Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Dissertation

Presented to the University of Dublin, Trinity College

in fulfilment of the requirements for the Degree of

Doctor of Philosophy

April 2021

Exploration of Deep Learning Techniques for Natural Image Matting

Sebastian Lutz, M.Sc.

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Sebastian Lutz

May 6, 2021

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Sebastian Lutz

May 6, 2021

Acknowledgments

There is a long list of people that I should probably thank. First my family of course, most importantly my mother who always encouraged me. Then, the lads residing at the *102*, former and current, who were always good for a laugh. I should probably thank my supervisor Aljosa, who hired me in the first place. But let's be honest here, there is only one person without whom nobody at *V-SENSE* would get anything done. So thank you Gail, you are a gem.

Sebastian Lutz

University of Dublin, Trinity College

April 2021

Exploration of Deep Learning Techniques for Natural Image Matting

Publication No. _____

Sebastian Lutz, PhD
University of Dublin, Trinity College, 2021

Supervisor: Aljosa Smolic

Natural image matting is the process of estimating the opacity mask between the foreground object and the background in any type of image. This technique has manifold applications in image and video processing and editing, as well as compositing, and has been an active research topic for many years. Due to the ill-posed nature of the problem, it is difficult to solve and even current state-of-the-art methods have not yet reached a level of performance that satisfies professional production. Therefore, in this thesis we are aiming to advance the performance of natural image matting methods and enhance their usability for professional and casual artists.

First, we introduce the first generative adversarial network for natural image matting. Our novel generator network is trained to predict visually appealing alphas with the addition of the adversarial loss from the discriminator that is trained to classify well-composited images. Further, we improve existing encoder-decoder architectures to better deal with the spatial localization issues inherited in convolutional neural networks by using dilated convolutions to capture global context information without downscaling feature maps and losing spatial information. We present state-of-the-art results on the alphamatting.com online benchmark for the gradient error and give comparable results in others. Our method is particularly well suited for fine structures like hair, which is of great importance in practical matting applications, e.g. in film/TV production.

Second, we investigate the specific problem of extracting the foreground object from an image using the predicted alpha and enhance the usability of our method. Most natural image matting algorithms only predict the alpha matte from the image, which is not sufficient to create high-quality compositions. Further, it is not possible to manually interact with these algorithms in any way except by directly changing their input or output. We propose a novel recurrent neural network that can be used as a post-processing method to recover the foreground and background colors of an image, given an initial alpha estimation. Our method outperforms the state-of-the-art in color estimation for natural image matting and shows that the recurrent nature of our method allows users to easily change candidate solutions that lead to superior color estimations.

Finally, we evaluate video matting methods and propose a neural network for the video matting task. Modern natural image matting algorithms currently outperform classical video matting algo-

rithms due to their high fidelity in predicted alphas in the individual frames of the video. However, these methods do not consider temporal consistency and therefore often introduce temporal artifacts such as flickering. We evaluate different approaches to introduce temporal consistency to these methods to make them suitable for the video matting task and propose a neural network for the video matting task and train it in a way that leverages the single image matting performance of modern algorithms while also introducing temporal consistency to reduce flickering.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter serves as introduction to this thesis. First, the motivation of researching natural image matting is covered. This is followed by a problem description, to familiarize the readers with the issues in solving the problem. Afterwards, the research question is explicitly stated. Following up, the structure of this thesis is presented and a brief overview of the contents of each chapter is given. Finally, a list of publications and contributions is given.

1.1 Motivation

In natural image matting the goal is to predict the opacity mask of an object, typically the foreground object, from an image. *Natural images* in this case refer to any type of image taken in the wild and without any constraints. Constraints to make this task easier could be to assume homogeneous backgrounds, fully centered foreground objects, foreground objects of only a certain type (e.g. portraits) and many more. The opacity mask is usually referred to as the *alpha matte* and contains values from 0 to 1 for every pixel of the image, indicating the range from fully transparent or background pixels to fully opaque ones. This task is related to *chroma keying*, where the goal is the same, but the input is constrained. In chroma keying, the inputs are typically shots from *green screen* scenes where the background contains only a mostly homogeneous single color background. This makes the task much easier, since the pure background color is known for every pixel in the image, which decreases the number of unknowns in the matting equation (See Equation 1.1). Due to the manifold types of backgrounds in natural images, however, this technique can not be used in natural image matting. The trade-off here is obvious: with chroma keying it is relatively easy to extract very high-quality results, but it only works in certain constrained environments. Natural image matting does not have these constraints, but also does not produce the same level of quality than chroma keying at this point in time. Ultimately, the objective in researching natural image matting is to produce results that are good enough for use in the movie industry or by other professional artists using Photoshop and similar products. Many movie and television productions nowadays rely heavily on *visual effects* (VFX) and *computer generated imagery* (CGI). This

has led to working environment that often consist almost entirely of green screen backgrounds. This is necessary for chroma keying to work, but can also heavily impair actors and directors since the physical scene looks nothing like the finished product as can be seen in Figure 1.1.

Other applications of natural image matting are geared more towards the casual creatives. For example, it can be used to easily extract portrait images [1, 2, 3] and create new images with a different background. It can also be used in other image processing tasks to apply various filters and effects only on the foreground or background, such as blurring the background in video calls.

1.2 Problem description

In natural image matting, the input is an image that is expected to contain a foreground object and the image background. Mathematically, every pixel i in the image is assumed to be a linear combination of the foreground and background colors, expressed as:

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \qquad \alpha_i \in [0, 1] \tag{1.1}$$

where α_i is a scalar value that defines the foreground opacity at pixel i and is referred to as the alpha value. Since neither the foreground, nor the background RGB values are known, this is a severely ill-posed problem, consisting of 7 unknown and only 3 known equations. Typically, some additional information in the form of scribbles [5] or a trimap [6] is given as additional information to decrease the difficulty of the problem. Both additional input methods already roughly segment the image in foreground, background and regions with unknown opacity. Generally they serve as initialization

Figure 1.1: Example image for a green screen working environment from the set *The Hobbit: An Unexpected Journey*. Top the finished scene, bottom the physical scene. Image taken from [4].

information and many methods propagate the alpha values from known image regions to the unknown region. As can be seen in Figure 1.2, both trimaps and scribbles can be easily created manually and can massively improve the matting results.

However, even perfect alpha matte predictions are insufficient when the aim is to create new compositions from the foreground object. Since the RGB values seen in the input image are always a

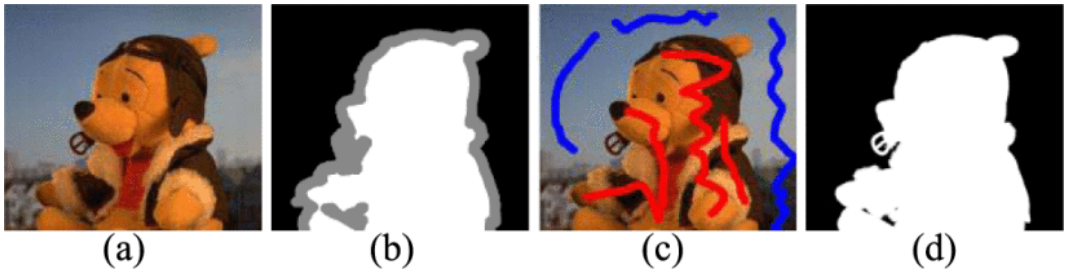

Figure 1.2: Example for trimap and scribbles. (a) Input image, (b) Trimap, (c) Scribbles, (d) Alpha matte. Image taken from [7].

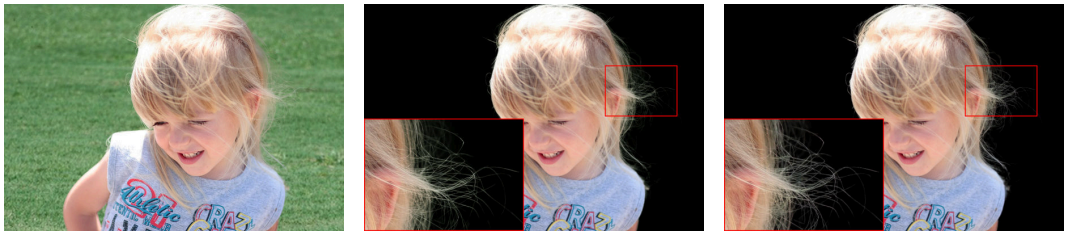

Figure 1.3: Example of background colors bleeding through in composition. From left to right: Input image, Composition using the ground-truth alpha with colors taken from the input image, Composition using the ground-truth alpha with colors taken from the method described in chapter 4. As can be seen, if the colors from the input image are taken for the composition, the green background from the grass is bleeding through into the new composition.

blend of the true foreground colors and the background, it is necessary to additionally predict the foreground colors to create faithful compositions, as can be seen in Figure 1.3. This is a problem that is often ignored by matting algorithms.

1.3 Research objectives

As has been stated previously, Natural Image Matting is a complex problem with plentiful applications in the industry, as well as one of the classical computer vision problems. In the last few years, many computer vision problems have seen great strides in employing deep convolutional neural networks (CNNs). As will be detailed in Section 2.1, this is also the case for natural image matting. How-

ever, CNNs need a large amount of training data to generalize well, which poses a problem especially for natural image matting, since collecting the ground-truth alpha for a large amount of images is very difficult. The process of matting an image by hand can be quite long and even if the alpha is created by an experienced artist, it is never going to be the true ground-truth, only a (possibly very good) estimation. Therefore this research thesis aims to find good neural network architectures to deal with this problem. Furthermore, the goal in many matting applications is to extract the foreground object and composite new images. Naively using the alpha matte to extract the foreground image for composition will not always lead to faithful new compositions, due to the color disparity in the foreground object and the respective backgrounds of the old and the new image. Further different image characteristics such as lighting conditions, noise, etc. in the foreground object and the new background can also lead to compositions that do not look real. The second aim of this thesis is therefore to investigate ways to create high-quality compositions. Finally, processing images independently in video sequences will often lead to temporal inconsistencies such as flickering. In this thesis we aim to investigate ways to alleviate this problem for natural image matting methods. The research objectives in this thesis are therefore:

- Can new deep learning architectures be used to train convolutional neural networks for the problem of natural image matting despite being trained on only a small set of training images.
- Can the foreground object in natural images be extracted to achieve realistic and high-quality compositions with a new background.

- Can temporal consistency be guaranteed when using natural image matting methods in video sequences, especially considering the sparsity of video matting data.

These research objectives are covered in chapter 3, 4 and 5 respectively.

1.4 This Dissertation

This dissertation is split into 6 chapters and an appendix. Following this introduction chapter, the background and related works relevant to this work are discussed in chapter 2. This will include a general introduction into the deep learning methods that are used in this and related works, a general overview into image-to-image translation networks, a historical overview of classical approaches to solve natural image matting, a more in-depth description of deep learning methods for natural image matting, a report on video matting methods and finally a thorough description on the available image and video matting data.

In chapter 3 a generative adversarial network for natural image matting is presented. This method was designed to predict alpha mattes that can be used to generate visually compelling compositions. By leveraging the discriminator and a modified cycle consistency during the training process, this method outperforms the previous state-of-the-art in predicting fine structures such as hair in the input image.

As mentioned earlier, a good alpha matte is necessary but insufficient to extract the foreground object in a way that leads to high-quality compositions. Also required are the unblended foreground colors of the object. A post-processing method that can be used

in addition to any other matting algorithm is presented in chapter 4. This method sits on top of any matting algorithm and can be used to faithfully predict the foreground and background colors of the image. It is designed as a recurrent inference machine and essentially solves the inverse compositing problem by separating the foreground and background given the alpha. It also offers additional user interaction that can be used to further enhance the quality of the results.
In chapter 5, we do an evaluation on video matting. We investigate ways to enforce temporal consistency in video matting sequences and propose a method trained on video matting data directly. This is the last method detailed in this dissertation and a conclusion of the whole thesis can be found in chapter 6. A summary of the methods presented is given and potential future work is presented.
Finally, a list of abbreviations can be found in appendix A, followed by additional results in appendix B.

1.5 Publications/Contributions

Several of the methods presented in this dissertation have been published as stand-alone research papers. Below, a list of publications is given:

- **Sebastian Lutz** and Konstantinos Amplianitis and Aljosa Smolic:
  AlphaGAN: Generative adversarial networks for natural image matting,
  British Machine Vision Conference, 2018 [8].
  This BMVC paper corresponds to the method presented in chapter 3.

- **Sebastian Lutz** and Aljosa Smolic:

Foreground color prediction through inverse compositing, Winter Conference on Applications of Computer Vision, 2021 [9].
This WACV paper corresponds to the method presented in chapter 4.

Publicly available code for the above contributions can be found on github: https://github.com/seblutz. All methods in this thesis were implemented in Python using the Pytorch [10] deep learning framework.
The author has also worked on several publications that are not part of this dissertation:

- Rafael Monroy and **Sebastian Lutz** and Tejo Chalasani: SalNet360: Saliency Maps for omni-directional images with CNN, Signal Processing: Image Communication, 2018, ISSN: 0923-5965 [11].

- **Sebastian Lutz** and Mark Davey and Aljosa Smolic: Deep Convolutional Neural Networks for estimating lens distortion parameters, Irish Machine Vision and Image Processing Conference, 2019 [12].

- Xu Zheng and Tejo Chalasani and Koustav Ghosal and **Sebastian Lutz** and Aljosa Smolic: STaDA: Style Transfer as Data Augmentation, 14th International Conference on Computer Vision Theory and Applications, 2019 [13].

- Matis Hudon and **Sebastian Lutz** and Rafael Pagés and Aljosa Smolic: Augmenting Hand-Drawn Art with Global Illumination Effects

through Surface Inflation,
The 16th ACM SIGGRAPH European Conference on Visual Media Production, 2019 [14].

Chapter 2

Background

In this chapter, the background works that serve as basis for the contributions of this thesis are presented. As with many computer vision tasks, neural networks have outperformed classical natural image matting algorithms in both speed and accuracy and now serve as the state-of-the-art for this problem. A complete review of deep learning would be outside the scope of this work, but some of the seminal deep learning papers that serve as basis and inspiration of the methods presented later in this thesis and the current state-of-the-art in natural image matting are presented in Section 2.1. This section is followed by a brief overview of the classical methods of natural image matting in Section 2.2. A more in-depth review of the current state-of-the-art will be given in Section 2.3, with a focus on the design choices of the respective networks. Afterwards, video methods will be discussed in Section 2.4. One of the most important aspects of any deep learning algorithm is the dataset the network is trained on. As such, the various existing datasets for natural image matting are presented in Section 2.5. Finally, some of the various methods indirectly related to the contributions in this thesis, but which are not fit to serve as comparisons are mentioned in Section 2.6 to give a more overarching overview of the field.

