfitting.

Parameter	*Prior*
Stride	1
Sequence Length	64
Recurrent Units	64
Number LSTM layers	1
Bidirectional	True
Dense Kernel $\mathcal{L}2$	6.192×10^{-6}
LSTM input dropout rate	0.1424
LSTM recurrent dropout rate	0.31265
Softmax dropout rate	0.28725
Initial learning rate	0.00125

Table F.3: The hyperparameter values of the best performing LSTM classifier.

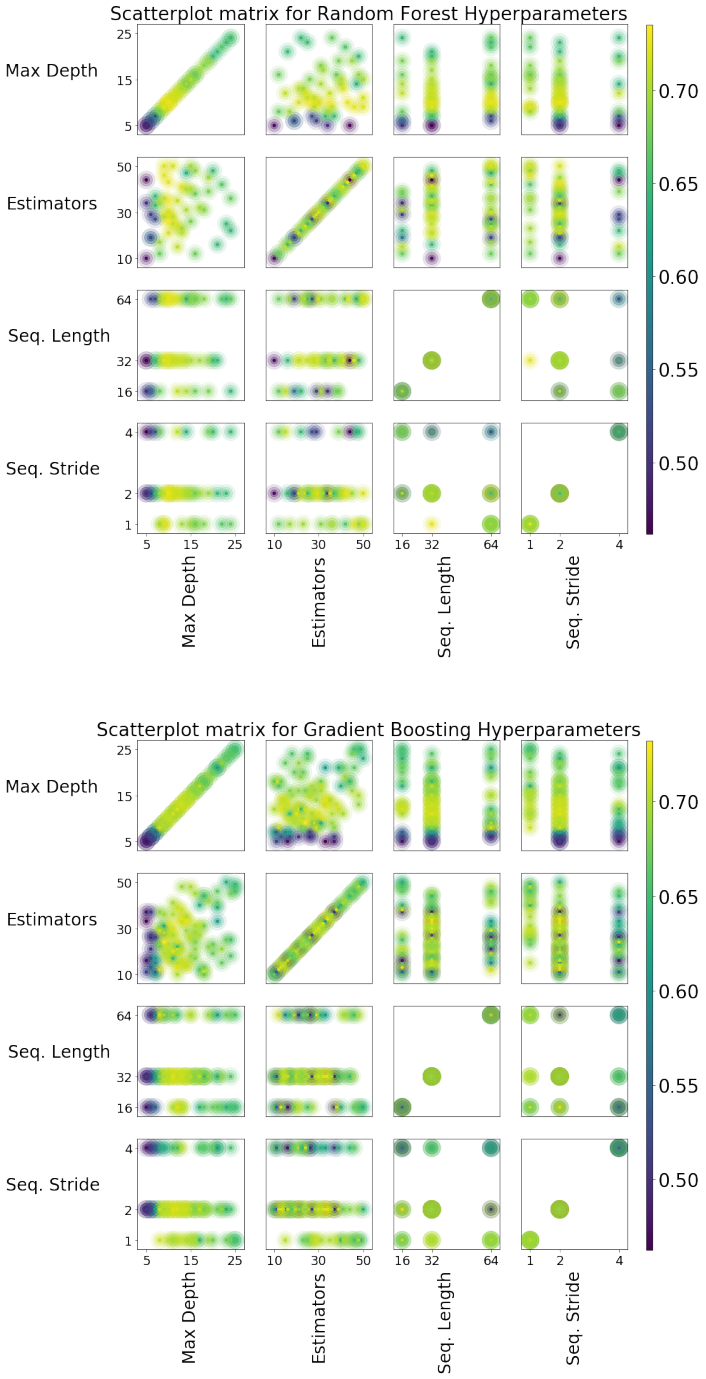


Figure F.1: The heat maps for the hyperparameter selection for the Random Forest classifier (top) and the Gradient Boosted Decision Trees (bottom).

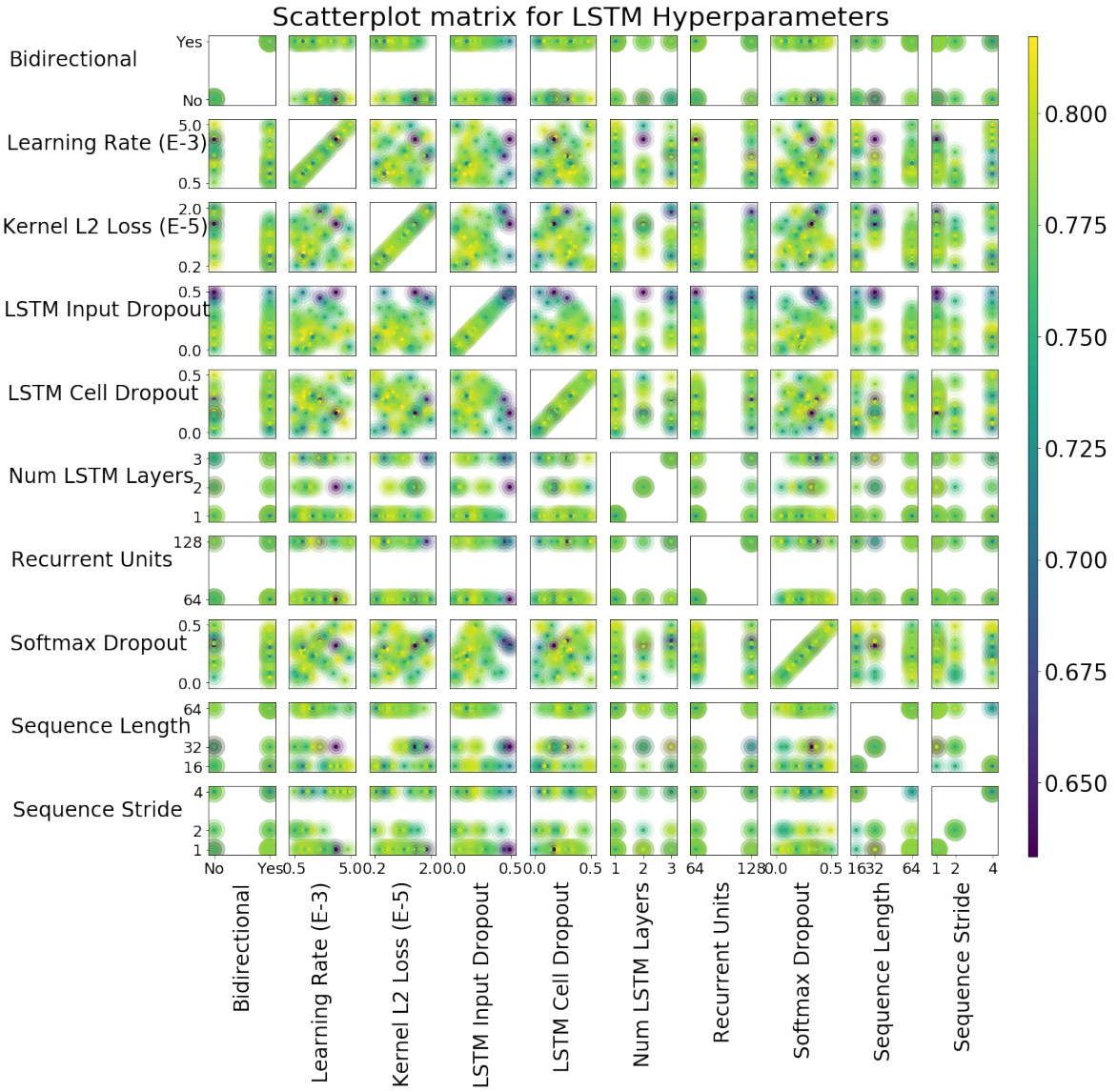


Figure F.2: A scatter plot showing the results of the hyperparameter selection for the LSTM classifier. We observe that certain regions of the hyperparameter space result in better results, specifically lower LSTM Input Dropout and lower kernel $\mathcal{L}2$ loss.

Appendix G

Cup Of Tea: Improving Data Quality

In Section 3.3.1, the implementation of a technique for improving the data received when multiple depth cameras observing the same scene was detailed. To validate this implementation, experiments are performed in line with the work of Butler et al. [320], with augmentations for our specific arrangement.

In the camera arrangement, offset weighted motors were attached to the leftmost (1) and the rightmost (3) cameras. There was no offset weight motor attached to the central camera (2).

To determine the change of quality of the point cloud data that when the vibratory effects are active, an experiment to count the number of invalid pixels was performed. This experiment involved capturing point clouds from each of the cameras under various different test conditions and counting the number of invalid pixels that were present in the resulting depth data. The results of this experiment are shown in Table G.1.

Test condition	Cam.	Mean bad pixels	Std. dev.	Min. bad pixels	Max. bad pixels	%age change mean
Single camera	1	37,895.1	56.5	37,730	38,042	-
	2	41,832.2	270.5	41,131	42,545	-
	3	44,517.1	299.7	43,614	44,517	-
Multi. camera, all static	1	52,543.9	386.2	51,369	53,788	38.7%
	2	45,140.4	207.0	44,603	45,710	7.9%
	3	60,992.8	511.9	59,416	62,143	37.0%
Multi. camera, 1 and 3 shaking	1	37,830.2	316.0	37,466	39,008	-0.2%
	2	44,040.6	225.9	43,457	44,580	5.3%
	3	45,791.2	643.5	44,466	47,741	2.9%

Table G.1: Statistics relating to the number of invalid pixels from 150 point clouds recorded of the same scene. The total number of potential pixels is 307,200. The rightmost column represents the percentage change with respect to the same single camera observing the scene.

In these results, it was found that there were significant increases in the number of invalid pixels for the two side cameras (1 and 3) when all three cameras were observing the same scene. However, with the shaking induced by the offset weight motors, the number of bad pixels was reduced to within 2.9% above the single-camera values. For the central camera (2), it was found that the presence of multiple cameras observing the scene did not have as drastic an effect as it did on the side cameras. With multiple cameras observing, mean invalid pixels increased by 7.9%, and with the shaking of cameras 1 and 3, this increase is reduced to 5.3%.

Thus, in our particular case of three depth cameras observing a small close scene, the camera shaking methodology is suitable for increasing the amount of valid data available for later processing.

The presence of noise due to multiple cameras, as shown in Figure G.1a, is another aspect of the technique to be evaluated. To measure this value, a segmentation of the table region is performed using the method detailed in 3.2.2. All points that lie above and below the convex hull estimated for the table are included as potential table points. The standard deviation of the signed distances of the points to the estimated table provides a measurement of the spread of points about the table and hence provides a heuristic for the amount of noise present in the point clouds. This experiment is performed for each camera for the three scenarios as used in the previous experiment, and the results are shown in Table G.2.

Test condition	Cam.	Mean distance	Std. dev.	Min. distance	Max. distance	%age change std. dev.
Single camera	1	0.1636	0.0926	-28.608	25.907	-
	2	0.1541	0.0755	-26.509	28.396	-
	3	-0.0548	0.0679	-25.699	29.024	-
Multi. camera, all static	1	-0.1661	0.1221	-24.494	26.627	31.81%
	2	0.2148	0.1040	-30.088	25.556	37.66%
	3	-0.0383	0.1033	-32.567	32.244	52.25%
Multi. camera, 1 and 3 shaking	1	0.1095	0.1068	-30.987	33.985	15.28%
	2	0.1537	0.0860	-30.755	27.023	13.91%
	3	-0.0718	0.0777	-33.377	39.257	14.53%

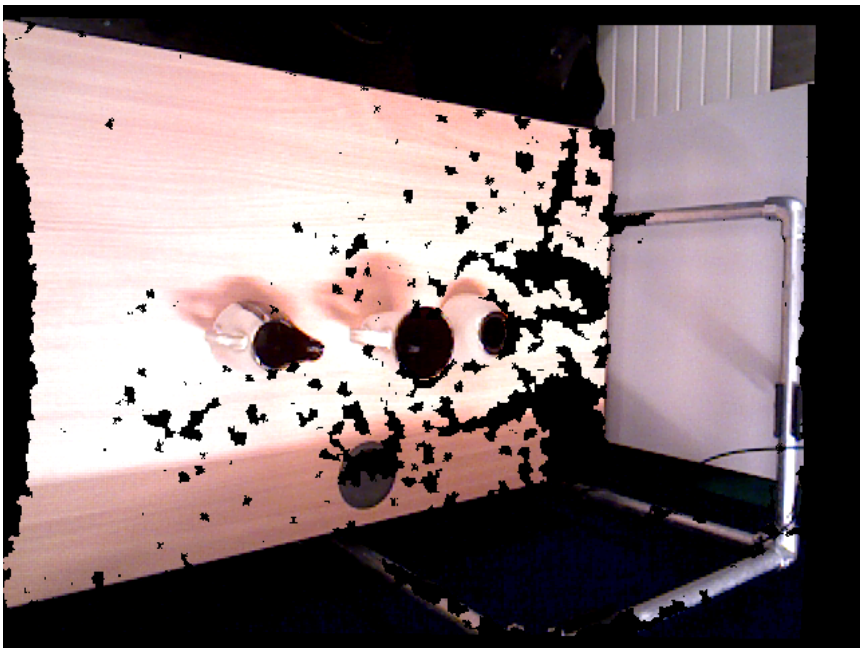
Table G.2: Statistics relating to the signed distance of points classified as belonging to the table to the fitted table model. The rightmost column represents the percentage change in standard deviation with respect to the value for the same single camera observing the scene. All units are reported in millimetres.

Similar to the previous experiment results, it was found that there were significant increases in the spread of points about the fitted table when all three cameras were observing the scene, with an increase in the standard deviation of up to 52%. This increase was significantly reduced when the vibration motors were used, with the maximum increase being 15%.

The minimum and maximum distance of all points are relatively large compared to the mean and standard deviation values in each of the three cases, indicating that these values may be outliers resulting from the incorrect inclusion of individual points that may not strictly belong to the table, but rather to some of the surrounding structures.

Again, in our multi-camera case, the camera shaking methodology significantly improves data quality in comparison to static cameras.

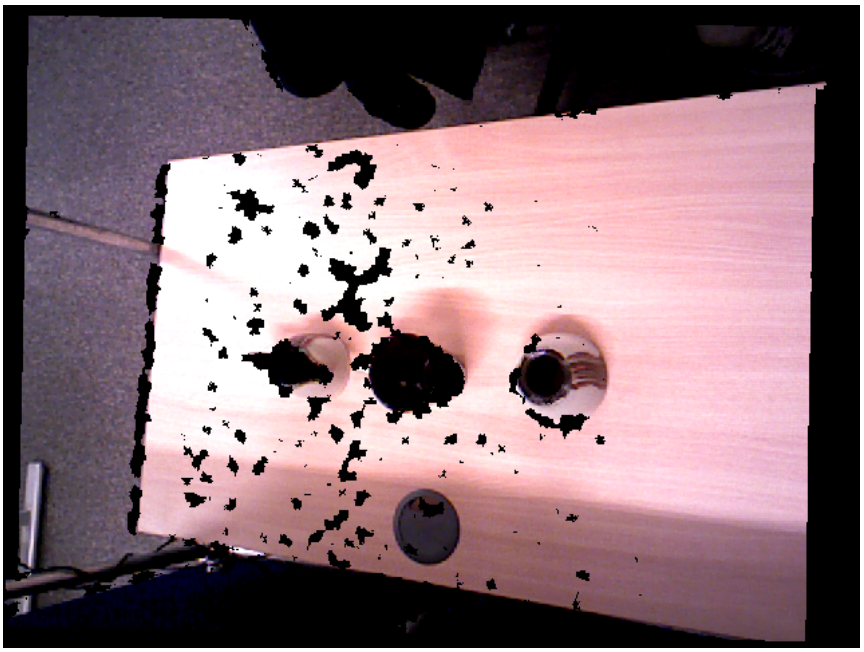
In Figures G.1 and G.2, qualitative results are shown that display the reduction in the number of holes present in point clouds when the multiple cameras are observing the same scene. The results with the Shake 'N' Sense active also preserve fine details along the edges of surfaces.



(a) Without Shake 'N' Sense.

(b) With Shake 'N' Sense.

Figure G.1: A qualitative result showing the difference in point cloud quality observed from Camera 1 (right). There is an observably larger presence of holes in (a) in comparison to (b).



(a) Without Shake 'N' Sense.

(b) With Shake 'N' Sense.

Figure G.2: A qualitative result showing the difference in point cloud quality observed from Camera 3 (left). There is an observably larger presence of holes in (a) in comparison to (b).

Appendix H

Gated Recurrent Unit Temporal Fusion: Fifty Salads Test Split Evaluation

We examine the performance of the GRU Temporal Fusion approach, of Section 4.2.1, across the test splits of the 50 Salads dataset. Comparing these results, shown in Table H.1, to those of the Single Image Network, in Table 4.4, we observe that precision, recall and F_1 scores are significantly improved. The accuracy metric, conversely, has deteriorated for each split. A further observation from Table H.1 is that two test splits, split 3 and split 5, show relative under-performance when compared to the results of the other splits.

Split	*Accuracy*	*Precision*	*Recall*	F_1 *score*
1	66.84	59.14	61.27	55.45
2	67.77	60.06	64.25	58.54
3	51.67	50.48	45.56	41.07
4	66.53	64.70	59.26	59.18
5	48.58	52.13	48.41	43.69

Table H.1: The classification metrics across each of the test splits of the 50 Salads dataset.

To examine the issue further, we inspect the training profiles for each of the splits, shown in Figure H.1. We note that there is a large variation between the test accuracy (feint green lines) for the different splits. As such, there lies potential benefit in testing further regularisation methods and network configurations (e.g. the number of RNN layers and recurrent unit per layers) to improve performance across these test splits. The elongated training time of the combined CNN-RNN model (six days for cross-validation testing with GRU Temporal Fusion approach, trained on a pair of Nvidia 1080 graphics cards), reduces the ability for testing a wider range of parameter configurations. In our testing, we used the Adam optimiser to avoid the costly process of manually scheduling the optimisation learning rate schedule by inspection. With sufficient resources, this manual scheduling process may lead to improved performance.

Figure H.1: The training accuracy profile for the GRU Temporal Fusion approach. The feint lines represent individual training and testing splits of the datasets.

Appendix I

OSCE-V Dataset Design

In this thesis, we identify techniques for the recognition of the fine-grained actions involved in goal-directed tasks, such as OSCEs. As discussed in Section 2.2.6, there does not exist any relevant dataset that covers performance of an OSCE skill. As such, attention needs to be paid to the design decisions that may affect the usefulness of the dataset for future research. Here, we discuss these decisions and give the motivations behind each of them.

I.1 OSCE Selection

The selection of the OSCE skill to study is an important decision for the dataset. A wide range of OSCEs exist to test medical practitioner knowledge and skills in specific areas, covering competencies such as patient communication and diagnostic ability. We limit our selection to OSCE skills to those that involve procedural techniques rather than examinations where the student must diagnose a patient in vivo (i.e. alive). We further limit selection to OSCE skills that can be performed on a simulated patient to reduce health and safety risks to any persons during the recording of our dataset. Under these criteria, the shortlist of possible OSCE skills were: intravenous cannulation; venepuncture; and male catheter insertion. It is necessary then to select one of these to concentrate on.

The possible OSCE skills vary in terms of complexity, length and difficulty. The male catheter insertion OSCE involves less steps, however these steps are of an arguably greater difficulty. These difficult steps may result in performances in which the task cannot be completed. This is more likely when the practitioner has had no previous training in the particular OSCE. As this may present a challenge for collection of the dataset, effectively reducing the number of valid recordings, we discount this OSCE skill from consideration. The intravenous cannulation OSCE and venepuncture OSCE are similar in the range of steps. However, the intravenous cannulation involves a larger number of steps, with many steps repeated. As the venepuncture contains a similar range of steps which can be performed in a shorter period, we give preference to this

OSCE skill in our selection. The shorter time required to complete this skill will also allow for a larger number of completed performances to be recorded under time and storage constraints.

I.2 OSCE Design

Venepuncture is the medical procedure in which a needle is inserted into a patient's blood vessel. As part of the process of phlebotomy, it is performed to collect blood samples for possible diagnostic analysis. It is performed by clinicians, nursing staff, or dedicated personnel [360], often as part of a pre-operative procedure.

Risks associated with the procedure include infection of the needle insertion site, needle stick injuries to the health worker, and mislabelling of samples leading to misdiagnoses. Risk factors for infections include a lack of aseptic techniques and good hand hygiene, poor or little training, and whether the procedure is being performed in an emergency room [361]. Aseptic techniques are a set of methods of handling medical implements that do not compromise the sterility of the implements. For the venepuncture procedure, a key aseptic technique is that the sterile needle is not touched by the performer during the procedure. There are other factors that determine success of a venepuncture procedure. If the sample is collected incorrectly, for example by using an incorrect needle gauge, the sample risks being invalidated by haemolysis (a breaking down of the blood cells), requiring the procedure to be repeated [362] which may cause undue discomfort and stress for the patient.

Figure I.1: Image showing the filling of a blood sample bottle as part of a venepuncture procedure. Image in public domain.