2.1 Deep learning in computer vision

In recent years, deep convolutional neural networks have become the state-of-the-art in many computer vision problems. The core concepts of artificial neural networks, what is now known colloquially as *deep learning*, are not new. However, only recently has it been possible to efficiently train networks of sufficient capacity due to the advances in the hardware used for training and the possibility to gather large datasets. In general, neural networks are a collection of *layers* that are stacked on top of each other. The input of the network is propagated through each of the layers until the final layer outputs the result. During training, the predicted result is used to calculate a *loss*, describing how far from the ground-truth the prediction is. This loss is then used to update the parameters of the individual neurons that compose the layers of the network. This is efficiently done through *back-propagation*: Using the *chain-rule*, the gradient of the loss in respect to the parameters of one layer can be calculated and propagated from the last layer to the front, hence the name. Since this can be done almost entirely through matrix multiplications, it can be very efficiently calculated on modern GPUs. A full explanation of deep learning would be beyond the scope of this work, however, a good foundation of all the concepts necessary can be found in [15].

Very Deep Convolutional Networks for Large-Scale Image Recognition

One of the seminal architectures in computer vision is the VGG [17] network. VGG was designed as a classification network and many of its design choices carry through to modern architectures. In their paper, Simonyan et al. showed that the depth of a network is a very

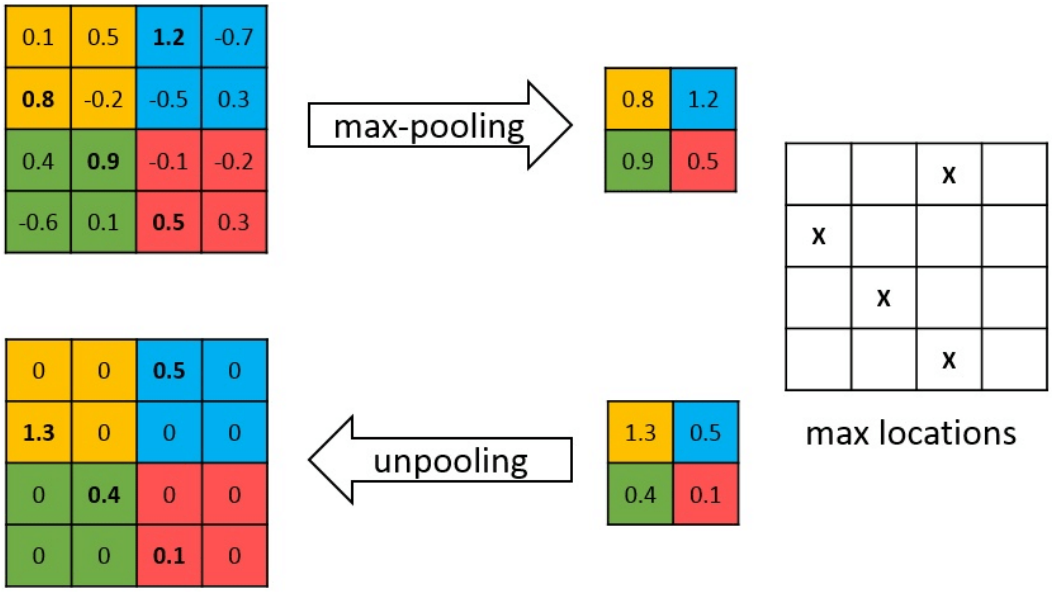

Figure 2.1: Example of max-pooling and unpooling layers with a kernel size and stride of 2. During max-pooling, the maximum value within the kernel window is taken and the location stored. During unpooling, the locations can be used to do a lossy reconstruction of the original features. Image taken from [16].

important factor for the network to learn good image representations. They demonstrated that multiple stacks of convolution layers with a small 3×3 filter size outperforms more shallow networks, with less layers but larger convolution filters without increasing the number of total parameters in the network. The network architecture is quite simple and consists of 5 blocks of stacked convolutions, each followed by a 2×2 max-pooling layer with stride 2 (see Figure 2.1) that is used to downsample the intermediate feature maps. These 5 blocks are followed by a set of 3 fully-connected layers that are used to classify the image. Each of the blocks of stacked convolutions consists of 3×3 convolution layers with stride and padding of 1, which keeps the spatial size of the output feature maps the same as the input. All convolution and fully-connected layers are followed by ReLU [18] non-linear activation layers. A visualization of the network architecture can be seen in Figure 2.2.

The design choice of having 5 blocks of convolution layers, each fol-

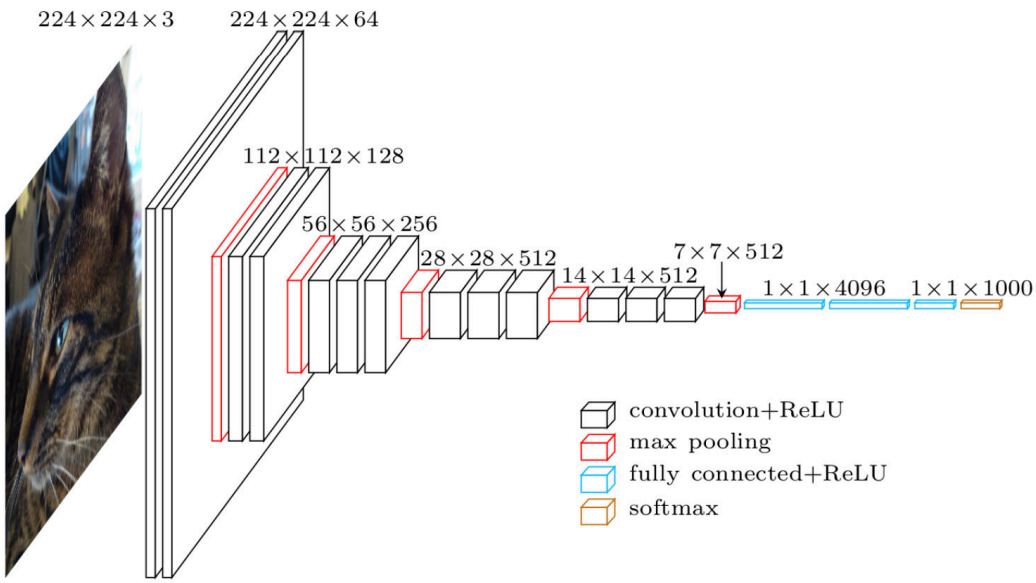

Figure 2.2: Network architecture for VGG-16. Image taken from [19].

lowed by some kind of downsampling has proven extremely popular and can be found even in more modern networks such as Resnet [20] and Densenet [21], as has the choice of using only 3×3 convolutions. It has been shown that as the network becomes more deep, the features learnt in each convolution layer become more abstract. The first layer learns local features such as edges, while the later blocks learn features such as textures and finally high-level features such as eyes and mouths [22]. The output of the last convolution layer in these kinds of classification networks is therefore a highly downsampled feature map consisting of highly discriminative features, which is why these classification networks are often used as an encoder for other computer vision tasks.

Fully convolutional networks for semantic segmentation

Such is also the case in the work of Long et al. [23]. Prior to their publication, patch classification [24] was a popular way to use CNNs for segmentation. In these methods, every pixel was independently classified into one of a set of semantic classes, based on a patch

around that pixel as input. This was mainly due to the fully-connected layers at the end of classification networks that need a fixed-sized input to the network. This changed when Long et al. proposed to concept of *fully convolutional networks* (FCN) in their paper. They took some of the well-performing classification networks at the time, AlexNet [18], GoogLeNet [25] and notably VGG [17], and transformed the fully connected layers of these networks into convolution layers to output spatial feature maps instead of classification scores (See Figure 2.3). These feature maps are then upsampled through transposed convolutions to produce per-pixel labeled outputs of the same size as the input. This allows the network to be trained end-to-end with training inputs of arbitrary sizes. They also managed to refine their architecture by introducing some skip-connections from the 3rd and 4th pooling layer to be able to predict finer details while retaining high-level semantic information. All these contributions make this FCN architecture an important forerunner for deep learning applied to pixel-wise classification or regression tasks.

The weakness of this architecture, however, is its inherent spatial invariance because of the max-pooling layers of the network that discard potentially useful global context information. It is also not instance-aware i.e. it does not differentiate between two or more unique objects of the same class. To overcome the first problem, two different approaches have been proposed. The first approach uses an encoder-decoder architecture, where the encoder consists of a classification network like VGG with the fully-connected layers removed. Additionally, all max-pooling layers retain the indices of the maximum element that they used to pool the feature maps (See Figure 2.1). Following the encoder, a decoder is used to upsample these low-resolution feature maps back to the original image

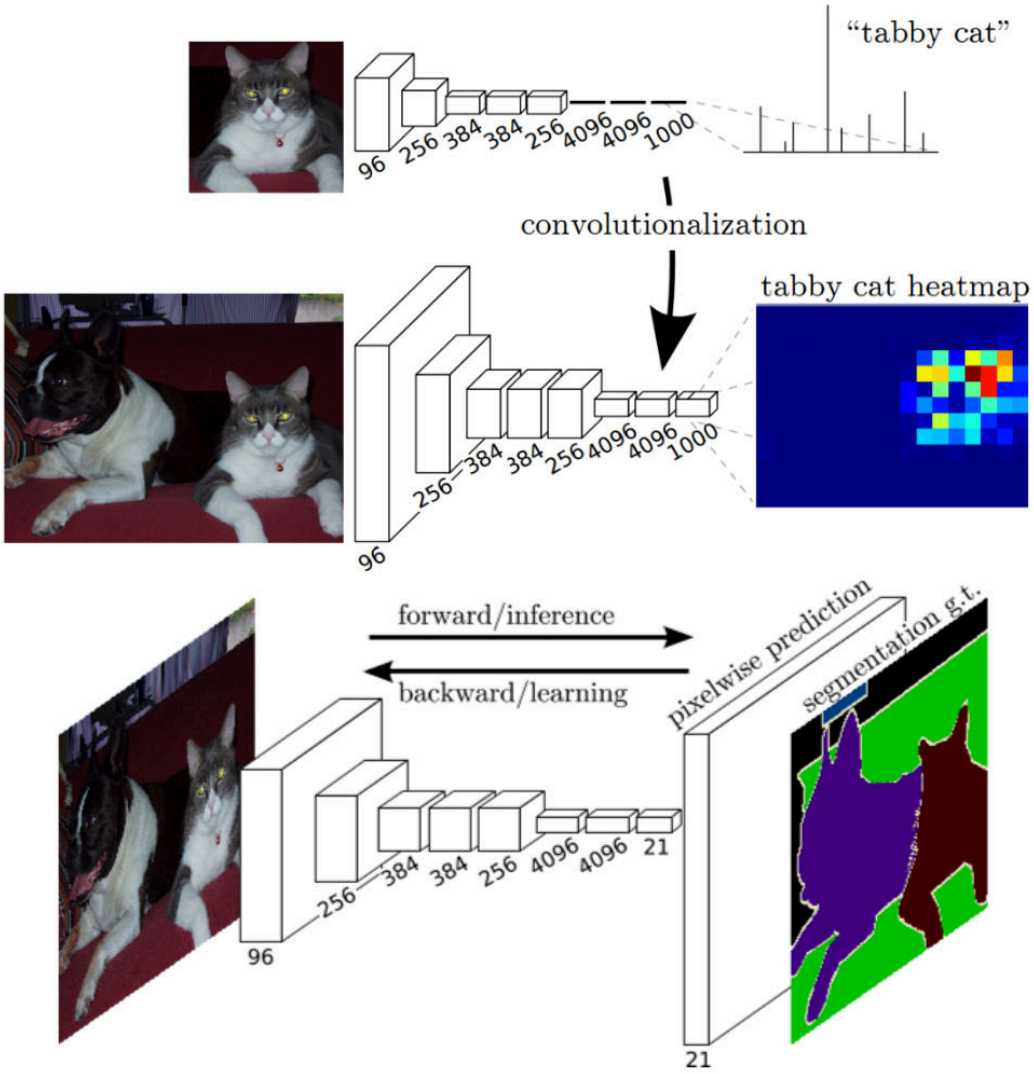

Figure 2.3: Starting from a pretrained image classification network, the fully-connected layers are transformed into convolution layers. The output feature maps of the last layer are adapted to the number of semantic classes and an upsampling layer is added. The network can now be trained fully end-to-end for semantic segmentation. Image taken from [19].

size by using the stored max-pooling indices to correctly upsample the feature maps in each stage, followed by some convolution layers. SegNet [26] is one example of this approach. Often, these architectures also include skip-connection between feature maps of the same size such as in the popular U-Net [27] architecture.

The second approach tries to solve the problem of missing global context with the use of *dilated* (or *atrous*) convolutions. Dilated

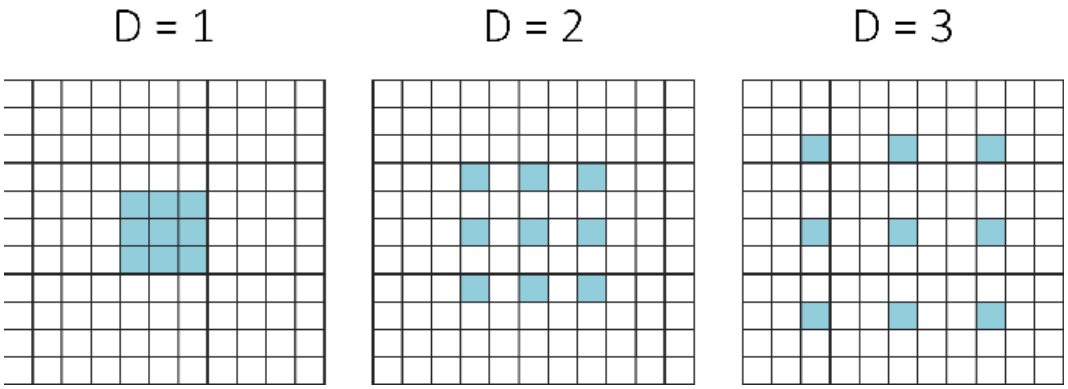

Figure 2.4: Normal convolutions are dilated convolutions with a dilation rate of 1. With higher dilation rates, the convolution filters are padded with zeros. Even though the number of weights stay the same, the receptive field is increased. Image taken from [19].

convolutions are convolution filters that support exponentially expanding receptive fields without losing resolution [28]. They are upsampled filters by a factor of r with zeros in between filter weights. An example of this can be seen in Figure 2.4. Even though the effective filter size increases, only the non-zero filter values need to be taken into account, so the number of parameters and the number of operations per position stay constant. This allows the explicit control of the spatial resolution of network feature responses. These dilated convolutions have been successfully used in several architectures for semantic segmentation [29, 29, 30, 31] and other image-to-image translation tasks.

Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation

The latest Deeplabv3+ [31] architecture combines both approaches. As their encoder, they use a modified aligned Xception [33] network, which can be seen in Figure 2.7. Xception uses depthwise separable convolutions (See Figure 2.5), which are normal convolutions factorized into depthwise convolutions, followed by a pointwise 1×1 convolution. The depthwise convolutions are independent

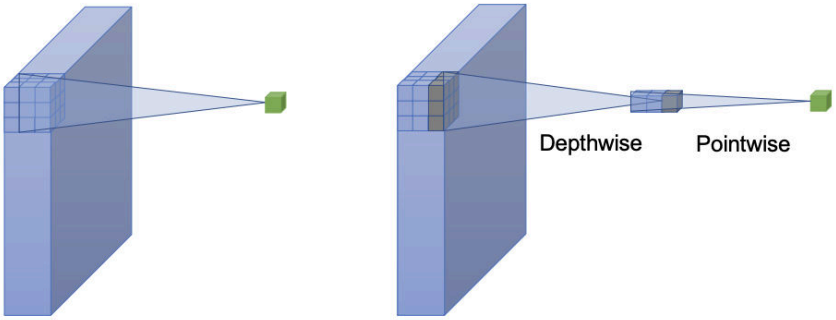

Figure 2.5: Depthwise separable convolutions are depthwise convolutions followed by a pointwise 1×1 convolution. Image taken from [32].

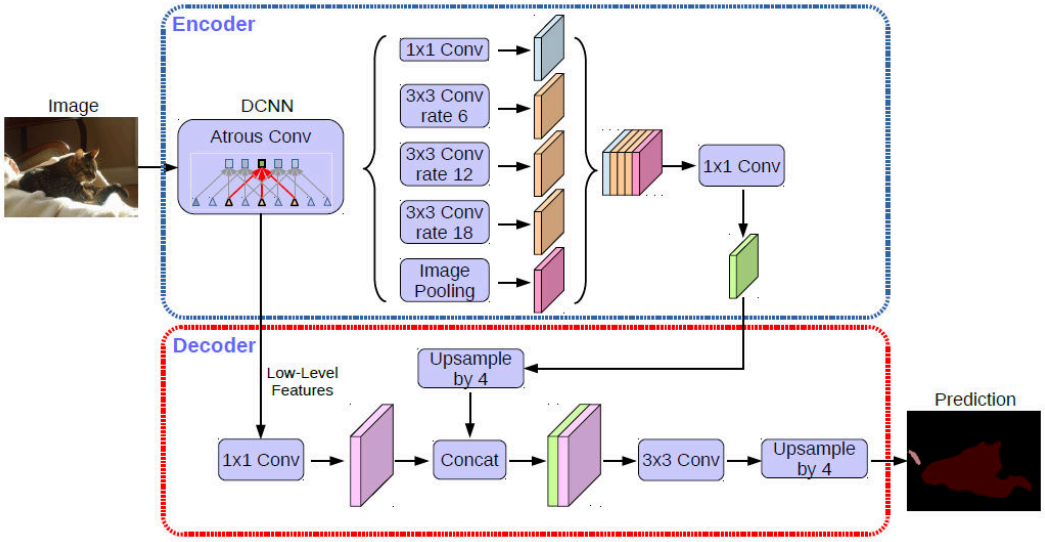

Figure 2.6: Network architecture for Deeplabv3+. Image taken from [31].

on each input channel and the pointwise convolution is used to combine these independent outputs. This drastically reduces the computation complexity of the layer over normal convolutions. For Deeplabv3+, the network is modified to be deeper in the middle-flow of the network, all max-pooling operations are replaced by strided depthwise separable convolutions and additional batch normalization [34] and ReLU activations are added after every 3×3 depthwise convolutions. These modifications allow the network to output feature maps of arbitrary resolutions by adjusting the strides of

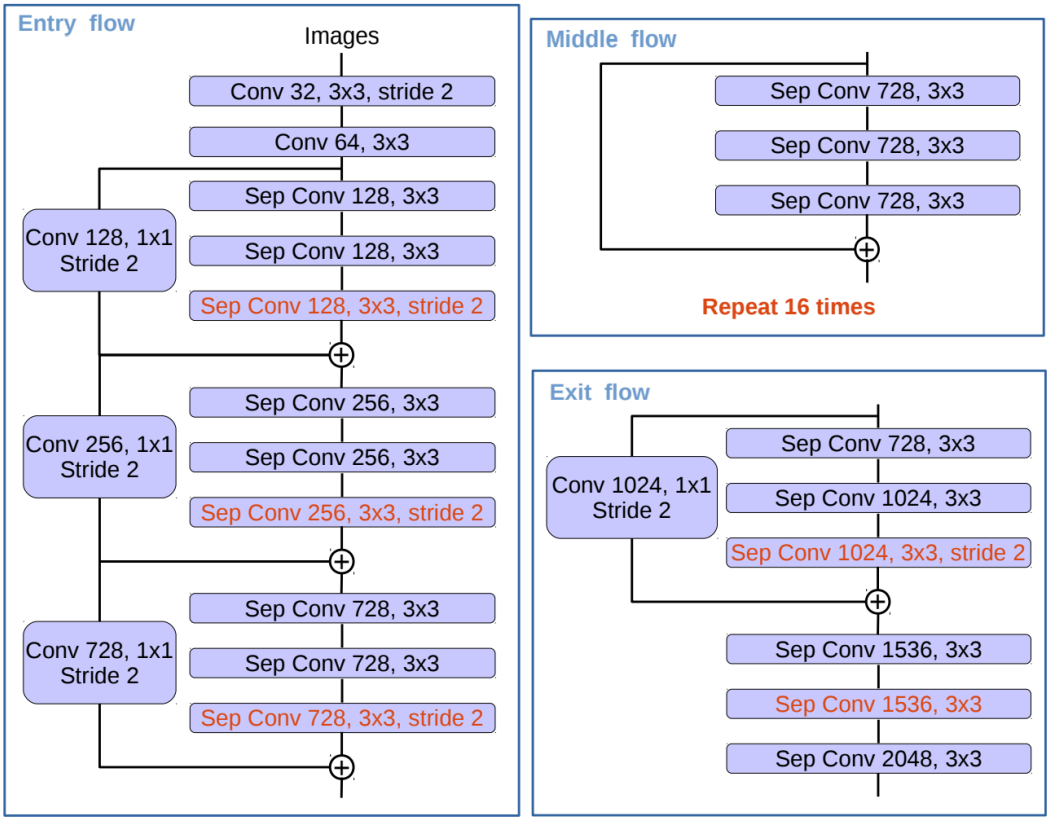

Figure 2.7: The modifications done by Deeplabv3+ on Xception: (1) More layers, (2) All max-pooling layers replaced by strided depthwise separable convolutions, (3) Extra batch normalization and ReLU layers added after every 3×3 depthwise convolutions. Image taken from [31].

the convolutions. In their best model, they train the network with output stride = 16 and use output stride = 8 during evaluation by reducing the stride and doubling the dilation rate in the later convolutions.

Following the encoder of the network, the deeplabv3+ architecture adds an *atrous spatial pyramid pooling* (ASPP) module. This module works by using multiple parallel dilated depthwise separable convolution layers to extract features at different sampling rates. These parallel convolutions are then fused to generate a final result (See Figure 2.8), which encodes multi-scale contextual information. Finally, Deeplabv3+ adds a decoder for pixel-wise classification at full resolution. The decoder first bilinearly upsamples the output of

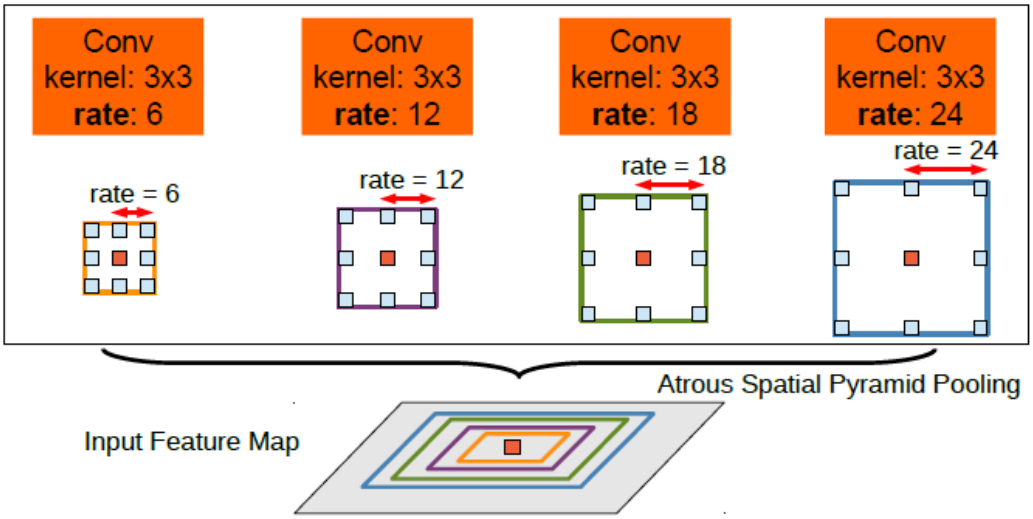

Figure 2.8: Atrous spatial pyramid pooling module. In this module, parallel dilated convolutions with different dilation rated are applied to the input feature map and concatenated together. Image taken from [29].

the ASPP to output stride = 4 and then concatenates the result with the low-level features of the encoder at the corresponding spatial resolution. However, before concatenation, the number of channels of the low-level features are reduced through a 1×1 convolution. After concatenation, 3×3 convolutions are used to refine the features, followed by another bilinear upsampling to the final output resolution.

Image-to-Image Translation with Conditional Adversarial Networks

In semantic segmentation, networks are trained with the cross-entropy loss function. Using only this single loss has been shown to work very well for this task, but this is not the case for all computer vision problems. In regression tasks especially, a single loss function or even a combination of loss functions often lead to networks that do not converge optimally. Even though the training is entirely data driven and the network learns features based on the training data,

the loss functions used to train are designed by hand and are therefore susceptible to certain preconception about the data and task that the person designing the training pipeline may have. In such cases, generative adversarial networks (GANs) have been shown to achieve superior results.

In the Pix2Pix architecture by Isola et al. [35], conditional generative adversarial networks are used as a general purpose solution to image translation problems. The trained networks not only learn the mappings from one image domain to another, but also (through the discriminator), the loss function required to train this mapping. This results in a very robust architecture that can be used in a wide variety of image-translation problems without hand-engineered loss functions, as long as a dataset of paired images is available.

GANs learn a mapping from a random noise vector $\mathbf{z}$ to an output image $\mathbf{y}$. Conditional GANs, however, learn a mapping from an observed input image $\mathbf{z}$ and a random noise vector $\mathbf{z}$ to an output image $\mathbf{y}$. This results in the following objective function:

$$\mathcal{L}_{cGAN}(G, D) = \log\, D(\mathbf{x}, \mathbf{y}) + \log\, (1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z})) \tag{2.1}$$

where G tries to minimize this objective against D, which tries to maximize it. Without the noise vector $\mathbf{z}$, the generator can still learn mappings from $\mathbf{x}$ to $\mathbf{y}$, but could only produce deterministic outputs. Other conditional GANs have therefore introduced $\mathbf{z}$ as an additional input to G (e.g. [36]). In their Pix2Pix approach, however, Isola et al. introduce noise only through dropout for several layers in the generator at training and test time. In addition to the conditional loss in Equation 2.1, they also add *L1* loss to the generator so that G is not only tasked to fool the discriminator D, but also to get close to the ground-truth.

Their network architectures are adapted from those of *DCGAN* [37]. Because image-to-image translation problems map from high-resolution images to high-resolution images, they use a U-Net [27] encoder-decoder structure with skip connections from the encoder to the decoder to share low-level information (e.g. edges) that can be useful to the generated mapping.

For their discriminator structure, they design a network that they call PatchGAN, which tries to classify each $N \times N$ patch in an image as real or fake. This discriminator is run convolutionally across the image and all responses are averaged to get a final output of D.

It is of note, that, because of the conditional GAN architecture, the discriminator needs to both have $\mathbf{y}$ (Or $G(\mathbf{x})$) and $\mathbf{x}$ as input. Therefore the input image and the mapping are appended into a multi-channel image before serving as an input for D.

Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks

In their paper, Isola et al. [35] show remarkable results in image-to-image translation, but they retain the problem of data. For their approach to work, it needs to be trained on a large enough, paired-dataset. In a follow-up paper, Zhu et al. from the same research group introduce their CycleGAN [38] approach. This novel approach can be used for data where there is no paired training data available.