To minimise the risk of these complications and improve outcomes, implementation of formal guidelines regarding the procedure of venepuncture is advised [363] and has been shown to improve sample quality [364]. Such formal guidelines lend themselves well to examination via an OSCE to improve practitioner knowledge and ensure adherence to these guidelines [365]. Furthermore, changes in the guidelines can be adopted into the OSCE, and should significant changes occur, practitioners can be examined with the revised OSCE to ensure that the new guidelines are followed. Accordingly, in designing the set of venepuncture OSCE steps for our dataset recording, we use the World Health Organisation guidelines [366] in conjunction with local guidelines, tailored for the specific equipment arrangement at the dataset collection location. This was performed in collaboration with the Clinical Skills Tutor at Tallaght Hospital Dublin.

The devised OSCE steps that we use in dataset recordings are broken down into

conceptually distinct stages to aid the practitioner in learning, as shown in Appendix K.1. The first stage involves steps relating to preparing the area for the procedure, such as cleaning a tray that is used for the procedure. This is followed by a stage relating to the identification of the vein for the procedure. In our case, a simulated adult male arm will be used which includes an anatomically accurate vasculature system. The next stage covers preparation, by cleaning with an alcohol swab, of the identified site for the venepuncture. The following stage is performing the venepuncture itself, which contains the most fine-grained and difficult steps. The final stage of the procedure relates to the completion of the sample documentation and the cleaning of the procedure area. In each of these stages, hand hygiene is ensured through repeated washing of hands using alcohol gel.

This arrangement closely resembles an examination station that would be used in a venepuncture OSCE. There are, however, certain aspects that differ from exam conditions. In an OSCE, there are requirements regarding the checking of patient details and communicating with the patient throughout the procedure. During an examination, this stage would be performed by communicating with the examiner. As we are interested in solving the challenging problem of recognising the actions from images, we do not enforce this aspect as part of the OSCE steps. In an instantiation of a real-world system, such a process could conceivably be performed using an off-the-shelf speech recognition method to record the students speech and check against a standard script. This is beyond the scope of the thesis goals which are concerned with the recognition of the fine-grained actions involved in the performance. Documenting the samples correctly is also an important aspect of a venepuncture OSCE. For similar reasons to the patient communication requirements, we do not enforce correct completion of the documentation details. Furthermore, in many modern venepuncture arrangements documentation may be pre-printed prior to the procedure to remove this as a source of human errors.

In future work, it may be possible to explore methods to tackle such issues, but for now we are focussed on the challenging problem of recognising the constituent actions of the OSCE performance, and thus the dataset will primarily reflect this concern.

I.3 Subject Selection

A total of 20 subjects were recruited for the recordings. The subjects were all adults, with a broad range of ages and backgrounds represented. There was no requirement of medical experience, of experience of the procedure, or of experience of the OSCEs. This decision reflects a desirable aspect of a possible pervasive computing application based on the dataset, i.e. that it can robustly recognises the actions of both novice and expert practitioners. As a result, the skill level of the subjects ranged from no experience, to some years of medical training, to many years of professional experience.

I.4 Subject Briefing and Instructions

Prior to the recording of each experimental subject's session, they were briefed on the experiment. They were asked to read an information sheet describing the experimental procedure, as shown in Appendix K.2. They were also briefed about the potential health and safety concerns regarding the use of a medical needle. Each subject is asked to perform the venepuncture task a total of three times. In test recordings, we found that the complexity of the task was challenging for novice practitioners during their early attempts, whilst a third attempt was often more assured.

Before the recordings, subjects are allowed to read over the list of steps of Appendix K.1, ask the experimenter any questions, and view a video of the task being performed correctly. The Medical Skills Tutor from Tallaght Hospital Dublin is present for the duration of the experiment to answer questions regarding the task.

The subjects are asked to attempt to perform the tasks based on their memory of the list of steps. This decision was taken because we wanted to capture the typical mistakes in the task order that people make when attempting the venepuncture OSCE. However, should the subject forget a next step at any point during the performance, they are allowed to ask for a prompt from the experimenter. In the cases of the novice subjects, there were frequent requests for prompts while the more experienced subjects were able to complete the task with few requests for prompts.

I.5 Recording Method

To record the dataset, the multi-modal recording framework of Chapter 3 is used, Three Asus Xtion Pro Live RGB-D cameras record each performance from different vantage points. Once again, the vibration motors are used to improve the data quality as described in Section 3.3.1.

The dataset is recorded at two different locations. This decision was taken for the logistical reason of maximising the total number of available subjects. For the dataset, this adds a challenge to the recognition problem in that any devised technique needs to be robust to changing location factors, such as lighting.

I.6 Action Labels

To quantitatively assess our system for the recognition of the fine-grained actions involved in an OSCE performance, it is necessary to label entire performances with canonical action labels. To do so, we must first identify the set of action labels to use. To determine the set of actions labels, we inspect the steps as listed in Appendix K.1. These steps also help delineate the boundaries between actions. The action boundaries are kept consistent as possible by identification of intentional action units, such as grab, pick up, or touch, for the start, and place, drop, or put down, for the end.

These boundaries are identified for each action through the inspection of a subset of the dataset, and are listed in Appendix K.3. In cases where the action boundary is indistinct, a dominant action is selected based on the perceived focus of attention of the subject. An example of such a case may be when the subject is applying cotton wool to the insertion site, while simultaneously removing the needle. In this case, the 'remove needle' action is considered to be of a higher focus than 'apply cotton wool' and so this action is selected as the canonical action. Labelling in this way produces a dataset with each frame uniquely labelled, which is more consistent for the task of checking action orders. Certain steps are repeated throughout a performance, such as hand-washing, and as these actions are conceptually equivalent, we use the same label for each occurrence.

The dataset is recorded using three cameras, each producing a set of recorded RGB-D images for each performance. To ensure consistency across the three viewpoints, we only assign labels for a single camera viewpoint. The algorithm of Section 3.2.1 is used to synchronise frames from the other views to the central camera view during the labelling process, and to assign the labels to these other camera view frames.

The dataset is labelled by a single independent labeller who is asked to follow the guidelines discussed here and the action label descriptions in Appendix K.3.

I.7 Evaluational Method

We must specify the approach for evaluating action recognition approaches against the OSCE dataset. In particular, we must choose the training and testing splitting method. Recent datasets targeted for use by deep learning techniques, such as ImageNet [24], use single splits for training, validation and testing. If multiple splits are used, the training needs to be repeated for each split, and due to the elongated training time of convolutional neural networks on large datasets, this may take an inordinate amount of time for practical training. For such datasets, strategies can be devised to ensure that the test set presents a generalised representation of the data (e.g. similar distributions of classes across training and test sets). Such strategies are difficult to apply to the OSCE dataset. This is due to the decision that we should compose the test set of complete performances by distinct subjects. If this was not the case, it could be argued that test set performance may be artificially high due to observation of a subject perform the same action elsewhere in the training set. As the dataset has 20 subjects and a total of 60 performances, to best characterise the generalisability of approaches, a cross-validation method should be used despite the consequent elongated training time.

For the dataset, we propose five test folds, with each fold composed of 12 performances across four subjects. The selection of performances for each fold is based on the time of recording, i.e. first 12 recorded performances in the first test split, second

12 in the second split, and so on. To calculate the metrics for each fold, we concatenate the predicted actions of the 12 performances and calculate the metrics across this set using the multi-class approach of Section 3.3.4. This approach has precedent in the 50 Salads dataset and is based on recommendations for such evaluation scenarios [323].

I.8 Dataset Uses

The design of this dataset differentiates it from other fine-grained action recognition datasets, such as 50 Salads [232]. The differences include the recording from multiple viewpoints and the novelty of the target application domain. Here we detail how these differentiating factors can be used for possible future research purposes.

The availability of image data from multiple viewpoints opens up the possibility of developing a recognition solution that generalises across the variable of camera placement. This could be performed, for example, by merging the data from the three vantage points, as in Section 3.2.2, and using meshing techniques, such as Poisson surface reconstruction [367], to construct a mesh that can be used to generate new viewpoints for each frame. Using a varied set of these new viewpoints, a level of robustness to camera placement could possibly be added to applicable action recognition techniques. As part of the dataset, we include the relative camera transformations, estimated using the technique of Section 3.2.1, to facilitate such an approach.

Much of the existing fine-grained action recognition datasets focus on the problem of recognising actions involved in everyday activities, such as Activities of Daily Living (ADL) [232, 93, 368]. Reliable recognition of ADL actions may provide the technical basis for situational support systems for infirm or cognitively-impaired persons. However, a significant alternative application direction, involving the recognition of fine-grained actions, is expert supervision of procedure-driven activities. By selecting an OSCE performance as one such activity, it is our intention that this dataset encourages further research in this application direction.

Due to the importance of correct technique and the following of procedure in the work, previous research has been carried out on performance assessment of clinical skills [369, 370]. The skills focussed on in these works were skills that are part of robotic surgery and laparoscopic procedures. These skills, such as suturing and knot-tying, are technical in nature, requiring dexterous and accurate manipulation of implements for correct performance. A framework for the evaluation of a surgeon's skill level in performing these skills exists, known as Objective Structured Assessments of Technical Skills (OSATS) [371]. The presence of a similar evaluation framework for a broader set of clinical skills (OSCEs) warrants the investigation of techniques for the automated evaluation of these skills in a manner similar to that of OSATS. With this dataset, we wish to encourage a broadening of such research into evaluation of technical procedural skills beyond the limited scenarios of surgical skills.

The OSCE dataset is labelled at the level of fine-grained actions as it corresponds closely to a human-level description of the task. This level of granularity is suitable for the problem of recognition of the order in which a task was performed. Should the need arise, it would be possible to break down these action labels into constituent parts of finer granularity. These finer granularity labels would correspond to fundamental action units, such as picking up and placing items. As the boundaries between different actions have been identified by such events as part of our labelling, these boundaries could be naturally subdivided to create a finer granularity. Possible applications of labelling at this granularity may include identifying the temporal segment within which a particularly important action unit is performed, such as piercing the skin with the needle, so that it can undergo further analysis.

The dataset is labelled for the purpose of classification of actions, so that an application could be developed that checks whether the correct order was followed. There are possible other methods of labelling the dataset for other application purposes. Other possible labellings include anomaly detection and ranking. An anomaly in this case may include instances where the subject touches the insertion site after cleaning or touches the sterile needle at any point, which is against the guidelines of Appendix K.1. Detection of such anomalies would be pertinent to a possible expert supervisor system for OSCE performances. A ranking labelling may also be beneficial for such a system, where a perceived level of expertise can be given as feedback to a student using the system.

The evaluation procedure outlined previously is based on the recognition of individual frames or short clips. However, depending on the class of action recognition problem, other evaluation criteria may be more relevant. For example, metrics such as coverage are used for the problem of action segmentation [19]. This problem is often posed as, given an ordered list of action annotations and an untrimmed video, can the entire video be annotated with the transitions between actions correctly identified in time. The labelling of our dataset permits such evaluations in addition to our evaluation methodology.

The unique factors discussed earlier, and possible future use cases, make this dataset a significant contribution to the state-of-the-art.

Appendix J

WeaveNet: Late Fusion Parameter Study

The WeaveNet-LF architecture uses the final outputs of Strand Blocks of the WeaveNet architecture to perform a late fusion between the two branches of the network. In the design (Figure 5.5), we combine the two branches (referred to as the main branch and strands branch) by first compressing the outputs with average pooling, learning a linear transformation of the features of each branch, and combining by concatenation. The selection of hyperparameters associated with the linear transformation layers may have a large effect upon classification results. Linear layers are prone to overfitting and so using different hyperparameters for the strands branch may improve results. To test this, we increase the dropout amount and the number of units in the linear transformation in the strands branch. Increasing the number of linear units gives greater precedence to the outputs of the strands branch to the final softmax classification layer. The values used for each hyperparameter are shown in Table J.1.

Name	*Late Fusion Dropout*	*Late Fusion No. Linear Units*
LF1	0.1	32
LF2	0.2	64

Table J.1: The two hyperparameters selections for late fusion. In LF2, units in the linear layer in the late fusion branch are increased to allow more complex features to be extracted. The dropout rate is also increased to discourage feature co-adaptation. The dropout and units in the linear layer of the main branch are fixed for both experiments.

To fairly compare the effect of these sets of hyperparameters, we train the WeaveNet-LF architectures for all five splits of the 50 Salads dataset. This is necessary as the two sets of hyperparameters may show varying results across the test splits.

The results in Table J.2 show that the LF2 hyperparameter configuration performs better. The importance of selecting appropriate hyperparameters is highlighted by the 2% increase in accuracy between the LF1 accuracy and LF2 accuracy over the five splits of the 50 Salads datasets. Further performance improvements could be found

Training Arrangement	*Accuracy*	*Precision*	*Recall*	F_1
LF1	80.29	75.58	74.94	74.31
LF1 (ensemble)	82.29	77.41	77.09	76.42
LF2	82.02	77.98	74.74	75.46
LF2 (ensemble)	**83.70**	**79.50**	**77.52**	**77.34**

Table J.2: Classification results as percentages for the two hyperparameter arrangements across all splits of the 50 Salads dataset. Accuracy refers to the proportion of correctly classified individual frames. Precision and recall are calculated for each of the ten classes, and the unweighted mean across classes is reported. We report the final model accuracy (at mini-epoch 310) as well as the ensemble produced by the cosine annealing snapshots.

through optimising these (and other) hyperparameters further. However, the need to perform cross validation makes an exhaustive exploration infeasible on reasonable hardware (e.g. three days training for each cross-validation on an Nvidia Titan V).

Appendix K

OSCE-V Dataset Collection Details

Details regarding the collection of the OSCE-V dataset are included in this appendix. Specifically, the steps of the procedure that a subject must attempt to follow during an experiment are listed, as well as the information sheet given to the subject prior to the experiment. Also included is the description of the action labels used during the labelling of the dataset.

K.1 OSCE Steps - Venepuncture

The following is the list of steps that were given to subjects to follow during recording of the Venepuncture OSCE performance. The steps are based on current WHO guidelines for the procedure [366], tailored towards the equipment utilised at the recording location.

Equipment Required

The following equipment will be needed:

(i) Procedure tray;
(ii) Hand wash;
(iii) Non-sterile gloves;
(iv) Tourniquet;
(v) Blood sampling device;
(vi) Blood specimen tubes;
(vii) Tape;
(viii) Alcohol swabs;
(ix) Cotton wool;
(x) Sharps container;
(xi) Transportation bag;
(xii) A ball-point pen.
(xiii) Sharps bin.

Prepare Equipment

1. Use the hand wash gel to wash your hands.
2. Check ID of the patient.
3. Clean the procedure tray using an alcohol wipe.
4. Gather the following items into the tray:
   (a) Tourniquet;
   (b) Cotton wool;
   (c) Blood collection device;
   (d) Sample tubes;
   (e) Alcohol swabs;
   (f) Tape;
   (g) Non-sterile gloves;

Identify Vein

1. Wash hands again using the alcohol gel.
2. Ensure that the simulated hand is in the correct position for the procedure. For a patient, you should be able to access it easily and it should be in a comfortable, resting, stable position.
3. Apply the tourniquet 6-7cm above the joint.
4. Identify an appropriate vein.

Venepuncture Site Preparation

1. Wash hands again using the alcohol gel.
2. Put on gloves.
3. Use an alcohol swab to thoroughly clean the site. Start from the centre and using a circular pattern work outwards to cover 5cm diameter. Note: do not touch the site again. The site should be given 30 seconds to dry.

Venepuncture Procedure

1. Remove the sheath from the needle.
2. Fix the skin with other hand, close to the chosen site, and advance the needle forward, bevel side upwards, at approximately 20 degrees to the skin with a steady motion.
3. Lower and anchor the needle to the simulate arm.
4. Push sample tube into holder, puncturing the diaphragm of the stopper, when blood flow is established.
5. After the first sample tube starts to fill, reduce the pressure of the tourniquet.
6. Fill sample tube to the line, remove and invert gently 4-5 times.
7. Remove the tourniquet fully.

8. Apply gauze over the needle site but do not apply pressure.
9. Remove the needle, locking it into the needle guard, and dispose of it directly in the sharps bin.
10. Apply a plaster to the site.

Finalising the procedure

1. Complete the patient details on the blood sample tubes and document.
2. Seal the sample tubes in an appropriate sample transportation bag.
3. Dispose of all of the clinical waste in the bin provided, including gloves.
4. Clean the dish used with the alcohol swabs.
5. Wash hands using alcohol gel.

K.2 Participant Information Sheet

The following is the content of the information sheet used to brief subjects for the Venepuncture OSCE performance.

Objective Structured Clinical Examination Performance Assessment

We are looking for **adult** participants only. In this experiment you will be asked to remain within the experimental venue.

This experiment involves performing a Objective Structured Clinical Examination (OSCE) task on a simulated set-up. The procedure involves the utilisation of a sterile medical needle. As this may pose a danger of injury we ask that due care is used in handling the needle during the experiment. The experiment will take place under the supervision of the Clinical Skills Tutor from the School of Medicine who will be available to answer any questions at any point during the experiment. There will be a first aid kit available for use in the potential case of injury. The contact details are available of who to contact in the unlikely event that you suffer any physical harm. **You are reminded that you are free to withdraw from this study at any time without any penalty**. If you have any questions at any time during the experiment, the experimenter will be available to answer them.

The experiment will be broken up into the following steps:

1. Firstly, you will be given a list of the steps involved in the procedure and a video of the performance being performed correctly. You are free to ask any questions about the procedure of the experimenter. There will be 8 minutes available to read these steps. You will then attempt to perform the procedure,

the experimenter will call out the steps for you as you are performing them. You are free to ask any questions during this performance. At the end of this performance, the experimenter will reset the area.
2. You will be given four minutes to read back over the steps and view the demonstration video. You are asked to attempt to memorise, as much as possible, the steps. However, it will be possible to ask the experimenter for any help if you get stuck. You will be asked to perform the procedure again.
3. You will be asked to repeat step 2 a further time.

Each of the above performances will be recorded by three mounted RGB-D cameras.

Before you begin the experiment, you will be asked to carefully read the informed consent and sign if you agree to continue with the experiment. The data we collect will be analysed together with the data collected from other participants, and generalised results and conclusions drawn from these experiments will be submitted for publication at conferences and/or scientific journals. Your name will not be used in any report or article.

The videos taken are intended to be used to develop techniques to understand task related interactions with objects. In order for these techniques to be published it may be necessary to display images/samples from these collected videos. Also, as these videos will represent a valuable resource to the research community to test relevant techniques, there may be reason to publish these videos online as a dataset. The exhaustive list of occasions of replay of these videos is as follows:

- As part of research publications;
- As part of lectures (whether academic or informal);
- As part of demonstrations of software systems that attempt to understand and classify the performance of the OCSE skill based on the recorded videos.
- As part of a dataset that will be released to members of the research community. Access to this dataset will be via gv2.cs.tcd.ie domain. Access will be granted to researchers who submit details including their name, research institution and research interests. The dataset will be made available for non-commercial purposes only, and any access to the dataset will be granted on this basis. The license agreement that will be used will be Creative Commons Attribution - Non-Commercial 4.0 International Public License.

If you or any of your family has a history of epilepsy, you are proceeding at your own risk. In an extremely unlikely event that illicit activity is reported during the study, these will be reported to appropriate authorities.

You are also free to withdraw from this study at any time without any penalty. Whilst the recording is ongoing, you may request that all recorded data pertaining to your

participation be deleted. Once the experiment is over, this request will no longer be possible as the data will not be linked to your identity and so can not be removed. Furthermore, once the data has been made available to other researchers, it will not be possible to remove your data from all copies of the data. Your identity will not be recorded as part of this experiment. Should a recording contain information that may potentially reveal your identity, it will be removed from the data collected, and will not be used for research or publication purposes.

The experiment will last approximately 45 minutes. Your participation in this study is voluntary.

If you have any further questions, please do not hesitate to ask the experimenter.

K.3 OSCE-V Action Labels

The following instructions for labelling individual actions were given to the independent labeller of the OSCE-V dataset.

Wash hands

Description: The participant is required to wash their hands with alcohol gel.
Begins: The participant touches the alcohol gel bottle.
Ends: The participant's hands are no longer touching as part of the hand washing.

Check ID

Description: The participant is required to check the ID of the patient to ensure the procedure is performed on the correct patient.
Begins: The participant touches the ID on the patient's wrist, or alternatively, touches the patient's arm to position the wrist to allow reading of the ID.
Ends: The participant releases the patient's ID or arm.

Clean procedure tray

Description: The participant must clean the procedure tray using an antiseptic wipe.
Begins: The participant touches the tray or the wipe to begin cleaning.
Ends: The participant disposes of the antiseptic wipe.

Gather equipment

Description: The participant simulates collecting the required equipment to complete the procedure, the equipment tray is filled out of shot from the camera.
Begins: The participant picks up the empty procedure tray.
Ends: The participant puts down the full procedure tray on the table.

Move arm into position

Description: The participant must adjust the arm so that it is in a comfortable position for the patient and for access.

Begins: The participant touches the patient's arm to move it into position.

Ends: The participant releases the patient's arm, having moved it into position.

Apply tourniquet

Description: A tourniquet is applied around the patient's upper arm to increase pressure for obtaining the blood sample.

Begins: The participant picks up the tourniquet from the procedure tray.

Ends: The participant lets go of the tourniquet, having locked it in position using the plastic toggle.