Because finding a mapping $G : \mathbf{x} \rightarrow \mathbf{y}$ using only the adversarial loss is severely under-constrained, Zhu et al. add the reverse mapping $F : \mathbf{y} \rightarrow \mathbf{x}$ and introduce a *cycle consistency* loss to enforce $F(G(\mathbf{x})) \approx \mathbf{x}$ (See Figure 2.9). The adversarial losses alone can not guarantee that the learned function can map an input $\mathbf{x}$ to a correct output $\mathbf{y}$,

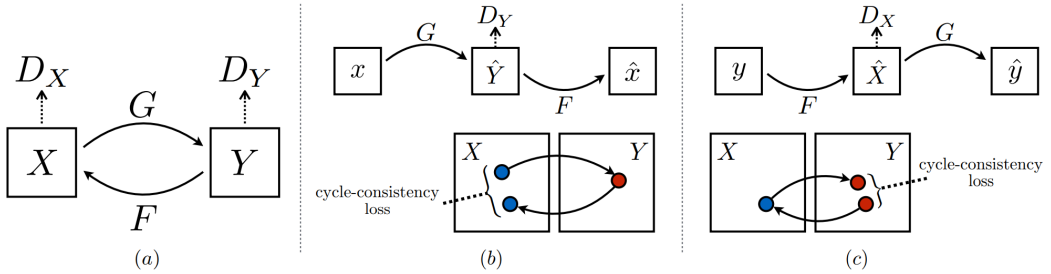

Figure 2.9: a) The *CycleGAN* model consists of two mapping functions G and F and their associated discriminators D_X and D_Y. b) The forward *cycle-consistency* loss. c) The backward *cycle-consistency* loss. Image taken from [38].

since a network with large enough capacity can map the same set of input images to any random permutation of images in the target domain. To reduce the space of possible mapping functions, Zhu et al. argue that the learned functions should be *cycle-consistent*: For every image $\mathbf{x}$, the image translation cycle should bring it back to the original image $\mathbf{x} \rightarrow G(\mathbf{x}) \rightarrow F(G(\mathbf{x})) \approx \mathbf{x}$ and similarly for $\mathbf{y}$. This defines the *cycle-consistency* loss:

$$\begin{aligned} \mathcal{L}_{cyc}(G, F) = &\|F(G(\mathbf{x})) - \mathbf{x}\|_1 \\ &+ \|G(F(\mathbf{y})) - \mathbf{y}\|_1 \end{aligned} \tag{2.2}$$

The full objective is then the two adversarial losses for G and F as in Equation 2.1, added with the *cycle-consistency* loss in Equation 2.2. For their discriminator structure, they use the *PatchGAN* architecture that was introduced in [35], for their generators, they adapt the architecture from [39], which was designed for style transfer and super-resolution. The network consists of two strided convolutions to downscale the image, several residual blocks [20] and two transposed convolutions to upscale the image back to the original resolution. They also use instance-normalization [40], similar to the Johnson et al. [39].

2.2 Classical natural image matting

Natural image matting is a classical computer vision problem and has been studied for many years. Generally, one can classify any matting algorithm into 3 categories: sampling-based, affinity-based and learning-based. Methods of the first two categories will be briefly described in this section, while a more in-depth description of the learning-based methods will be given in Section 2.3. A more comprehensive overview on classical natural image matting can be found in the work by Wang et al. [7].

Sampling-based

Sampling-based matting algorithms consider foreground and background samples in a local region around the unknown pixel and seek to estimate the alpha based on these samples and the color distribution of the pixels in the local neighbourhood. Bayesian matting [6] is one such approach that uses a continuously sliding window that collects samples that were previously estimated or in the known regions of trimap. The matting problem is defined in a Bayesian framework and Chuang et al. use a maximum a posteriori technique to solve it. This approach has been improved in [41] by representing the color distributions with global statistics instead of local means, which reduces the computational costs while increasing the components of the *Gaussian Mixture Model (GMM)* to explain all the local colors in the image. In [42], Chang et al. do not model the foreground or background colors and instead build the probabilistic terms of the Bayesian framework on weights that depend on spatial distances and color differences. This makes the approach more robust to sampling outliers.

Gastal et al. proposed a shared-sampling method [43] to improve

the sampling efficiency. They show that sharing samples within a small neighborhood can reduce the computational cost with little defect to the matting quality. They use a new objective function to select good samples based on their spatial and photometric and characteristics. Other recent Sampling-based methods include [44], [45] and [46].

Affinity-based

In Affinity-based matting algorithms, the alpha values are propagated from regions of known values based on pixel similarity metrics. A closed-form solution for the matting problem was proposed in [47] with the assumption that the foreground of each pixel can be represented by a linear combination of two constant colors within a local neighbourhood. This local neighbourhood is used to determine the information flow and to propagate alpha values. This local affinity definition is also often used as a post-processing step as proposed by [43]. Non-local affinity definitions such as *KNN-Matting* [48] calculates several neighbourhoods for every unknown pixel and assigns them to have similar alpha values based on their distance to each other in feature-space. Similar to this Chen et al. propose a method [49] that represents each pixel as a linear combination of its neighbors in their feature space. Their affinity-definitions have been improved by [50]. There are also several hybrid approaches such as [51] that use a Sampling-based matting approach as a starting point and refine the results with non-local affinities.

2.3 Deep learning in natural image matting

Deep learning has been used very successfully in many computer vision tasks and has been pushing the performance boundaries of new algorithms to an unprecedented degree. It is therefore no surprise that these algorithms also perform very well in natural image matting.

Natural Image Matting Using Deep Convolutional Neural Networks

The first of these approaches was DCNN matting by Cho et al. [52]. In their method, they use a shallow neural network consisting of 6 convolution layers to predict the alpha. Their input is a 5 channel image: The RGB image (normalized to $[0, 1]$), the predicted alpha from closed-form matting [47] and the predicted alpha from KNN matting [48]. This means the method does not learn to predict the alpha in itself, but rather the non-linear combination of previous matting methods within small local patches. CF and KNN matting are examples of local and non-local affinity-based matting methods and complement each other well. In their paper, Cho et al. show that their network learns to combined the results of both methods in a way to outperform both.

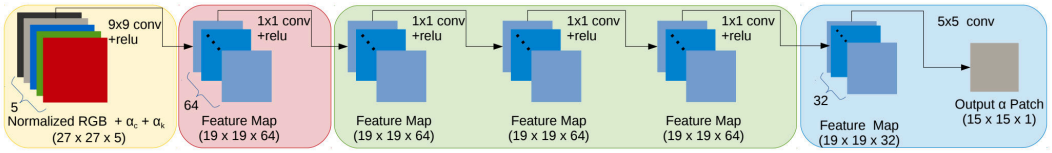

Figure 2.10: Network architecture of DCNN matting. Image taken from [52].

Deep Image Matting

One of the disadvantages that Cho et al. faced when training DCNN

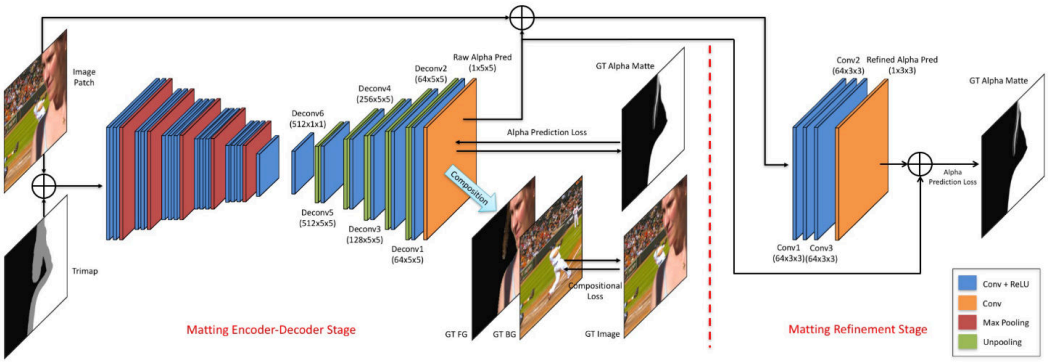

Figure 2.11: Network architecture of Deep Image Matting. Image taken from [54].

matting was the scarcity of training data. They were only able to train their method using the 27 training images available as part of the alphamatting.com online benchmark [53]. When Xu et al. [54] released their Deep Image Matting paper, a big part of their contribution was the matting dataset discussed in detail in Section 2.5, which they used to create a training dataset of $49,300$ images through data augmentation. They were able to use this enhanced dataset to train a deep neural network to predict the alpha from only the RGB image and the trimap as input. Their network is a simple encoder-decoder network that uses VGG16 as encoder and a slightly smaller decoder consisting of only a set of unpooling, convolution and ReLU in each stage. This encoder-decoder directly predicts the full resolution alpha matte and is followed by a matte refinement stage, which concatenates the RGB image with the initial alpha prediction and refines the alpha prediction through an additional set of convolution layers as can be seen in Figure 2.11. To train their network, they used two loss functions: The alpha prediction loss and the composition loss, which are defined as:

$$\begin{aligned} \mathcal{L}_\alpha^i &= \sqrt{(\alpha_p^i - \alpha_g^i)^2}, & \alpha_p^i, \alpha_g^i \in [0,1]. \\ \mathcal{L}_C^i &= \sqrt{(c_p^i - c_g^i)^2}, & c_p^i, c_g^i \in [0,1]. \end{aligned} \tag{2.3}$$

where α_p^i is the output prediction at pixel i, α_g^i the corresponding ground-truth, c_p^i is the RGB value at pixel i of a newly composited image out of the ground-truth foreground, ground-truth background and predicted alpha, and c_g^i the corresponding RGB value composited from the ground-truth alpha. In their paper, Xu et al. show that their network massively outperforms previous state-of-the-art and proves to be far more agnostic to the size of the unknown region in the input trimap than previous methods.

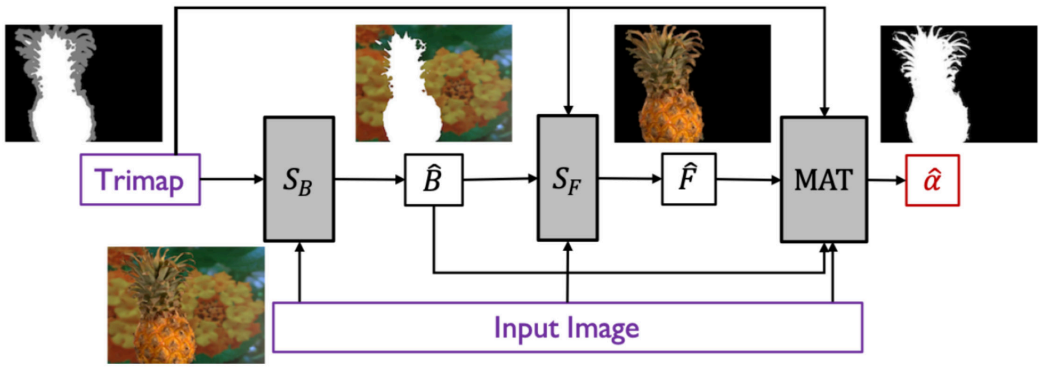

Figure 2.12: Network architecture of Samplenet. Image taken from [55].

Learning-Based Sampling for Natural Image Matting

Following Xu et al. and the proposed method in chapter 3, Tang et al. [55] developed Samplenet. Their core idea involves the intrinsic link between the foreground color, background color and the alpha, and how information of either of these values can help predicting the others. To leverage this, their method has 3 stages: first, they take the input RGB image and the trimap as input and predict the background colors in the unknown trimap region using a sampling network S_B. This prediction is then concatenated to the original input and serves as input to the second stage, where a second sampling network S_F predicts the foreground colors in the unknown

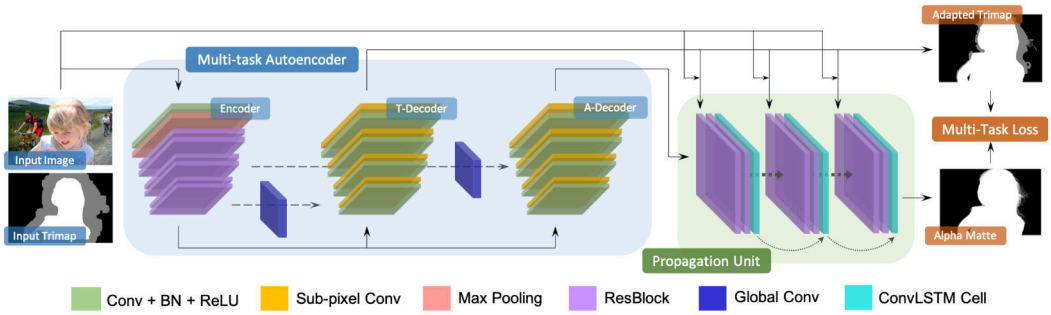

Figure 2.13: Network architecture of AdaMatting. Image taken from [57].

region. In the third and final stage, both predictions are concatenated to the original input and processed by a matting network MAT that outputs the final alpha prediction. For their sampling networks, Tang et al. use the inpainting networks by Yu et al. [56] and for their matting network they use the architecture proposed in Section 3.2.2. However, potentially any inpainting or alpha matting network could be used in their method. This 3-stage approach outperforms the previous state-of-the-art and has the additional benefit of also predicting foreground colors, which can be used to generate much more realistic new compositions as discussed in Section 1.2. However, the downside of their approach is that it requires 3 different networks which are used to subsequently process the input.

Disentangled Image Matting

Most natural image matting methods rely on a trimap as additional input. However, most trimaps are quite coarse and most pixels in the unknown region are either fully opaque or fully transparent. Cai et al. [57] argue that this is a major limitation, since networks have to solve two different problems at the same time: Identifying true blending pixels inside the trimap region, and estimating accurate alpha values for them. To solve this issue, they propose to disentangle the problem into two sub-tasks. In the trimap adap-

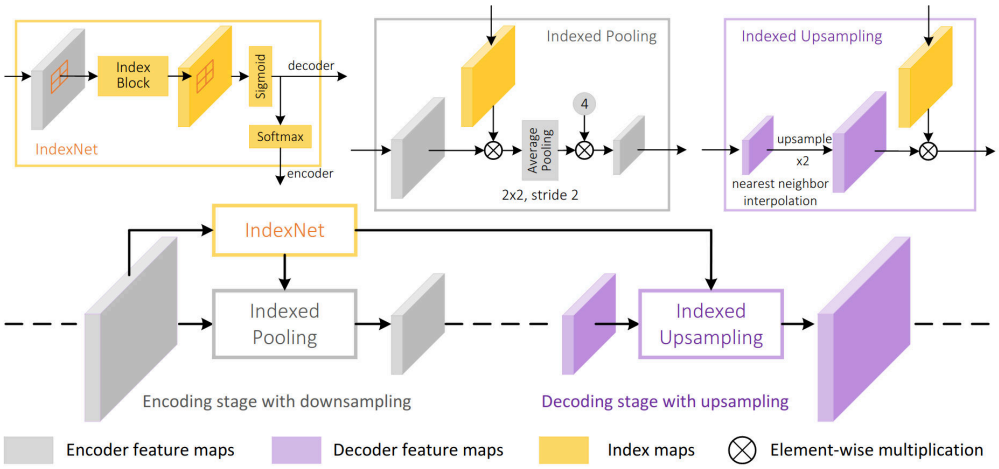

Figure 2.14: IndexNet architecture. Image taken from [59].

tion tasks, they estimate the global structure of the image through the pixel-wise classification into pure foreground, pure background and semi-transparent regions. In the alpha estimation task, they use this information to estimate the true opacity of each pixel in the semi-transparent regions. In their AdaMatting architecture, they achieve both sub-tasks in a single network. Their architecture can be seen in Figure 2.13 and consists of a single encoder to produce shared features and one decoder for each sub-task. Skip connections are used from the encoder to the decoders, with high-level features linked to the trimap decoder and low-level features linked to the alpha decoder. The output of both decoders are concatenated together as input of the propagation unit, which consists of two residual blocks and a convolutional LSTM [58] cell. This recurrent network can refine the alpha prediction over several iterations.

Indices Matter: Learning to Index for Deep Image Matting

Lu et al. [59] propose a different approach to improve the quality of alpha predictions. They observe that using index-guided unpooling layers in the decoder of networks such as with Deep Image Matting [54] leads to better results than using upsampling operators such as

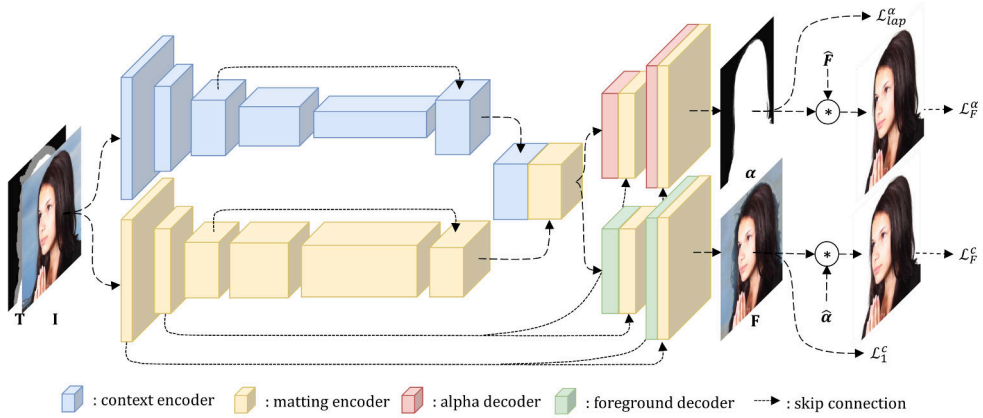

Figure 2.15: Network architecture of Context-Aware Matting. Image taken from [60].

bilinear interpolation or transposed convolutions. They introduce the concept of learning to index, where indices are predicted as a function of feature maps. They present IndexNet, where pooling and unpooling indices are self-learned adaptively from data and which can be plugged into any encoder-decoder network. As seen in Figure 2.14, IndexNet replaces all downsampling and upsampling layers in a network. For each stage in the encoder, IndexNet takes the final feature map as input and predicts a feature map corresponding to indices. In the encoder, this index feature map is activated by a sigmoid and element-wise multiplicated with the initial feature map of the encoder. Afterwards, the 2×2 patches are average pooled to half the spatial size of the feature map and multiplicated by a constant. In the decoder, the feature map is first upsampled using nearest-neighbour interpolation and then element-wise multiplicated using the generated index feature map from IndexNet at the corresponding stage of the encoder.

Context-Aware Image Matting for Simultaneous Foreground and Alpha Estimation

In Context-Aware Matting, Hou et al. [60] develop a two encoder,

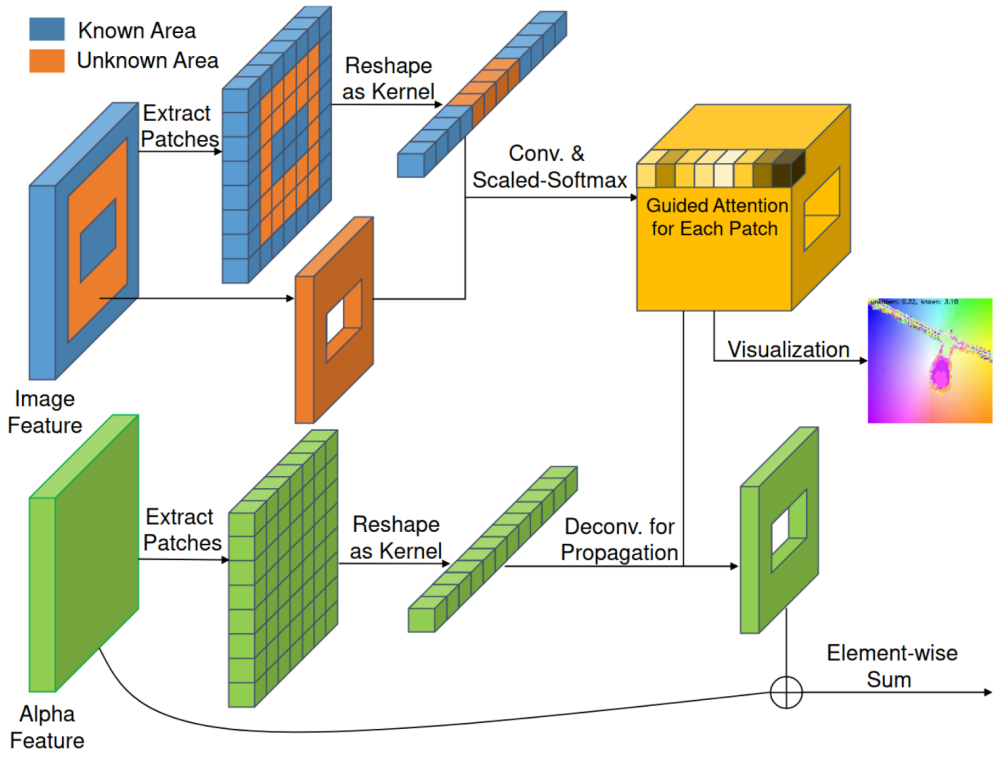

Figure 2.16: Architecture of guided contextual attention module. Image taken from [61].

two-decoder network for the simultaneous prediction of the alpha matte and the foreground color. As seen in Figure 2.15, the matting encoder and the context encoder capture both visual features and more global context information. The output of these encoders is combined and serves as input for the decoders that predict the alpha and the foreground color. Feature maps are shared from the foreground decoder to the matting decoder to further enhance the quality of the predicted alpha mattes. Additionally, they investigate more complex loss functions and investigate the impact of using laplacian and feature loss functions over the loss functions first introduces by Deep Image Matting.

Natural Image Matting via Guided Contextual Attention

Another approach to enhance alpha predictions in neural networks was proposed by Li et al. [61]. They propose a guided contextual at-

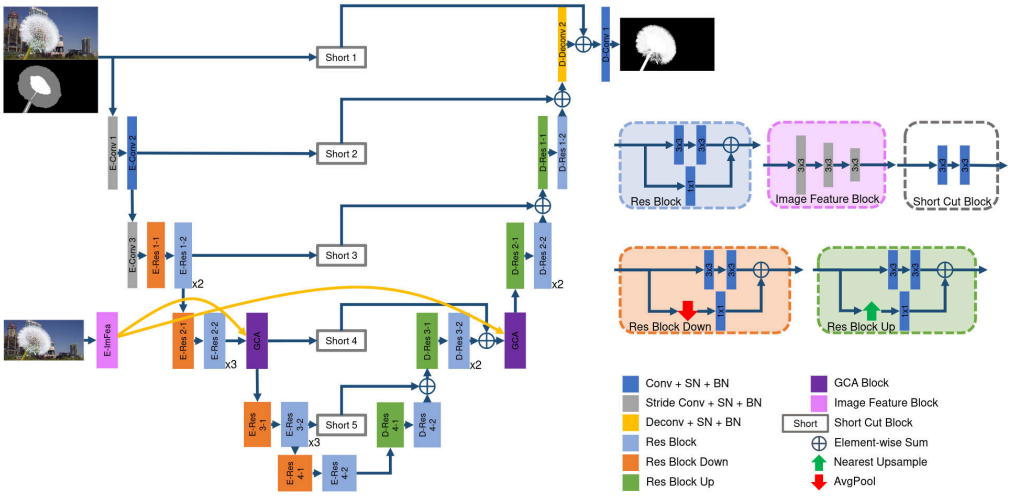

Figure 2.17: Network architecture of GCA Matting. Image taken from [61].

tention module, which propagates high-level opacity information globally based on the learned low-level affinity. Affinity-based methods work on the idea that local regions with almost identical appearance should have a similar opacity, which allows them to propagate alpha values from known regions in the trimap to the unknown region. Based on this, GCA Matting define two different feature flows in their network. The alpha feature flow is equivalent to a standard encoder-decoder network structure, where the alpha features are generated from the RGB + trimap input and flows according to the encoder-decoder structure of the network to output the final alpha matte. By contrast, the low-level image features are generated only from the RGB image and analogous to the local color statistics in conventional affinity-based methods. Given the opacity information from the alpha features and the appearance information from the image features, GCA Matting builds an affinity graph that carries out opacity propagation, which is done in their guided contextual attention module, which can be seen in Figure 2.16.

To calculate the image features, a small auxiliary network with 3

convolution layers of stride $= 2$ is used, which uses only the RGB image as input. For their alpha feature flow, they use a U-net [27] style encoder-decoder. Their combined network can be seen in Figure 2.17 and adds two guided contextual attention module in the 4th block of the encoder and the 4th block of the decoder respectively.

A Late Fusion CNN for Digital Matting

One of the disadvantages of all the previous mentioned matting algorithms is the reliance of a trimap as additional input. Recognizing this, Zhang et al. [62] developed an algorithm that only needs the RGB image as input. They use Densenet [21] as encoder and two decoders, one for the prediction of the foreground and one for the background respectively. This results in foreground and background probability maps, that can be blended into the alpha prediction. To calculate the blending weights, they use a small fusion network consisting of 5 convolutions, which takes the probability maps as well as features from the original image as input. The full network architecture can be seen in Figure 2.18. During training, they use a mix of cross-entropy, L1 loss on the alpha values directly and L1 loss on the gradients of the alpha values to train the decoders. Additionally, they use a weighted L1 loss to train the fusion network to find good blending weights.

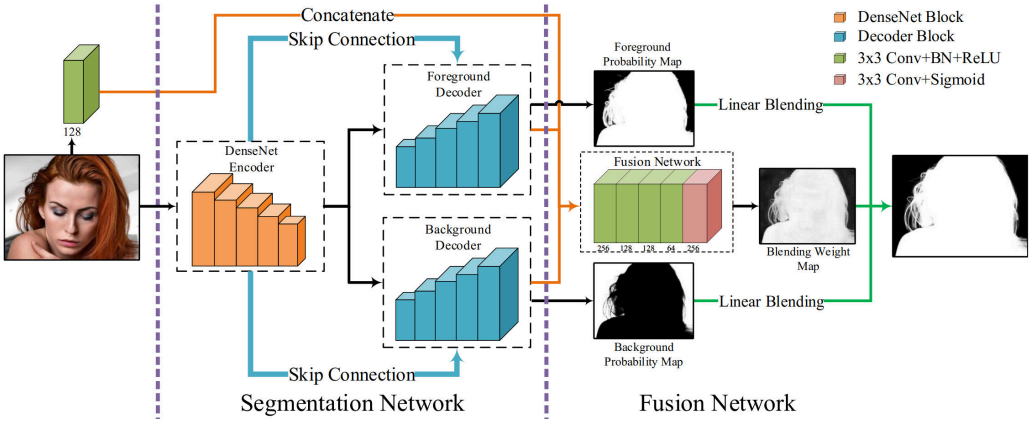

Figure 2.18: Network architecture of Late Fusion Matting. Image taken from [62].

Attention-Guided Hierarchical Structure Aggregation for Image Matting

Building on the approach to disregard trimaps entirely and predict alphas from the RGB image alone, Qiao et al. [63] propose a new attention-guided network to predict alpha mattes directly from input RGB images. Their idea is to reduce unnecessary or redundant semantic information and remove structural details in background regions. To do this, their network consists of 3 parts as can be seen in Figure 2.19. The first part is their feature extraction module, which consists of a ResNeXt [64] convolutional backbone to extract high-level semantic information. These advanced feature maps are then processed by an ASPP [29] module to capture multi-scale semantic features. These features are used by their pyramid features distillation module, which performs channel-wise attention on the pyramidal features from the ASPP to distill adaptive semantic features. Their appearance cues filtration module takes in low-level features from their feature extraction module, which employs spatial attention to filter appearance cues located in the background and enhance cues located in the foreground. Afterwards, the filtered appearance cues and distilled pyramidal features are concatenated to

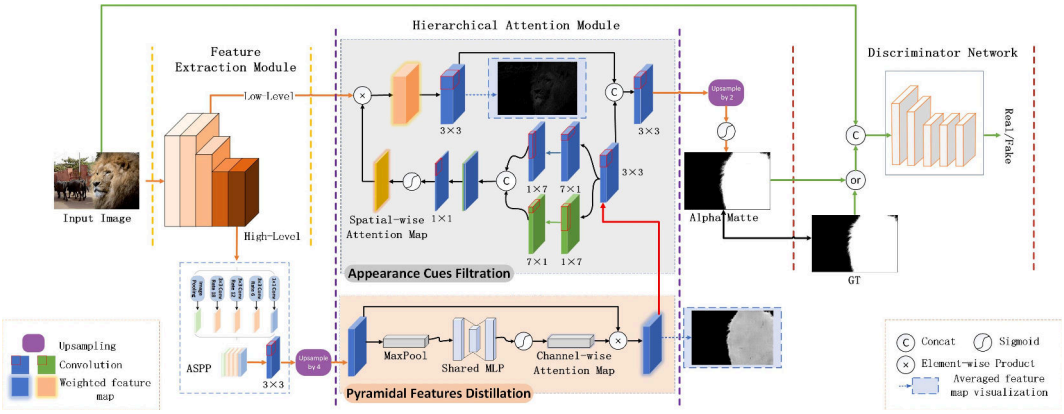

Figure 2.19: Network architecture of HAttMatting. Image taken from [63].

generate the final alpha predictions. During training, they model their network as a GAN, with additional mean-squared-error (MSE) and structural similarity (SSIM) loss directly on the predicted alpha. As an additional contribution, they also release a new matting dataset similar to the dataset released as part of *Deep Image Matting* [54], which consists of 646 distinct foreground objects.

2.4 Video matting

Most recent natural image matting algorithms focus on individual images, but naturally many methods have been designed and adapted for video sequences. Since creating ground-truth data for video sequences is even more cumbersome than for single images, video data containing alpha mattes are even more scarce. The existing video matting dataset will be discussed in Section 2.5. Thus, to the best of our knowledge, there are no learning-based video matting methods. Furthermore, the learning-based methods for single images outperform classical methods to such a degree, that they outperform classical video matting methods even though they were only trained on single images. Therefore only a very brief overview

on classical methods for video matting will be given.