Identify suitable vein

Description: The participant must physically inspect the patient's arm to identify a suitable vein to collect a blood sample.

Begins: The participant touches an intended location for venepuncture.

Ends: The participant finishes touching the venepuncture location.

Put on gloves

Description: The participant must put on non-sterile gloves before correctly completing the venepuncture procedure.

Begins: The participant touches the non-sterile gloves or the box containing the gloves.

Ends: The participant has gloves on both hands and is no longer adjusting the gloves in any way.

Prepare equipment

Description: The participant must place the equipment in the equipment tray and ensure that the sealed needle box is open.

Begins: The participant picks up any item in the equipment tray.

Ends: The participant stops interacting with the items in the equipment tray.

Swab insertion site

Description: This participant must clean the insertion site on the patient's arm with a sterile alcohol swab.

Begins: The participant begins to pull the swab from its packet.

Ends: The swab is no longer in contact with the patient's arm and the participant's hand is moving away from the injection site.

Prepare needle

Description: The participant must remove the needle from its packet, screw the specimen bottle holder onto the tube connected to the needle, and remove the needle sheath.

Begins: The participant lifts the needle from its packaging.

Ends: The participant moves to position the needle at the injection site once the needle sheath is removed.

Insert needle

Description: The participant must insert the needle into the patient's vein.

Begins: The participant moves to position the needle towards the injection site.

Ends: The participant is holding the needle to the injection site and the needle tip is no longer visible.

Draw blood

Description: The participant must draw blood from the injection site into a specimen bottle.

Begins: The participant picks up the specimen bottle, while the needle is still inserted into the patient's arm.

Ends: The participant has removed the specimen bottle and it is no longer connected to the needle.

Jostle bottle

Description: The participant must invert the blood specimen bottle several times, once a specimen is extracted from the patient's arm.

Begins: The participant starts to invert the blood specimen bottle, while the bottle is detached from the needle tube.

Ends: The participant is no longer inverting the blood sample bottle.

Open tourniquet

Description: The participant must release the tourniquet from the patient's arm to decrease blood pressure.

Begins: The participant's hands are in contact with the plastic toggle of the tourniquet.

Ends: The participant hands are not in contact with the tourniquet and it is completely released from the patient's arm.

Apply cotton wool to site

Description: The participant must place cotton wool on the injection site before removing the needle to stop bleeding.

Begins: The participant picks up the cotton wool, while the needle is in the patient's arm.

Ends: The cotton wool has been placed in contact with the patient's arm at the injection site.

Extract needle

Description: The participant must carefully and safely remove the needle from the patient's arm.

Begins: The participant begins to pull the needle out of the injection site on the patient's arm .

Ends: The needle is no longer in contact with the patient's arm and the participant presses the safety button to retract the needle tip.

Dispose of needle hazard

Description: The participant must dispose of the used needle in a sharp hazard container to ensure the safe disposal of the object.

Begins: The participant has pressed the safety button to retract the needle tip.

Ends: The participant releases the needle into the sharp hazard container.

Tape cotton wool to site

Description: The participant must apply tape to secure the cotton wool to the injection site.

Begins: The participant picks up the tape from the equipment tray.

Ends: The tape is applied to the patient's arm and the participant is no longer touching the tape.

Document sample bottle

Description: The participant must write the patient's name and details on the blood specimen bottle.

Begins: The participant's pen touches the blood specimen bottle.

Ends: The participant finishes writing on the specimen bottle and moves pen away.

Document specimen sheet

Description: The participant must write the patient's name and details on a specimen sheet that must be placed with the blood specimen bottle in a specimen bag.

Begins: The participant's pen touches the specimen sheet.

Ends: The participant finishes writing on the specimen sheet and moves pen away.

Seal specimen in specimen bag

Description: This participant must seal the blood specimen bottle in a plastic specimen bag.
Begins: The participant touches the specimen bag and the blood specimen bottle.
Ends: The participant no longer touches the specimen bag, and the specimen sheet and specimen bottle are inside the bag.

Dispose of waste

Description: The participant places waste material in a bin placed to the side of the station.
Begins: The participant holds waste material and their hand begins to outstretch towards the bin.
Ends: The participant has released the waste material in their hands.

Remove gloves

Description: The participant must remove the non-sterile gloves from both of their hands.
Begins: The participant begins to pull the gloves off either of their hands.
Ends: The gloves are completely removed from the participants hands.

Background

Description: Any action carried out by the participant that does not fall into any of the listed categories is labelled as background.
Begins: Not applicable.
Ends: Not applicable.

Bibliography

[1] A. Gupta, A. Kembhavi, and L. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1775–1789, 2009.

[2] B. Yao and L. Fei-Fei, "Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1691–1703, 2012.

[3] G. Gkioxari, R. Girshick, P. Dollár, and K. He, "Detecting and recognizing human-object interactions," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8359–8367, 2018.

[4] R. M. Harden, "What is an OSCE?," *Medical Teacher*, vol. 10, no. 1, pp. 19–22, 1988.

[5] K. Z. Khan, S. Ramachandran, K. Gaunt, and P. Pushkar, "The objective structured clinical examination (OSCE): AMEE guide no. 81. part I: An historical and theoretical perspective," *Medical Teacher*, vol. 35, no. 9, pp. e1437–e1446, 2013.

[6] I. Lavery and P. Ingram, "Prevention of infection in peripheral intravenous devices," *Nursing Standard (Royal College of Nursing (Great Britain): 1987)*, vol. 20, no. 49, pp. 49–56, 2006.

[7] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager, "CoSTAR: Instructing collaborative robots with behavior trees and vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 564–571, 2017.

[8] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[9] D. Tang, H. Chang, A. Tejani, and T. K. Kim, "Latent regression forest: Structured estimation of 3d hand poses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[10] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[11] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 816–833, Springer, 2016.

[12] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 409–419, 2018.

[13] Y. Kong and Y. Fu, "Bilinear heterogeneous information machine for RGB-D action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1054–1062, 2015.

[14] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, "Large-scale isolated gesture recognition using convolutional neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 7–12, 2016.

[15] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4207–4215, 2016.

[16] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems 27*, pp. 568–576, 2014.

[17] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, 2017.

[18] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense RGB-D scene flow," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 98–104, 2015.

[19] C. Lea, A. Reiter, R. Vidal, and G. D. Hager, "Segmental spatiotemporal CNNs for fine-grained action segmentation," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 36–52, Springer, 2016.

[20] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1003–1012, 2017.

[21] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 754–763, 2017.

[22] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. NJ, USA: Springer, 2006.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[25] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 3320–3328, 2014.

[26] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, 2014.

[27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, pp. 2048–2057, 2015.

[28] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5168–5177, 2017.

[29] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*, pp. 153–160, 2007.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008.

[31] A. R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[33] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609–616, 2009.

[34] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, pp. 448–456, 2015.

[38] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556 [cs]*, 2014.

[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[41] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv:1312.6120 [cond-mat, q-bio, stat]*, 2013.

[42] D. Mishkin and J. Matas, "All you need is a good init," in *International Conference on Learning Representations*, 2016.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[44] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814, 2010.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," (MA, USA), pp. 696–699, MIT Press, 1988.

[46] A. Graves, "Supervised sequence labelling," in *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 5–13, Springer, 2012.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 1. MA, USA: MIT Press, 2016.

[48] L. K. Li, "Approximation theory and recurrent networks," in *IJCNN International Joint Conference on Neural Networks*, vol. 2, pp. 266–271 vol.2, 1992.

[49] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, no. 1, pp. 107–123, 2000.

[50] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997.

[52] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computing*, vol. 12, no. 10, pp. 2451–2471, 2000.

[53] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, pp. II–1764–II–1772, 2014.

[54] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137, 2015.

[55] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández, "Unconstrained on-line handwriting recognition with recurrent neural networks," in *Advances in Neural Information Processing Systems 20*, pp. 577–584, 2008.

[56] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, pp. 3104–3112, 2014.

[57] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv:1704.01444 [cs]*, 2017.

[58] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

[59] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

[60] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, pp. 2342–2350, 2015.

[61] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv:1611.01578 [cs]*, 2016.

[62] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv:1410.5401 [cs]*, vol. 1410.540, 2014.

[63] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[64] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, 2014.

[65] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[66] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang, "Robust single image super-resolution via deep networks with sparse prior," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3194–3207, 2016.

[67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 630–645, Springer, 2016.

[69] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[70] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 646–661, Springer, 2016.

[71] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *arXiv:1206.5538 [cs]*, 2012.

[72] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv:1505.00387 [cs]*, 2015.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[74] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI'17 Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, vol. 4, p. 12, 2017.

[75] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in Neural Information Processing Systems 29*, pp. 550–558, 2016.

[76] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, pp. 87.1–87.12, 2016.

[77] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992.*, pp. 379–385, 1992.

[78] J. Yang, Y. Xu, and C. S. Chen, "Human action learning via hidden markov model," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 27, no. 1, pp. 34–44, 1997.

[79] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, 2008.

[80] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pp. 2929–2936, 2009.

[81] G. Gkioxari, R. Girshick, and J. Malik, "Actions and attributes from wholes and parts," *arXiv:1412.2604 [cs]*, 2014.

[82] M. Sadeghi and A. Farhadi, "Recognition using visual phrases," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1745–1752, 2011.

[83] N. T. Siebel and S. J. Maybank, "The advisor visual surveillance system," in *ECCV 2004 workshop Applications of Computer Vision (ACV)*, 2004.

[84] L. Chen, C. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 961–974, 2012.

[85] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[86] D. Newtson, G. A. Engquist, and J. Bois, "The objective basis of behavior units," *Journal of Personality and Social Psychology*, vol. 35, no. 12, pp. 847–862, 1977.

[87] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys*, vol. 43, no. 3, pp. 16:1–16:43, 2011.

[88] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A review of human activity recognition methods," *Frontiers in Robotics and AI*, vol. 2, 2015.

[89] G. Gkioxari and J. Malik, "Finding action tubes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 759–768, 2015.

[90] Z. Shou, D. Wang, and S. F. Chang, "Temporal action localization in untrimmed videos via multi-stage CNNs," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1049–1058, 2016.

[91] D. Han, L. Bo, and C. Sminchisescu, "Selection and context for action recognition," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 1933–1940, 2009.

[92] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2641–2648, 2014.

[93] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1194–1201, 2012.

[94] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, pp. 3551–3558, 2013.

[95] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3169–3176, 2011.

[96] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pp. 1–8, 2008.

[97] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 1, pp. 886–893, 2005.

[98] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Computer Vision - ECCV 2006*, Lecture Notes in Computer Science, pp. 428–441, Springer, 2006.