As stated in Section 2.2, video matting methods can be classified into sampling-based and affinity-based methods. However, most algorithms use optical flow as additional input to generate temporal consistent video mattes, such as [65, 66, 67]. For sampling-based methods, it is possible to extend the sampling region to adjacent frames, such as done in [68]. For affinity-based methods, it is possible to extend the affinity matrix over multiple frames such as in [69, 70, 71]. However, all these methods run into computational problems, either through the extra computation of optical flow or increasingly large sampling sets or affinity matrices that may lead to intolerable time and memory consumption. Some other methods [72, 73] divide the image or video sequence into patches and assure spatial and temporal consistency through multi-frame graph models. Related to these methods is *Video SnapCut* [74], which aims to cut out the foreground object in a video by propagating segmentation masks across frames and refining them to alpha mattes.

In general, video matting can be considered a special case of video-to-video synthesis. This more general problem is an active research problem and has seen many improvements over the last years. Methods such as [75, 76] can achieve high-quality results on a variety of tasks. However, these methods need to be trained on datasets of sufficient size, which makes them unusable for video matting unless a large dataset is released. Another possible way to enforce temporal consistency is to use blind video consistency methods, which are designed to work as post-processing methods on any kind of video with individually processed frames to reduce temporal artifacts such as flickering. The current state-of-the-art in this field is proposed by Lei et al. [77]. Their work leverages a *Deep Video*

Prior by training a convolutional neural network with the assumption that the output of the CNN for corresponding patches in the video frames should be consistent. In their method they train a U-Net [27] directly on the test video and do not need to train on a large scale dataset. Their network learns to mimic the original function that transforms the unprocessed video to the processed output using a perceptional loss function [39], but stops before artifacts are overfitted. They show in a variety of computer vision task that this method outperforms the previous state-of-the-art.

2.5 Natural image matting datasets

Natural image matting is a difficult problem not only due to its under-constrained nature, but also because it is difficult to generate a large amount of training data. For many computer vision tasks such as image inpainting or super-resolution, ground-truth data can be generated automatically. For others, such as classification or segmentation, the ground-truth has to be labelled manually. Natural image matting is a problem of the latter case, but it requires much more expertise to generate ground-truth labels by hand. Since the whole point of predicting an alpha matte is to predict the opacity values of even very fine structures, it takes time and a high attention to detail for the manual creation of ground-truth alphas.

For many years, the only publicly available dataset was released as part of the alphamatting.com online benchmark [53]. However, this dataset only contains 27 training images and 8 test images that are not publicly available. In 2018, however, Adobe released a new dataset as part of the *Deep Image Matting* paper by Xu et al. [54].

This new dataset contains 493 unique foreground objects in the training split and 50 in the test set. All these images come from natural images with a plain background that have been carefully manually processed in Photoshop to create a ground-truth alpha matte and pure foreground colors as can be seen in Figure 2.20. Given the alpha and the foreground colors, it is possible to select a random background image from MS COCO [78] and Pascal VOC [79] and composite a completely new image. This allows for heavy data augmentation, since each foreground object can essentially be used to compose an unlimited amount of new images. To create the corresponding trimaps for each image, the ground-truth alpha can be randomly dilated. A concern of the composited nature of the dataset is that a network could learn to differentiate between image characteristics of the foreground and background images, such as different lighting conditions, noise, etc. Another concern is the unrealistic nature of some of the newly composited images, since the foreground object is composited onto a new background without any context, as can be seen in Figure 2.21. This can lead to images where the foreground object is floating in the middle of a scene, which is something that would not normally happen in real images. However, an advantage of the composited images is that some backgrounds can have colors that are very close to the colors of the foreground object and very complex textures, which makes the dataset quite challenging. In the end, Xu et al. [54] and subsequent publications that use the dataset for training show experimentally that superior results can be achieved by using it for training. Following Xu et al., Qiao et al. [63] recently released an additional dataset build the same way that consists of 646 distinct foreground objects.

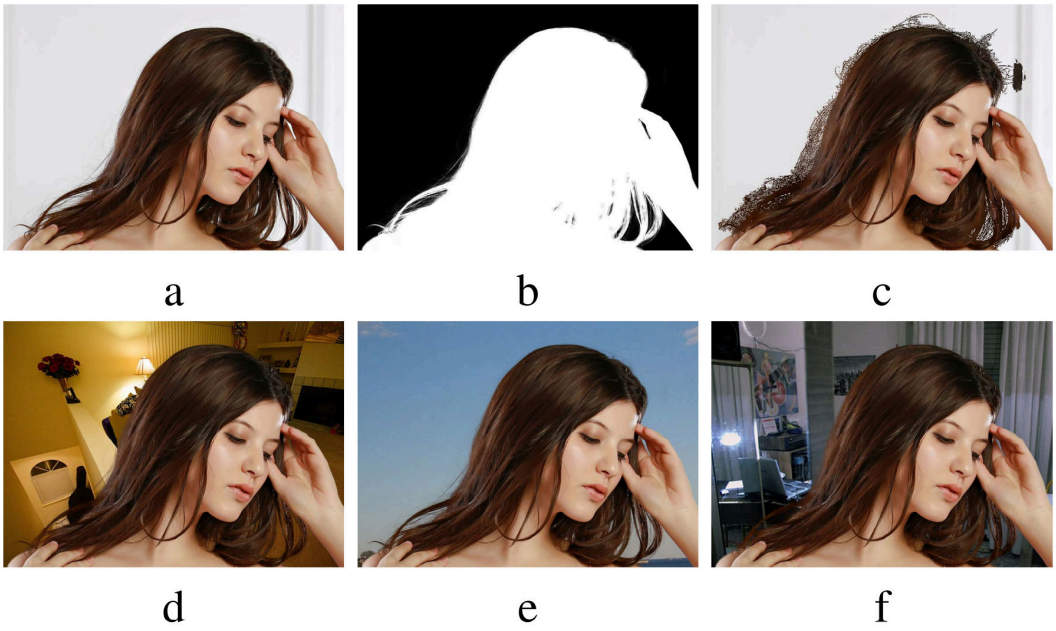

Figure 2.20: Dataset creation. (a) Input image with simple background, (b) The manually extracted alpha matte, (c) The manually extracted foreground colors, (d-f) New composited using various background images. Image taken from [54].

Figure 2.21: Examples of non-realistic images introduced in the Composition-1k test dataset.

2.5.1 Video matting dataset

Similar to the alphamatting.com online benchmark [53], there exists an online benchmark for video matting: videomatting.com [80]. This dataset consists of 5 video sequences of moving objects captured in front of a green-screen and 7 sequences captured using a stop-motion technique. 3 of these sequences are available as public training data, containing 150, 285 and 150 frames respectively. Trimaps are provided for each frame and are dilated from the ground-truth alpha mattes. In the case of the green-screen captures, chroma keying is used to extract the alpha matte and foreground colors for

Figure 2.22: Alpha mattes from chroma keying and stop-motion capture for the same image region. The stop-motion result is significantly better at preserving details. Image taken from [80].

the sequences. For the stop-motion captures, a fuzzy toy is placed in front of a LCD monitor. The toy is rotated along a predefined trajectory and a digital camera takes a picture after each discrete step. Afterwards, the toy is removed and all the background is captured. A comparison of captured alpha mattes using chroma keying and stop-motion can be seen in Figure 2.22

2.6 Related applications

The natural image matting methods that have been mentioned so far have all been general methods, i.e. methods that are designed to predict the alpha matte for any foreground object. However, it is easy to imagine how constraining this assumption could lead to performance improvements in special cases. The following methods are all different from the previously mentioned, in that they either only work in special cases, are designed for constrained environments (e.g. to work on mobile phones) or need additional inputs. They are relevant for an overview of the whole field, but not technically relevant to any of the methods discussed in the following chapters and therefore not much detail about them will be given.

To serve the interests of casual creatives, it is important for mat-

ting algorithms to work on mobile devices. This naturally limits the available computation resources and special care has to be taken to design the algorithm as efficient as possible. Levinshtein et al. [81] design a neural network based on Mobilenet [82] for hair matting on mobile devices, which allows users to automatically recolor hair in real-time.

Similarly, Zhu et al. [1] proposed an automatic portrait matting algorithm for mobile phones. They first predict a coarse binary mask of the portrait in the image, followed by an edge-preserving and matting adaptive feathering block. This architecture allows the method to run at ~ 15 FPS on mobile phones.

Following this, Seo et al. [83] proposed a new method outperforming Zhu et al. in both computation time and gradient error of the predicted alphas by using a network architecture build on dilated convolutions and bottleneck blocks.

Instead of focusing on mobile devices, Shen et al. [2] propose a method for automatic portrait matting. Their method automatically predicts a trimap from which they predict the final alpha matte.

Liu et al. [84] propose a method to handle the lack of available training data for human matting by making use of coarse annotated data. They train a mask prediction network on the coarse data to predict coarse semantic masks, followed by a quality unification network to unify the quality of the previous coarse mask predictions. Afterwards a matting refinement network uses this data to predict the final alpha matte.

Chen et al. [85] focus their proposed method on transparent objects. They formulate transparent object matting as a refractive flow estimation problem and train a network to learn the refractive flow. They also release a dataset of images containing transparent

objects.

Shin et al. [86] adapt the network proposed by Xu et al. [54] for the matting of garments. They replace the refinement network originally proposed by a recursive convolutional network.

Similar to the methods introduced in 2.3, Sengupta et al. [87] propose a method to automatically predict the alpha matte from any kind of image. However, they require an additional image of the background without the subject in the image as additional input. They first train their network in a supervised manner, followed by training another matting network guided by the first network and by a discriminator that judges the quality of composites. Similar to [60] they output both the foreground color and the alpha of the foreground object.

As the final method presented in this section, Zou et al. [88] propose a cloud detection and removal algorithm as a mixed energy separation between foreground and background images. They propose a GAN to achieve weakly supervised matting of cloud images.

2.7 Conclusion

In this chapter, we introduced the background and related methods relevant to natural image matting. As with many computer vision tasks, CNNs have proven to outpace classical methods in performance to a large degree. Therefore, we introduced some of the seminal deep learning papers that establish the foundation on which the current state-of-the-art is build upon in addition to the recent deep learning methods themselves. Aside from a general increase in performance, modern methods contribute additional advantages such as speed of computation and increasingly less reliance on trimaps.

However, most recent methods still ignore the need of precise foreground color estimation for high-quality composites and focus on the matting task exclusively. There have also been no recent video matting methods proposed, most likely due to the scarcity of training data. While the current state-of-the-art in natural image matting outperforms classical video matting approaches in video sequences, the individual processing of frames often introduces temporal inconsistencies that negatively impact video matting performance. Aside from the natural image matting task, many new methods have been proposed in recent years that aim to solve matting in more constrained cases, such as matting on mobile devices or portrait and garment matting. While research in this field has made great strides over the recent years, natural image matting methods can still not compete with time consuming solutions made by artists using professional image editing software and we foresee no decline in research activity in the next few years.

Chapter 3

Generative adversarial networks for natural image matting

The first contribution of this thesis will be presented in this chapter. By the time our method was published, the only published learning-based natural image matting algorithms were *DCNN Matting* [52] and *Deep Image Matting* [54], which nevertheless outperformed all the classical approaches. An overview of both these networks can be found in Section 2.3. This chapter is organized as follows. First, a small introduction is given, outlining issues and weaknesses with the then state-of-the-art. Next, our method is described, detailing the dataset and data augmentation strategy used, our network structure and training pipeline, as well as all training details. Afterwards, an evaluation of our method is given, comparing against the then state-of-the-art. Finally, we give a conclusion discussing the weaknesses and limitations of our method and potential improvements.

3.1 Motivation

With their *Deep Image Matting* [54] method, Xu et al. managed to dramatically outperform the previous state-of-the-art in natural image matting. Naturally though, no method is perfect and we identify two potential improvements to be made. First, their network is kept relatively simple, consisting of *VGG16* [17] as encoder and a smaller decoder without any skip-connections. We believe improvements on this structure can lead to better results. Further, the alpha prediction results from their encoder-decoder lead to overly smooth results, which Xu et al. try to alleviate through a second refinement stage. We believe an updated loss function by modelling the problem as a generative adversarial network (GAN) will lead to sharper alpha predictions.

Our Contribution. We propose a generative adversarial network for natural image matting. We improve on the network architecture of Xu et al. [54] to better deal with the spatial localization issues inherent in CNNs by using dilated convolutions to capture global context information without downscaling feature maps and losing spatial information. Furthermore, we improve on the decoder structure of the network and use it as the generator in our generative adversarial model. The discriminator is trained on images that have been composited with the ground-truth alpha and the predicted alpha and therefore learns to recognize images that have been composited well, which helps the generator learn alpha predictions that lead to visually appealing compositions.

3.2 Method

To tackle the problem of image matting, we use a generative adversarial network. The generator of this network is a convolutional encoder-decoder network that is trained both with help of the ground-truth alphas as well as the adversarial loss from the discriminator. We detail our network in more detail in the following sections.

3.2.1 Training dataset

Deep learning approaches need a lot of data to generalize well. Large datasets like Imagenet [89] and MSCOCO [78] have helped tremendously in this regard for several computer vision tasks. One of the problems of natural image matting, however, is that it is significantly more difficult to collect ground-truth data than for most other tasks. The quality of the ground-truth also needs to be very high, because the methods need to capture very fine differences in the alpha to provide good results. Thankfully a new matting dataset [54] consisting of 431 unique foreground objects and their corresponding alpha has recently been published. This dataset has finally made it possible to train deep networks such as ours. Nevertheless, 431 images is not enough to train on their own, so we enhance the dataset in the following way, similar to how Xu et al. [54] propose in their approach:

For every foreground object, a random background image from MSCOCO is taken, which allows us to composite a new unique image out of the foreground, the provided ground-truth alpha and the background image. For further data augmentation, we randomly rotate the foreground and alpha by n degrees, sampled from a normal distribution with a mean of 0 and standard deviation of 5. We

then generate a trimap by dilating the ground-truth alpha with random kernel sizes from 2 to 20. Next, we randomly crop a rectangular part of the foreground, alpha, trimap and background images, centered on some pixel within the unknown region of the trimap of a size chosen randomly from 320×320 to 720×720, and resize it to 320×320. This allows the network to be more scale invariant. Finally, we randomly flip the cropped images to get the final foreground, alpha, trimap and background images, which will be used to composite a new image as part of the training process.

3.2.2 Network architecture

Xu et al. [54] have recently shown that it is possible to train an encoder-decoder network with their matting dataset to produce state-of-the-art results. We build on their approach and trained a deep generative adversarial network on the same dataset. Our AlphaGAN architecture consists of one generator G and one discriminator D. G takes an image composited from the foreground, alpha and a random background appended with the trimap as 4th-channel as input and attempts to predict the correct alpha. D tries to distinguish between *real* 4-channel inputs and *fake* inputs where the first 3 channels are composited from the foreground, background and the predicted alpha. The full objective of this network can be seen in 3.2.3.

Generator

Our generator consists of a an encoder-decoder network similar to those that have achieved good results in other computer vision tasks, such as semantic segmentation [31] [90]. For the encoder, we take the Resnet50 [20] architecture, pretrained on Imagenet [89]

and convert the convolutions in the 3rd and 4th Resnet blocks to dilated convolutions with rate 2 and 4 respectively for a final output stride of 8, similar to Chen et al. [30]. Since the training inputs are fixed to a size of 320×320, this leads to a feature map size of 40×40 in the final feature map of Resnet block 4. Even though the feature maps are downsampled less often, the dilated convolutions can still capture the same global context of the original Resnet50 classification network, while not losing as much spatial information. After the Resnet block 4, we add the atrous spatial pyramid pooling (ASPP) module from [30] to resample features at several scales for accurately and efficiently predicting regions of an arbitrary scale. We then feed the output of the ASPP to the decoder part of the network. We also change the first layer of the network slightly to accommodate our 4-channel input by initializing the extra channel in the convolution layer with zeros.

The decoder part of the network is kept simple and consists of several convolution layers and skip connections from the encoder to improve the alpha prediction by reusing local information to capture fine structures in the image [35]. First, the output of the encoder is bilinearly upsampled 2 times so that the feature maps have the same spatial resolution as those coming from Resnet block 1, which have an output stride of 4. The final feature map from block 1 is fed into a 1×1 convolution layer to reduce the number of dimensions and then concatenated with the upsampled feature maps from the encoder. This is followed by three 3×3 convolutions that steadily reduce the number of dimensions to 64. The saved pooling indices from the max-pooling layer in the encoder are used to upsample these feature maps to an output stride of 2, where they are concatenated again with the feature maps of the same resolu-

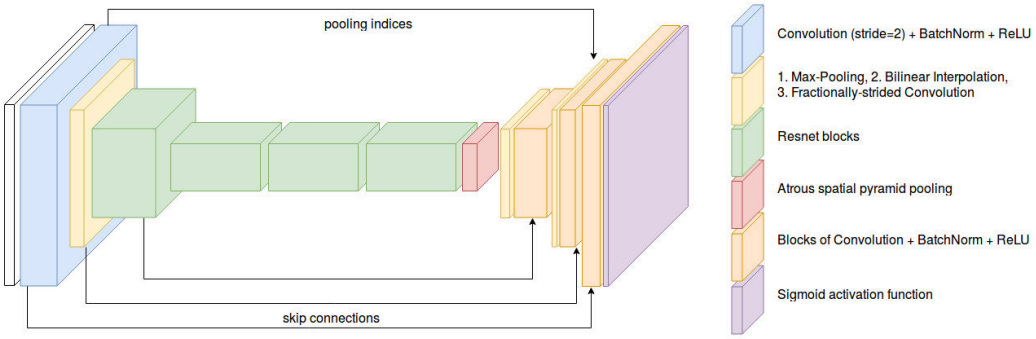

Figure 3.1: The generator is an encoder-decoder network with skip connections.

tion from the encoder, followed by some convolution layers. Finally, the feature maps are upsampled again using fractionally-strided convolutions, concatenated with the RGB input image and fed to a final set of convolution layers. All of these layers are followed by ReLU activation functions and batch-normalization layers [34], except the last one, which is followed by the sigmoid activation function to scale the output of the generator between 0 and 1, as needed for an alpha prediction (See Figure 3.1). A table detailing all layers in the network can be seen in Table 3.1.

Discriminator

For the discriminator in our network, we use the PatchGAN introduced by Isola et al. [35]. This discriminator attempts to classify every $N \times N$ patch of the input as real or fake. The discriminator is run convolutionally over the input and all responses are averaged to calculate the final prediction of the discriminator D.
PatchGAN was designed to capture high-frequency structures and assumes independence between pixels that cannot be located in the same $N \times N$ patch. This suits the problem of alpha prediction, since the results of the generator trained only on the alpha-prediction loss can be overly smooth, as noted in [54]. The discriminator helps

to alleviate this problem by forcing the generator to output sharper results. To help the discriminator focus on the right areas of the input and to guide the generator to predict alphas that would result in good compositions, the input of D consists of 4 channels. The first 3 channels consist of the RGB values of a newly composited image, using the ground-truth foreground, a random background and the predicted alpha. The 4th channel is the input trimap to help guide the discriminator to focus on salient regions in the image. We found that for our network $N = 70$ is sufficient to balance good results and a low amount of parameters and running time of D.

Encoder			**Decoder**		
layer name	output size	filter size	layer name	output	filter size
conv1	160×160	$7 \times 7, 64$, stride 2	bilinear	80×80	bilinear upsampling
conv2_x	80×80	3×3 max pool, stride 2 $\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$	deconv1_x	80×80	skip from conv2_x, $1 \times 1, 48$ $\begin{bmatrix} 3 \times 3, & 256 \\ 3 \times 3, & 128 \\ 3 \times 3, & 64 \end{bmatrix}$
conv3_x	40×40	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 512 \end{bmatrix} \times 4$	unpooling	160×160	2×2 unpool, stride 2
conv4_x	40×40	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \quad r = 2 \\ 1 \times 1, & 1024 \end{bmatrix} \times 6$	deconv2_x	320×320	skip from conv1_x, $1 \times 1, 32$ $\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64, \text{ stride} \frac{1}{2} \\ 3 \times 3, & 32 \end{bmatrix}$
conv5_x	40×40	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \quad r = 4 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$	deconv3_x	320×320	skip from RGB image $\begin{bmatrix} 3 \times 3, & 32 \\ 3 \times 3, & 32 \end{bmatrix}$
aspp	40×40	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, r = 6, & 256 \\ 3 \times 3, r = 12, & 256 \\ 3 \times 3, r = 18, & 256 \\ \text{Image Pooling}, & 256 \end{bmatrix}$	deconv4_x	320×320	$3 \times 3, \quad 1$

Table 3.1: Architecture of the proposed generator. The encoder consists of the standard Resnet50 architecture with the last two layers removed and ASPP [30] module added to output 256 40×40 feature maps. The decoder is kept small and uses bilinear interpolation, unpooling and fractionally-strided convolution to upsample the feature maps back to 320×320. For the max-pooling operation in the encoder, the maximum indices are saved and used in the unpooling layer. All convolutional layers except the last one are followed by batch-normalization layers [34] and ReLU activation functions. The last convolutional layer is followed by a sigmoid activation function to scale the output between 0 and 1. r is the dilation rate of the convolution. The default stride or dilation rate is 1. Skip connections are added to retain localized information.

3.2.3 Network objectives

The goal of our networks is to predict the true alpha of an image, given the trimap. In their paper Xu et al. [54] introduce two loss functions specifically for the problem of alpha matting, the alpha-prediction loss $\mathcal{L}_{alpha}$ and the compositional loss $\mathcal{L}_{comp}$:

$$\begin{aligned} \mathcal{L}_{alpha} &= \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \sqrt{(\alpha_p^i - \alpha_g^i)^2}, \qquad \alpha_p^i, \alpha_g^i \in [0,1]. \\ \mathcal{L}_{comp} &= \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \sqrt{(c_p^i - c_g^i)^2}, \qquad c_p^i, c_g^i \in [0,1]. \end{aligned} \tag{3.1}$$

where α_p^i is the output prediction at pixel i, α_g^i the corresponding ground-truth, c_p^i is the RGB value at pixel i of a newly composited image out of the ground-truth foreground, ground-truth background and predicted alpha, c_g^i the corresponding RGB value composited from the ground-truth alpha, and $\mathcal{U}$ the unknown region of trimap. Additionally to these, we also use the adversarial loss [91] $\mathcal{L}_{GAN}$, which is defined as:

$$\mathcal{L}_{GAN}(G, D) = \log D(x) + \log\left(1 - D(C(G(x)))\right) \tag{3.2}$$

where x is a *real* input: an image composited from the ground-truth alpha and foreground appended with the trimap. $C(y)$ is a composition function that takes the predicted alpha from G as an input and uses it to composite a *fake* image. G tries to generate alphas that are close to the ground-truth alpha, while D tries to distinguish *real* from *fake* composited images. G therefore tries to minimize $\mathcal{L}_{GAN}$ against the discriminator D, which tries to maximize it. The above losses are combined and lead to the full objective of our

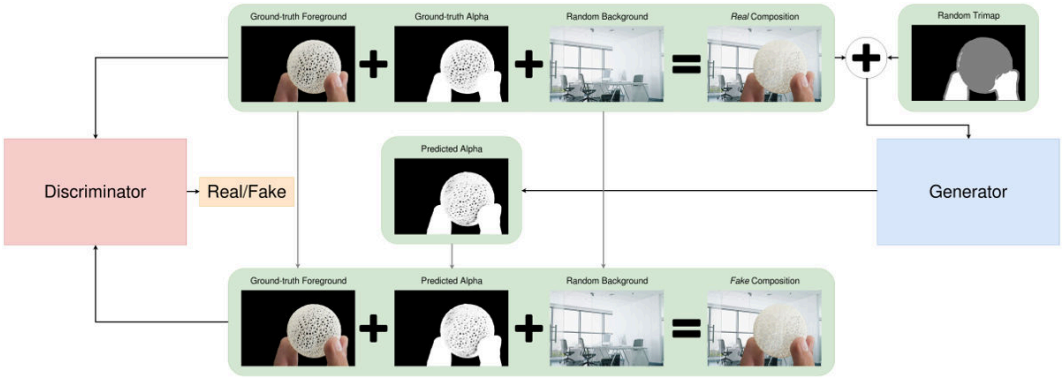

Figure 3.2: Overview over the objective pipeline. Every training iteration, a new batch of training images is composited. These training images are processed by the generator to predict their respective alpha matte. This predicted alpha is then used to composite a new batch of images, using the same foreground and background. These *real* and *fake* compositions are used to train the discriminator, while the generator is trained to fool the discriminator.

network:

$$\mathcal{L}_{AlphaGAN}(G, D) = \mathcal{L}_{alpha}(G) + \mathcal{L}_{comp}(G) + \mathcal{L}_{GAN}(G, D) \tag{3.3}$$

where we aim to solve $\arg\min_G \max_D \mathcal{L}_{AlphaGAN}$. An overview over the objective pipeline is shown in Figure 3.2.

3.3 Results

In this section, we evaluate our approach on two datasets. The first one is the well-known alphamatting.com [53] evaluation benchmark, which consists of 28 training images and 8 test images. For each set, three different sizes of trimaps are provided, namely, ”small”, ”large” and ”user”. The second one is the Composition-1k dataset [54], which includes 1000 test images composed from 50 unique foreground objects. We evaluate the quality of our results using the well known sum of absolute differences (SAD) and mean square error (MSE) but also the gradient and connectivity errors, which mea-

sure the matting quality as perceived by the human eye [53]. To avoid deviations from the original formulation of the metrics, as seen in other works ([1], [2]), we make use of the publicly available evaluation code provided by [54]. We use the default values for the gradient and connectivity error as proposed by the original authors of the evaluation metrics [53] throughout all our experiments.

3.3.1 Evaluating the network architecture

The Composition-1k test dataset consists of 1000 images composited out of 50 unique objects. However, since random background images are chosen when compositing, the resulting images do not look realistic, in the sense that they show scenes that do not exist in nature, e.g. a glass in the foreground floating before a woodland scene as the background. Furthermore, the foreground and background images also have different characteristics, like lighting, noise, etc., that lead to images that cannot be considered natural. Therefore, we mainly used the Composition-1k dataset to test our network architecture. We started by using a similar encoder-decoder architecture as [54], but replaced VGG16 [17] as encoder for Resnet50 [20]. We also tried other Resnet architectures, but found that Resnet50 performed best. By including the atrous pyramid pooling module (ASPP) [30] and using dilated convolutions for an output stride of 8 for the encoder, we further improved our performance. We also tried a multi-grid approach following [90], but found that this did not lead to better results. Finally, we added skip connections from the 1st and 2nd Resnet blocks and the RGB input to get to the final model we use for the generator. A comparison of some of the different network architectures that we tried are shown in Table 3.2. Additionally, we compare several top matting methods where there

ASPP	OS=8	OS=16	MG	Skip	GAN loss	MSE
					✓	0.049
✓	✓				✓	0.033
✓		✓			✓	0.038
✓	✓		✓		✓	0.039
✓	✓			✓	✓	**0.031**
✓	✓			✓		0.041

Table 3.2: Comparison of different generator architectures. **ASPP**: Atrous spatial pyramid pooling [30], **OS**: Output stride of the final feature map, **MG**: Multi-Grid for dilated convolutions [90], **Skip**: Skip connections from the encoder, **GAN loss**: Additional adversarial loss during training.

Method	**SAD**	**MSE**	**Gradient ($\times 10^4$)**	**Connectivity ($\times 10^4$)**
Shared Matting (SM) [43]	117.0 (68.7)	0.067 (0.032)	10.1 (5.1)	5.4 (5.2)
Comprehensive Sampling (CM) [44]	56.5 (53.7)	0.032 (0.030)	**3.4** (4.0)	5.7 (5.4)
KNN Matting (KNN) [92]	99.0 (53.6)	0.070 (0.030)	6.2 (4.0)	8.5 (5.4)
DCNN Matting (DCNN) [52]	155.8 (68.8)	0.083 (0.032)	11.5 (5.1)	7.3 (6.0)
Three-layer Graph (TLGM) [93]	106.4 (52.4)	0.066 (0.030)	7.0 (3.9)	5.0 (4.3)
Information-flow Matting (IF) [50]	75.4 (52.4)	0.066 (0.030)	6.3 (3.8)	7.5 (5.3)

Table 3.3: Quantitative results on the Composition-1k dataset. Our results are shown in parenthesis. We achieve better results than all the tested methods, with the sole exception marked in bold.

is public code available with our approach on the Composition-1k dataset [54]. For all methods, the original code from the authors is used, without any modifications. It was found that there were multiple failed cases when directly applied to the entire dataset. We believe that this is due to the inherently unrealistic nature of the dataset (see the appendix B.1.1 for examples). To overcome this issue, we only provide comparison for results in which the images successfully produced a valid matting prediction. In contrast, our method succeeded in all images in the dataset. Quantitative results under all metrics are shown in Table 3.3. Our method delivers noticeably better results than the other approaches. The gradient error from the comprehensive sampling approach [44] is the only case where we do not achieve the best result as shown in Table 3.3. Some comparisons of results for this dataset can be seen in Figure 3.3. Additional results are provided in the appendix B.1.1.