[99] B. Packer, K. Saenko, and D. Koller, "A combined pose, object, and feature model for action understanding," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1378–1385, 2012.

[100] V. Singh and R. Nevatia, "Action recognition in cluttered dynamic scenes using pose-specific part models," in *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 113–120, 2011.

[101] C. Rao, A. Yilmaz, and M. Shah, "View-invariant representation and recognition of actions," *International Journal of Computer Vision*, vol. 50, no. 2, pp. 203–226, 2002.

[102] L. Wang, Y. Qiao, and X. Tang, "Video action detection with relational dynamic-poselets," in *Computer Vision - ECCV 2014*, Lecture Notes in Computer Science, pp. 565–580, Springer, 2014.

[103] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pp. 2004–2011, 2009.

[104] A. Prest, V. Ferrari, and C. Schmid, "Explicit modeling of human-object interactions in realistic videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 835–848, 2013.

[105] M. Meng, H. Drira, M. Daoudi, and J. Boonaert, "Human object interaction recognition using rate-invariant shape analysis of inter joint distances trajectories," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 999–1004, 2016.

[106] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 65–72, 2005.

[107] A. Gilbert, J. Illingworth, and R. Bowden, "Scale invariant action recognition using compound features mined from dense spatio-temporal corners," in *Computer Vision - ECCV 2008*, no. 5302 in Lecture Notes in Computer Science, pp. 222–233, Springer, 2008.

[108] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[109] M. Bregonzio, S. Gong, and T. Xiang, "Recognising action as clouds of space-time interest points," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pp. 1948–1955, 2009.

[110] G. Willems, T. Tuytelaars, and L. V. Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *Computer Vision - ECCV 2008*, Lecture Notes in Computer Science, pp. 650–663, Springer, 2008.

[111] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[112] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, vol. 15, p. 50, 1988.

[113] A. Klaeser, M. Marszalek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *Proceedings of the British Machine Vision Conference*, pp. 99.1–99.10, 2008.

[114] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 3, pp. 32–36 Vol.3, 2004.

[115] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *Proceedings of the British Machine Vision Conference*, pp. 124.1–124.11, 2009.

[116] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, "Beyond gaussian pyramid: Multi-skip feature stacking for action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 204–212, 2015.

[117] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors," in *Computer Vision - ECCV 2014*, Lecture Notes in Computer Science, pp. 581–595, Springer, 2014.

[118] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4305–4314, 2015.

[119] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu, "Action recognition with actons," in *2013 IEEE International Conference on Computer Vision*, pp. 3559–3566, 2013.

[120] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1234–1241, 2012.

[121] I. Lillo, J. C. Niebles, and A. Soto, "A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1981–1990, 2016.

[122] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian, "Interaction part mining: A mid-level approach for fine-grained action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3323–3331, 2015.

[123] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Computer Vision - ECCV 2010*, Lecture Notes in Computer Science, pp. 140–153, Springer, 2010.

[124] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan, "DL-SFA: Deeply-learned slow feature analysis for action recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2632, 2014.

[125] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[126] L. Wang, Y. Qiao, X. Tang, and L. V. Gool, "Actionness estimation using hybrid fully convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2708–2717, 2016.

[127] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[128] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4694–4702, 2015.

[129] Z. Li, E. Gavves, M. Jain, and C. G. M. Snoek, "VideoLSTM convolves, attends and flows for action recognition," *arXiv:1607.01794 [cs]*, 2016.

[130] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," in *International Conference on Learning Representations*, 2016.

[131] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *arXiv:1506.04214 [cs]*, 2015.

[132] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[133] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding*, Lecture Notes in Computer Science, pp. 29–39, Springer, 2011.

[134] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[135] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.

[136] L. Sun, K. Jia, D. Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4597–4605, 2015.

[137] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1933–1941, 2016.

[138] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Advances in Neural Information Processing Systems 29*, pp. 3468–3476, 2016.

[139] B. Fernando, E. Gavves, J. O. M, A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[140] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3034–3042, 2016.

[141] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6201–6210, 2019.

[142] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 20–36, Springer, 2016.

[143] X. Wang and A. Gupta, "Videos as Space-Time Region Graphs," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 413–431, Springer, 2018.

[144] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal Relational Reasoning in Videos," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 831–846, Springer, 2018.

[145] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in neural information processing systems*, pp. 4967–4976, 2017.

[146] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick, "Long-Term Feature Banks for Detailed Video Understanding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 284–293, 2019.

[147] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local Neural Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

[148] J. Lin, C. Gan, and S. Han, "TSM: Temporal Shift Module for Efficient Video Understanding," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7082–7092, 2019.

[149] B. Korbar, D. Tran, and L. Torresani, "SCSampler: Sampling Salient Clips From Video for Efficient Action Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6231–6241, 2019.

[150] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "AdaFrame: Adaptive Frame Selection for Fast Video Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1278–1287, 2019.

[151] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-Relation Networks for Video Classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1430–1439, 2018.

[152] Q. Fan, C.-F. R. Chen, H. Kuehne, M. Pistoia, and D. Cox, "More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation," in *Advances in Neural Information Processing Systems*, pp. 2264–2273, 2019.

[153] P. Wang, Z. Li, Y. Hou, and W. Li, "Action recognition based on joint trajectory maps using convolutional neural networks," in *Proceedings of the 2016 ACM on Multimedia Conference*, pp. 102–106, 2016.

[154] S. Gupta, J. Hoffman, and J. Malik, "Cross Modal Distillation for Supervision Transfer," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2827–2836, 2016.

[155] N. C. Garcia, P. Morerio, and V. Murino, "Modality Distillation with Multiple Stream Networks for Action Recognition," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 106–121, Springer, 2018.

[156] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid, "MARS: Motion-Augmented RGB Stream for Action Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7874–7883, 2019.

[157] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural networks*, vol. 22, no. 5-6, pp. 544–557, 2009.

[158] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," *arXiv:1511.03643 [cs, stat]*, 2016.

[159] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[160] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[161] S. Bhardwaj, M. Srinivasan, and M. M. Khapra, "Efficient Video Classification Using Fewer Frames," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 354–363, 2019.

[162] Z. Luo, J.-T. Hsieh, L. Jiang, J. C. Niebles, and L. Fei-Fei, "Graph Distillation for Action Detection with Privileged Modalities," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 174–192, Springer, 2018.

[163] N. C. Garcia, P. Morerio, and V. Murino, "Learning with Privileged Information via Adversarial Discriminative Modality Distillation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2581–2593, 2020.

[164] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv:1411.1784 [cs, stat]*, 2014.

[165] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations*, 2018.

[166] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," in *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pp. 649–666, Springer, 2016.

[167] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1422–1430, 2015.

[168] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain Generalization by Solving Jigsaw Puzzles," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2224–2233, 2019.

[169] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-Supervised Video Representation Learning with Odd-One-Out Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5729–5738, 2017.

[170] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman, "Learning and Using the Arrow of Time," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8052–8060, 2018.

[171] R. Arandjelović and A. Zisserman, "Objects that Sound," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 451–466, Springer, 2018.

[172] R. Arandjelovic and A. Zisserman, "Look, Listen and Learn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 609–617, 2017.

[173] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization," in *Advances in Neural Information Processing Systems*, pp. 7763–7774, 2018.

[174] A. Owens and A. A. Efros, "Audio-Visual Scene Analysis with Self-Supervised Multisensory Features," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 639–658, Springer, 2018.

[175] J. Munro and D. Damen, "Multi-modal domain adaptation for fine-grained action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 122–132, 2020.

[176] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling Egocentric Vision: The EPIC-Kitchens Dataset," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 753–771, Springer, 2018.

[177] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1093–1096, 2011.

[178] P. Trindade, J. Lobo, and J. Barreto, "Hand gesture recognition using color and depth images enhanced with hand angular pose data," in *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 71–76, 2012.

[179] Y. Yang, I. Saleemi, and M. Shah, "Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1635–1648, 2013.

[180] E. Ohn-Bar and M. M. Trivedi, "Joint angles similarities and hog2 for action recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 465–470, 2013.

[181] X. Yang and Y. Tian, "Super normal vector for activity recognition using depth sequences," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 804–811, 2014.

[182] O. Oreifej and Z. Liu, "HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences," in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 716–723, 2013.

[183] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 821–826, IEEE, 2011.

[184] Y. Kong, B. Satarboroujeni, and Y. Fu, "Hierarchical 3D kernel descriptors for action recognition using depth sequences," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1, pp. 1–6, 2015.

[185] A. Yao, J. Gall, G. Fanelli, and L. V. Gool, "Does human action recognition benefit from pose estimation?," in *Proceedings of the British Machine Vision Conference*, pp. 67.1–67.11, 2011.

[186] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black, "Towards understanding action recognition," in *2013 IEEE International Conference on Computer Vision (ICCV)*, pp. 3192–3199, 2013.

[187] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from RGBD images," in *2012 IEEE International Conference on Robotics and Automation*, pp. 842–849, 2012.

[188] L. Xia, C. C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27, 2012.

[189] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. M. Campos, "On the improvement of human action recognition from depth map sequences using space-time occupancy patterns," *Pattern Recognition Letters*, vol. 36, pp. 221–227, 2014.

[190] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu, "Robust 3D action recognition with random occupancy patterns," in *Computer Vision - ECCV 2012*, Lecture Notes in Computer Science, pp. 872–885, Springer, 2012.

[191] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. M. Campos, "STOP: Space-time occupancy patterns for 3D action recognition from depth map sequences," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Lecture Notes in Computer Science, pp. 252–259, Springer, 2012.

[192] A. Eweiwi, M. S. Cheema, C. Bauckhage, and J. Gall, "Efficient pose-based action recognition," in *Computer Vision – ACCV 2014*, no. 9007 in Lecture Notes in Computer Science, pp. 428–443, Springer, 2014.

[193] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection," in *2013 IEEE International Conference on Computer Vision*, pp. 2752–2759, 2013.

[194] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a lie group," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 588–595, 2014.

[195] C. Wan, A. Yao, and L. Van Gool, "Hand Pose Estimation from Local Surface Normals," in *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pp. 554–569, Springer, 2016.

[196] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests," in *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pp. 852–863, Springer, 2012.

[197] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks," *ACM Transactions on Graphics*, vol. 33, no. 5, pp. 169:1–169:10, 2014.

[198] J. S. Supancic, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-Based Hand Pose Estimation: Data, Methods, and Challenges," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1868–1876, 2015.

[199] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 824–832, 2015.

[200] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2605–2613, 2017.

[201] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3593–3601, 2016.

[202] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3D Convolutional Neural Networks for Efficient and Robust Hand Pose Estimation from Single Depth Images," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5679–5688, 2017.

[203] J. Y. Chang, G. Moon, and K. M. Lee, "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5079–5088, 2018.

[204] S. Yuan, Q. Ye, G. Garcia-Hernando, and T.-K. Kim, "The 2017 hands in the million challenge on 3d hand pose estimation," *arXiv:1707.02237 [cs]*, 2017.

[205] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. Argyros, and T.-K. Kim, "Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2636–2645, 2018.

[206] S. Baek, K. I. Kim, and T.-K. Kim, "Augmented Skeleton Space Transfer for Depth-Based Hand Pose Estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8330–8339, 2018.

[207] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand PointNet: 3D Hand Pose Estimation Using Point Sets," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8417–8426, 2018.

[208] C. Wan, T. Probst, L. V. Gool, and A. Yao, "Dense 3D Regression for Hand Pose Estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2018.

[209] Y. Chen, Z. Tu, L. Ge, D. Zhang, R. Chen, and J. Yuan, "SO-HandNet: Self-Organizing Network for 3D Hand Pose Estimation With Semi-Supervised Learning," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6960–6969, 2019.

[210] K. Du, X. Lin, Y. Sun, and X. Ma, "CrossInfoNet: Multi-Task Information Sharing Based Hand Pose Estimation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9888–9897, 2019.

[211] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan, "A2J: Anchor-to-Joint Regression Network for 3D Articulated Pose Estimation From a Single Depth Image," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 793–802, 2019.

[212] L. Yang and A. Yao, "Disentangling Latent Hands for Image Synthesis and Pose Estimation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9869–9878, 2019.

[213] C. Zimmermann and T. Brox, "Learning to Estimate 3D Hand Pose from Single RGB Images," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4913–4921, 2017.

[214] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 49–59, 2018.

[215] A. Boukhayma, R. de Bem, and P. H. Torr, "3D Hand Shape and Pose From Images in the Wild," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10835–10844, 2019.

[216] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3D Hand Shape and Pose Estimation From a Single RGB Image," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10825–10834, 2019.

[217] L. Yang, S. Li, D. Lee, and A. Yao, "Aligning Latent Spaces for 3D Hand Pose Estimation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2335–2343, 2019.

[218] X. Zhang, Q. Li, H. Mo, W. Zhang, and W. Zheng, "End-to-End Hand Mesh Recovery From a Monocular RGB Image," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2354–2364, 2019.

[219] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt, "Real-time joint tracking of a hand manipulating an object from RGB-D input," in *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pp. 294–310, Springer, 2016.

[220] C. Choi, S. H. Yoon, C.-N. Chen, and K. Ramani, "Robust Hand Pose Estimation during the Interaction with an Unknown Object," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3142–3151, 2017.

[221] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-Time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1163–1172, 2017.

[222] S. Baek, K. I. Kim, and T.-K. Kim, "Weakly-Supervised Domain Adaptation via GAN and Mesh Model for Estimating 3D Hand Poses Interacting Objects," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6120–6130, 2020.

[223] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1110–1118, 2015.

[224] V. Veeriah, N. Zhuang, and G. J. Qi, "Differential recurrent neural networks for action recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4041–4049, 2015.

[225] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu, "Online human action detection using joint classification-regression recurrent neural networks," in *Com-*

puter Vision - ECCV 2016, Lecture Notes in Computer Science, pp. 203–220, Springer, 2016.

[226] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona, "Action recognition from depth maps using deep convolutional neural networks," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 4, pp. 498–509, 2016.

[227] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona, "Scene flow to action map: A new representation for RGB-D based action recognition with convolutional neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 416–425, 2017.

[228] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 9–14, 2010.

[229] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall, "Recognizing daily activities with RFID-based sensors," in *Proceedings of the 11th International Conference on Ubiquitous Computing*, pp. 51–60, 2009.

[230] J. Hoey, P. Poupart, A. v. Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis, "Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 503–519, 2010.

[231] J. Hoey, T. Plötz, D. Jackson, A. Monk, C. Pham, and P. Olivier, "Rapid specification and automated generation of prompting systems to assist people with dementia," *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 299–318, 2011.

[232] S. Stein and S. J. McKenna, "Combining embedded accelerometers with computer vision for recognizing food preparation activities," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 729–738, 2013.

[233] A. Khan, S. Mellor, E. Berlin, R. Thompson, R. McNaney, P. Olivier, and T. Plötz, "Beyond activity recognition: Skill assessment from accelerometer data," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1155–1166, 2015.

[234] C. Lavania, S. Thulasidasan, A. LaMarca, J. Scofield, and J. Bilmes, "A weakly supervised activity recognition framework for real-time synthetic biology laboratory assistance," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 37–48, 2016.

[235] C. Wu, J. Zhang, S. Savarese, and A. Saxena, "Watch-n-patch: Unsupervised understanding of actions and relations," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4362–4370, 2015.

[236] S. Stein and S. J. McKenna, "Recognising complex activities with histograms of relative tracklets," *Computer Vision and Image Understanding*, vol. 154, pp. 82–93, 2017.

[237] B. Ni, X. Yang, and S. Gao, "Progressively parsing interactional objects for fine grained action detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1020–1028, 2016.

[238] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8, 2016.

[239] A. Richard and J. Gall, "Temporal action detection using a statistical language model," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3131–3140, 2016.

[240] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1961–1970, 2016.

[241] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6047–6056, 2018.

[242] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding," in *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pp. 510–526, Springer, 2016.

[243] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 17–24, 2010.

[244] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep Learning on Spatio-Temporal Graphs," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5308–5317, 2016.

[245] Z. Deng, A. Vahdat, H. Hu, and G. Mori, "Structure Inference Machines: Recurrent Neural Networks for Analyzing Relations in Group Activity Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4772–4781, 2016.

[246] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[247] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5425–5434, 2017.

[248] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[249] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907 [cs, stat]*, 2017.

[250] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori, "Object Level Visual Reasoning in Videos," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 106–122, Springer, 2018.

[251] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Advances in Neural Information Processing Systems*, pp. 4539–4547, 2017.

[252] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[253] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[254] R. Girdhar, J. João Carreira, C. Doersch, and A. Zisserman, "Video Action Transformer Network," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 244–253, 2019.

[255] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[256] Y. Zhang, P. Tokmakov, M. Hebert, and C. Schmid, "A Structured Model for Action Detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9967–9976, 2019.

[257] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles, "Action Genome: Actions As Compositions of Spatio-Temporal Scene Graphs," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10233–10244, 2020.

[258] J. M. Zacks, B. Tversky, and G. Iyer, "Perceiving, remembering, and communicating structure in events.," *Journal of Experimental Psychology: General*, vol. 130, no. 1, p. 29, 2001.

[259] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.

[260] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, "Motion Feature Network: Fixed Motion Filter for Action Recognition," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 392–408, Springer, 2018.

[261] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "STM: SpatioTemporal and Motion Encoding for Action Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2000–2009, 2019.

[262] A. Piergiovanni and M. S. Ryoo, "Representation Flow for Action Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9937–9945, 2019.

[263] M. Zolfaghari, K. Singh, and T. Brox, "ECO: Efficient Convolutional Network for Online Video Understanding," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 713–730, Springer, 2018.

[264] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed Video Action Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6026–6035, 2018.

[265] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, "MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 449–458, 2018.

[266] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

[267] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification," in *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pp. 318–335, Springer, 2018.

[268] D. Tran, H. Wang, M. Feiszli, and L. Torresani, "Video Classification With Channel-Separated Convolutional Networks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5551–5560, 2019.

[269] S. Sudhakaran, S. Escalera, and O. Lanz, "Gate-Shift Networks for Video Action Recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1099–1108, 2020.

[270] B. M. Martínez, D. Modolo, Y. Xiong, and J. Tighe, "Action Recognition With Spatial-Temporal Discriminative Filter Banks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5481–5490, 2019.

[271] N. Hussein, E. Gavves, and A. W. Smeulders, "Timeception for Complex Action Recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 254–263, 2019.

[272] C. Feichtenhofer, "X3D: Expanding Architectures for Efficient Video Recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 200–210, 2020.

[273] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv:1212.0402 [cs]*, 2012.

[274] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *2011 International Conference on Computer Vision*, pp. 2556–2563, 2011.

[275] Y. Zhou, B. Ni, S. Yan, P. Moulin, and Q. Tian, "Pipelining localized semantic features for fine-grained action recognition," in *Computer Vision - ECCV 2014*, Lecture Notes in Computer Science, pp. 481–496, Springer, 2014.

[276] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2847–2854, 2012.

[277] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh, *et al.*, "JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling," in *Modeling and Monitoring of Computer Assisted Interventions (M2CAI) – MICCAI Workshop*, 1997.

[278] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. B. Haro, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager, "A Dataset and Benchmarks for Segmentation and Recognition of Gestures in Robotic Surgery," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2025–2041, 2017.

[279] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1595–1618, 2017.

[280] C. Rupprecht, C. Lea, F. Tombari, N. Navab, and G. D. Hager, "Sensor substitution for video-based action recognition," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5230–5237, 2016.

[281] C. Lea, G. D. Hager, and R. Vidal, "An Improved Model for Segmentation and Recognition of Fine-Grained Activities with Application to Surgical Training Tasks," in *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 1123–1129, 2015.

[282] M. J. Fard, S. Ameri, R. D. Ellis, R. B. Chinnam, A. K. Pandya, and M. D. Klein, "Automated robot-assisted surgical skill evaluation: Predictive analytics approach," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 14, no. 1, p. e1850, 2018.

[283] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1290–1297, 2012.

[284] J. Wang, X. Nie, Y. Xia, Y. Wu, and S. C. Zhu, "Cross-view action modeling, learning, and recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2649–2656, 2014.

[285] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian, "Histogram of oriented principal components for cross-view action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 12, pp. 2430–2443, 2016.

[286] A. Shahroudy, J. Liu, T. T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1010–1019, 2016.

[287] A. Zia, Y. Sharma, V. Bettadapura, E. L. Sarin, M. A. Clements, and I. Essa, "Automated Assessment of Surgical Skills Using Frequency Analysis," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pp. 430–438, Springer, 2015.

[288] B. B. Oğul, M. F. Gilgien, and P. D. Şahin, "Ranking Robot-Assisted Surgery Skills Using Kinematic Sensors," in *Ambient Intelligence*, Lecture Notes in Computer Science, pp. 330–336, Springer, 2019.

[289] Q. Zhang and B. Li, "Relative Hidden Markov Models for Video-Based Evaluation of Motion Skills in Surgical Training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 6, pp. 1206–1218, 2015.

[290] A. Zia, Y. Sharma, V. Bettadapura, E. L. Sarin, and I. Essa, "Video and accelerometer-based motion analysis for automated surgical skills assessment," *International Journal of Computer Assisted Radiology and Surgery*, vol. 13, no. 3, pp. 443–455, 2018.

[291] A. Malpani, S. S. Vedula, C. C. G. Chen, and G. D. Hager, "Pairwise Comparison-Based Objective Score for Automated Skill Assessment of Segments in a Surgical Task," in *Information Processing in Computer-Assisted Interventions*, Lecture Notes in Computer Science, pp. 138–147, Springer, 2014.

[292] Y. Sharma, T. Plötz, N. Hammerld, S. Mellor, R. McNaney, P. Olivier, S. Deshmukh, A. McCaskie, and I. Essa, "Automated surgical OSATS prediction from videos," in *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pp. 461–464, 2014.

[293] H. Pirsiavash, C. Vondrick, and A. Torralba, "Assessing the Quality of Actions," in *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pp. 556–571, Springer, 2014.

[294] P. Parmar and B. T. Morris, "Learning to Score Olympic Events," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 76–84, 2017.

[295] G. Bertasius, H. S. Park, S. X. Yu, and J. Shi, "Am I a Baller? Basketball Performance Assessment from First-Person Videos," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2196–2204, 2017.

[296] Y. Li, X. Chai, and X. Chen, "ScoringNet: Learning Key Fragment for Action Quality Assessment with Ranking Loss in Skilled Sports," in *Computer Vision – ACCV 2018*, Lecture Notes in Computer Science, pp. 149–164, Springer, 2019.

[297] C. Xu, Y. Fu, B. Zhang, Z. Chen, Y.-G. Jiang, and X. Xue, "Learning to Score Figure Skating Sport Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019.

[298] J.-H. Pan, J. Gao, and W.-S. Zheng, "Action Assessment by Joint Relation Graphs," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6330–6339, 2019.

[299] H. Doughty, D. Damen, and W. Mayol-Cuevas, "Who's Better? Who's Best? Pairwise Deep Ranking for Skill Determination," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6057–6066, 2018.

[300] H. Doughty, W. Mayol-Cuevas, and D. Damen, "The Pros and Cons: Rank-Aware Temporal Attention for Skill Determination in Long Videos," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7854–7863, 2019.

[301] P. Parmar and B. T. Morris, "What and How Well You Performed? A Multitask Learning Approach to Action Quality Assessment," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 304–313, 2019.

[302] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 954–962, 2015.

[303] J. López, D. Pérez, E. Paz, and A. Santana, "WatchBot: A building maintenance and surveillance system based on autonomous robots," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1559–1571, 2013.

[304] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2901–2910, 2019.

[305] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "PointNetGPD: Detecting Grasp Configurations from Point Sets," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3629–3635, 2019.

[306] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11441–11450, 2020.

[307] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, "Scene labeling with LSTM recurrent neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3547–3555, 2015.

[308] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[309] P. Lancaster and K. Salkauskas, "Surfaces generated by moving least squares methods," *Mathematics of Computation*, vol. 37, no. 155, pp. 141–158, 1981.

[310] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[311] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer Graphics Forum*, vol. 26, pp. 214–226, 2007.

[312] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," *KI - Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[313] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*, vol. 2, 2009.

[314] R. Bellman, J. Holland, and R. Kalaba, "On an application of dynamic programming to the synthesis of logical systems," *Journal of the ACM*, vol. 6, no. 4, pp. 486–493, 1959.

[315] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[316] Olah, Christopher, "Understanding LSTM networks – colah's blog." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. [Online; accessed 29-September-2019].

[317] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013.

[318] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, pp. 2047–2052 vol. 4, 2005.

[319] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems 29*, pp. 1019–1027, 2016.

[320] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim, "Shake'n'sense: Reducing interference for overlapping structured light depth cameras," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1933–1936, 2012.

[321] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[322] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pp. 265–283, 2016.

[323] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[324] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems 24*, pp. 2546–2554, 2011.

[325] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[326] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[327] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[328] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[329] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* NY, USA: Springer, 2 ed., 2016.

[330] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints," in *2011 International Conference on Computer Vision*, pp. 2088–2095, 2011.

[331] L. Sifre and P. S. Mallat, *Rigid-Motion Scattering For Image Classification, Ecole Polytechnique, CMAP PhD thesis.* 2014.

[332] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555 [cs]*, 2014.

[333] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, p. 419, 1962.

[334] J. W. Davis and A. F. Bobick, "The representation and recognition of human movement using temporal templates," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pp. 257–267, 1997.

[335] M. A. R. Ahad, *Motion History Images for Action Recognition and Understanding*. SpringerBriefs in Computer Science, London: Springer-Verlag, 2013.

[336] J. Wang, A. Cherian, F. Porikli, and S. Gould, "Action representation using classifier decision boundaries," *arXiv:1704.01716 [cs]*, 2017.

[337] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Point cloud library," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/12, 2012.

[338] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 681–687, 2015.

[339] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[340] H. Gammulle, T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Coupled generative adversarial network for continuous fine-grained action segmentation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 200–209, 2019.

[341] P. Lei and S. Todorovic, "Temporal deformable residual networks for action segmentation in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6742–6751, 2018.

[342] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and less than 0.5mb model size," *arXiv:1602.07360 [cs]*, 2016.

[343] A. W. Bernard, G. Ceccolini, R. Feinn, J. Rockfeld, I. Rosenberg, L. Thomas, and T. Cassese, "Medical students review of formative OSCE scores, checklists, and videos improves with student-faculty debriefing meetings," *Medical Education Online*, vol. 22, no. 1, p. 1324718, 2017.

[344] I. Schleicher, K. Leitner, J. Juenger, A. Moeltner, M. Ruesseler, B. Bender, J. Sterz, K.-F. Schuettler, S. Koenig, and J. G. Kreuder, "Examiner effect on the objective structured clinical exam - a study at five medical schools," *BMC Medical Education*, vol. 17, p. 71, 2017.

[345] D. Hope and H. Cameron, "Examiners are most lenient at the start of a two-day OSCE," *Medical Teacher*, vol. 37, no. 1, pp. 81–85, 2015.

[346] L. Stroud, J. Herold, G. Tomlinson, and R. B. Cavalcanti, "Who you know or what you know? effect of examiner familiarity with residents on OSCE scores," *Academic Medicine*, vol. 86, no. 10, p. S8, 2011.

[347] S. Maloney, M. Storr, P. Morgan, and D. Ilic, "The effect of student self-video of performance on clinical skill competency: a randomised controlled trial," *Advances in Health Sciences Education*, vol. 18, no. 1, pp. 81–89, 2013.

[348] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *, Fourth IEEE Workshop on Applications of Computer Vision, 1998. WACV '98. Proceedings*, pp. 214–219, 1998.

[349] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 858–865, 2011.

[350] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision - ACCV 2012*, Lecture Notes in Computer Science, pp. 548–562, Springer, 2012.

[351] K. Lai, L. Bo, X. Ren, and D. Fox, "RGB-D object recognition: Features, algorithms, and a large scale benchmark," in *Consumer Depth Cameras for Computer Vision*, Advances in Computer Vision and Pattern Recognition, pp. 167–192, Springer, 2013.

[352] E. Auvinet, J. Meunier, and F. Multon, "Multiple depth cameras calibration and body volume reconstruction for gait analysis," in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pp. 478–483, 2012.

[353] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[354] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[355] A. Fitzgibbon, "Robust registration of 2D and 3D point sets," *Image and Vision Computing*, vol. 21, pp. 1145–1153, 2002.

[356] S. J. D. Prince, *Computer Vision: Models, Learning, and Inference*. NY, USA: Cambridge University Press, 1 ed., 2012.

[357] F. Tombari, S. Salti, and L. Di Stefano, "Performance evaluation of 3d keypoint detectors," *International Journal of Computer Vision*, vol. 102, no. 1-3, pp. 198–220, 2012.

[358] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[359] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *, 1991 IEEE International Conference on Robotics and Automation, 1991. Proceedings*, pp. 2724–2729 vol.3, 1991.

[360] A.-M. Simundic, M. Cornes, K. Grankvist, G. Lippi, M. Nybo, S. Kovalevskaya, L. Sprongl, Z. Sumarac, and S. Church, "Survey of national guidelines, education and training on phlebotomy in 28 european countries: an original report by the european federation of clinical chemistry and laboratory medicine (EFLM) working group for the preanalytical phase (WG-PA)," *Clinical Chemistry and Laboratory Medicine*, vol. 51, no. 8, pp. 1585–1593, 2013.

[361] S. S. Dychter, D. A. Gold, D. Carson, and M. Haller, "Intravenous therapy: a review of complications and economic considerations of peripheral access," *Journal of Infusion Nursing: The Official Publication of the Infusion Nurses Society*, vol. 35, no. 2, pp. 84–91, 2012.

[362] L. Fang, S.-H. Fang, Y.-H. Chung, and S.-T. Chien, "Collecting factors related to the haemolysis of blood specimens," *Journal of Clinical Nursing*, vol. 17, no. 17, pp. 2343–2351, 2008.

[363] G. Lippi, G. L. Salvagno, M. Montagnana, M. Franchini, and G. C. Guidi, "Phlebotomy issues and quality improvement in results of laboratory testing," *Clinical Laboratory*, vol. 52, no. 5-6, pp. 217–230, 2006.

[364] G. Lippi, C. Mattiuzzi, and G. C. Guidi, "Laboratory quality improvement by implementation of phlebotomy guidelines," *MLO: Medical Laboratory Observer*, vol. 38, no. 1, pp. 6–7; author reply 7, 2006.

[365] M. Collins, S. Phillips, L. Dougherty, A. de Verteuil, and W. Morris, "A structured learning programme for venepuncture and cannulation," *Nursing Standard (Royal College of Nursing (Great Britain): 1987)*, vol. 20, no. 26, pp. 34–40, 2006.

[366] WHO, *WHO Guidelines on Drawing Blood: Best Practices in Phlebotomy*. WHO Guidelines Approved by the Guidelines Review Committee, Geneva: World Health Organization, 2010.

[367] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 29:1–29:13, 2013.

[368] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 780–787, 2014.

[369] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. B. Haro, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager, "A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2025–2041, 2017.

[370] H. Law, K. Ghani, and J. Deng, "Surgeon technical skill assessment using computer vision based analysis," in *Machine Learning for Healthcare Conference*, vol. 68 of *Proceedings of Machine Learning Research*, pp. 88–99, 2017.

[371] J. A. Martin, G. Regehr, R. Reznick, H. MacRae, J. Murnaghan, C. Hutchison, and M. Brown, "Objective structured assessment of technical skill (OSATS) for surgical residents," *The British Journal of Surgery*, vol. 84, no. 2, pp. 273–278, 1997.