3.3.2 The alphamatting.com dataset

We submitted our results on the alphamatting.com benchmark [53] achieving state-of-the-art results for the Troll and Doll images, both for the SAD and MSE evaluation metrics and first place overall on the gradient evaluation metric. Even though we are not first in the SAD or MSE, our results are numerically very close to the top-performing results for the remaining images as shown in Table 3.4. Overall, we achieve very visually appealing results, as seen in Figure 3.4, further supported by our results in the gradient metric, which was introduced in [53]. This metric is shown to have a high correlation to the human perception of good alpha mattes. Similar to [46] we do not report the connectivity measure since it is not robust [53].

Our best results are for the Troll and Doll images, which is due to the ability of our approach to correctly predict the alpha values for very fine structures, like the hair. This is where the adversarial loss from the discriminator helps, since the discriminator is able to capture high-frequency structures and can distinguish between overly smooth predictions and ground-truth compositions during training, which allows the generator to learn to predict sharper structures. Our worst results come from the Net image. However, even though we appear low in the rankings for this image, we believe that our results still look very close to the top-performing approaches. Some examples of the alphamatting results are shown in Figure 3.4.

3.4 Conclusion

In this paper we proposed a novel generative adversarial network architecture for the problem of natural image matting. To the best

	Average Rank*				Troll			Doll			Donkey			Elephant			Plant			Pineapple			Plastic Bag			Net		
	Overall	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U
Sum of Absolute Differences																												
DI [54]	4.6	5.6	3.6	4.6	10.7	11.2	11.0	4.8	5.8	5.6	**2.8**	**2.9**	**2.9**	1.1	**1.1**	2.0	6.0	7.1	8.9	2.7	**3.2**	3.9	19.2	19.6	18.7	21.8	23.9	24.1
IF [50]	5.4	6.5	4.9	4.8	10.3	11.2	12.5	5.6	7.3	7.3	3.8	4.1	3.0	1.4	2.3	2.0	5.9	7.1	8.6	3.6	5.7	4.6	18.3	19.3	15.8	20.2	22.2	22.3
DCNN [52]	6.8	8.6	4.9	7.0	12.0	14.1	14.5	5.3	6.4	6.8	3.9	4.5	3.4	1.6	2.5	2.2	6.0	6.9	9.1	4.0	6.0	5.3	19.9	19.2	19.1	19.4	20.0	21.2
Ours	7.8	8.6	7.5	7.4	**9.6**	**10.7**	**10.4**	**4.7**	**5.3**	**5.4**	3.1	3.7	3.1	1.1	1.3	2.0	6.4	8.3	9.3	3.6	5.0	4.3	20.8	21.5	20.6	25.7	28.7	26.7
TLGM [93]	11.5	8.1	8.9	17.6	10.7	15.2	13.8	4.9	5.6	8.1	3.9	4.4	3.6	**1.0**	1.8	3.0	5.9	7.3	12.4	4.2	8.0	8.5	24.2	25.6	24.2	20.5	23.5	22.2
Gradient																												
Ours	9.3	8.0	6.8	13.3	0.2	0.2	0.2	0.2	0.2	0.3	**0.2**	0.3	0.3	0.2	**0.2**	0.4	1.8	2.4	2.7	1.1	1.4	1.5	0.9	1.1	1.0	0.5	0.5	0.6
DCNN [52]	10.9	13.6	10.4	8.8	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.3	0.3	0.4	0.4	1.5	1.5	2.1	1.1	1.3	1.5	1.5	1.4	1.0	0.6	0.6	0.5
DI [54]	11.4	8.1	8.4	17.6	0.4	0.4	0.5	**0.2**	0.2	**0.2**	**0.1**	**0.1**	0.2	0.2	0.2	0.6	1.3	1.5	2.4	0.8	0.9	1.3	**0.7**	**0.8**	1.1	0.4	0.5	0.5
IF [50]	12.5	15.1	10.1	12.1	0.2	**0.2**	0.2	0.2	0.2	0.4	0.4	0.4	0.4	0.3	0.4	0.4	1.7	1.8	2.2	0.9	1.3	1.3	1.5	1.4	0.8	0.5	0.6	0.5
TLGM [93]	14.6	11.6	11.8	20.5	**0.2**	0.2	0.2	0.2	0.2	0.4	0.3	0.4	0.3	**0.1**	0.3	0.5	1.6	1.7	2.7	1.1	1.9	2.4	1.6	1.6	1.0	0.5	0.6	0.4

Table 3.4: SAD and gradient results for the top five methods on the alphamatting.com dataset. Best results are shown in bold.

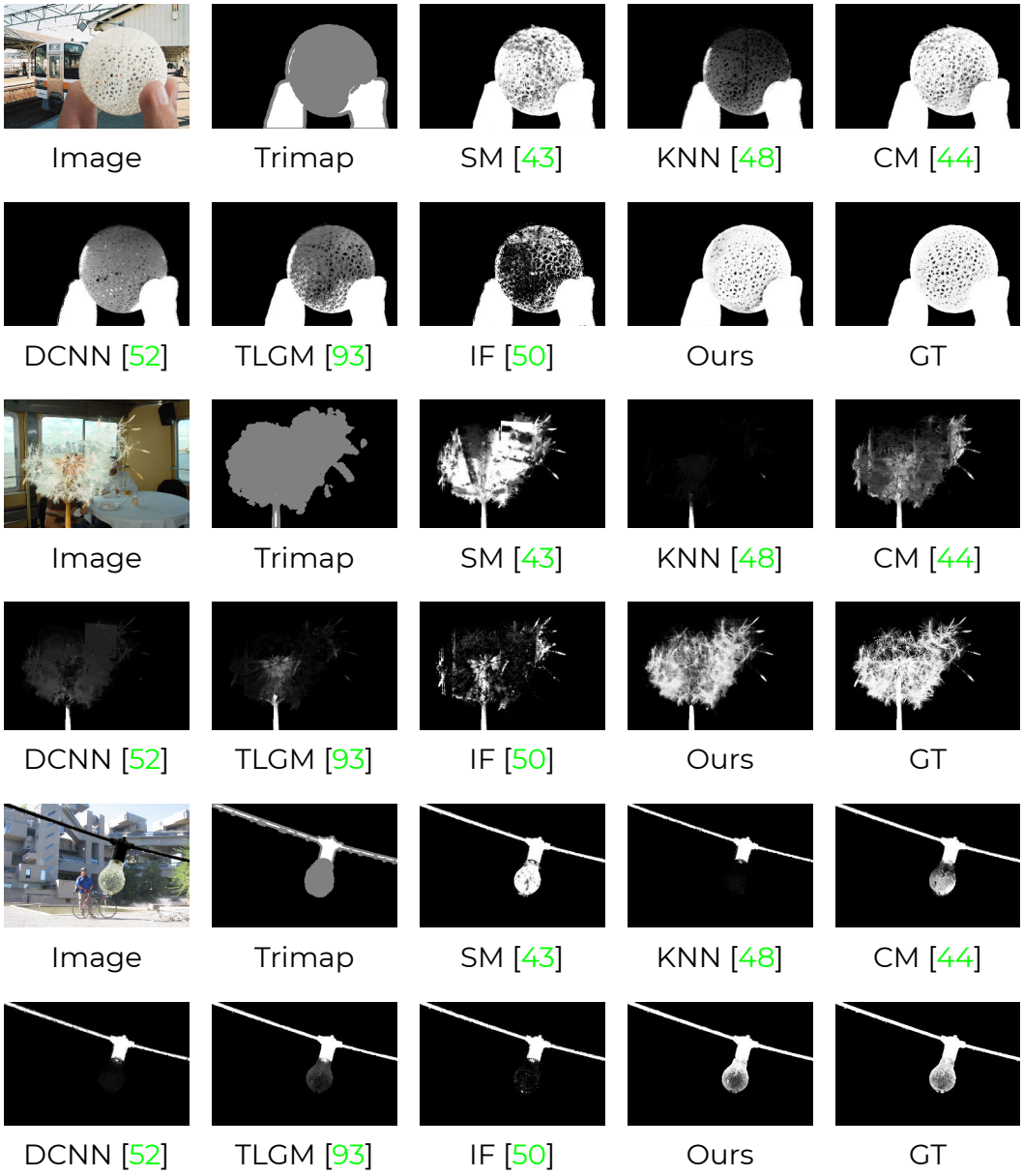

Figure 3.3: Comparison of results on the Composition-1k testing dataset.

of our knowledge, this is the first work that uses GANs for this computer vision task. Our generator is trained to predict alpha mat-

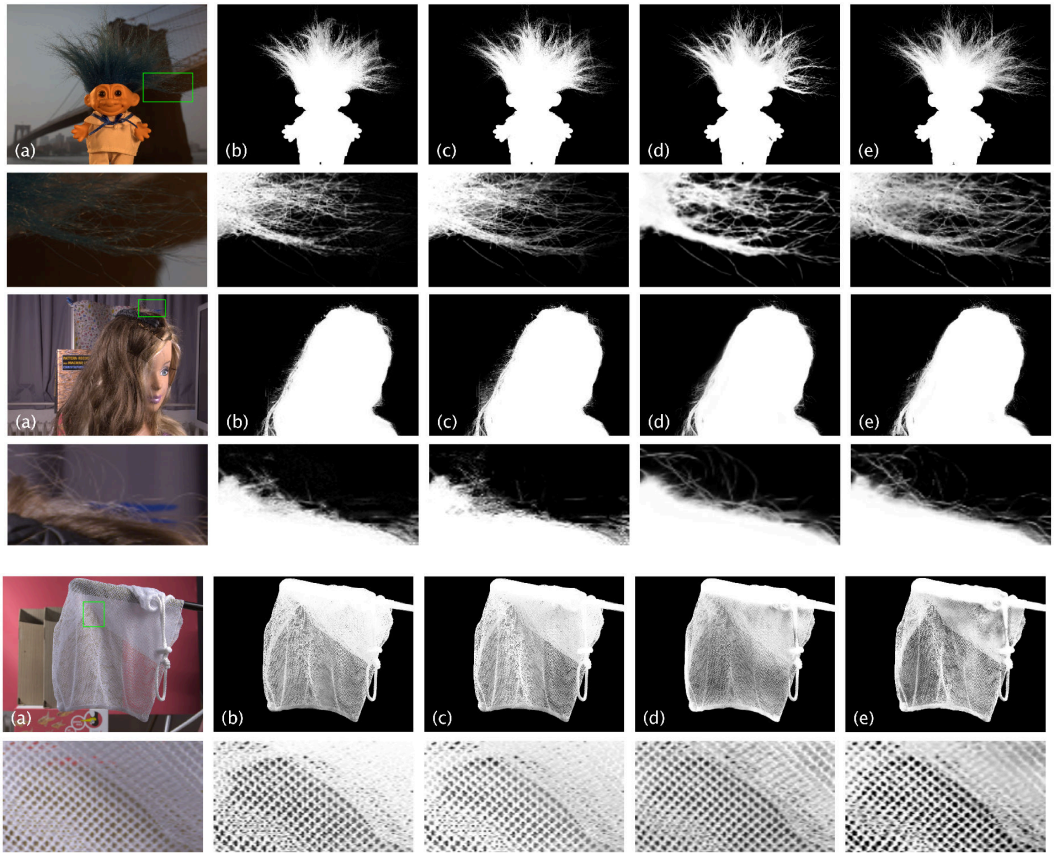

Figure 3.4: Alpha matting predictions for the "Troll" and "Doll" images (our best results) and the "Net" image (our worst result) taken from the alphamatting.com dataset. From left to right: a) Input image, b) DCNN [52], c) IF [50], d) DI [54], e) Ours.

tes from input images while the discriminator is trained to distinguish good images composited from the ground-truth alpha from images composited with the predicted alpha. Additionally, we introduce some network enhancements to the generator that have been shown to give an increase in performance for the task of semantic segmentation. These changes allow us to train the network to predict alphas that lead to visually appealing compositions, as our results in the alphamatting.com benchmark show. Our method ranks first in this benchmark for the gradient metric, which was designed as a perceptual measure. For all the other metrics we show comparable results to the state-of-the-art and are first in the SAD

and MSE errors for the Troll and Doll images. Our results in these images especially manage to capture the high-frequency hair structures, which might be attributed to the addition of the adversarial loss during training. Additionally, we compare with publicly available methods on the Composition-1k test dataset and achieve state-of-the-art results.

Chapter 4

Foreground color prediction through inverse compositing

In this chapter the second contribution of this thesis is presented. This chapter is organized as follows. First, a small introduction is given, outlining issues and weaknesses with the current state-of-the-art. Next, our method is described, including our network structure and training pipeline, as well as all training details. Afterwards, an evaluation of our method is given, comparing against the state-of-the-art. Finally, we give a conclusion discussing the weaknesses and limitations of our method and potential improvements.

4.1 Motivation

In recent years, natural image matting algorithms have achieved a level of performance that make them very suitable to use by casual creatives for image and video editing or compositing. While not yet up to the standards of high-quality movie productions, the predicted alphas from state-of-the-art algorithms are sufficiently accurate to allow the extraction of the foreground object from an image and composition of a new image using a different background. However, there are still issues when using these algorithms in practise. First of all, if the predicted alpha is used for extracting the foreground object from an image to composite a new one, the alpha alone is often not enough to create a high quality composite even with a perfect alpha prediction. This is due to the mixture of foreground and background colors in the transparent regions of the object. The background color of the original image bleeding through will be extracted as well and can lead to a disparity of colors in the new composite in many cases, as seen in Figure 4.1. For high quality composites, it is therefore necessary to also estimate the foreground color of the object. The second problem with current matting methods is that they do not allow for much user interaction if the prediction is not quite satisfactory. Often, the only tuning option for a user is to refine the input trimap, which may not be good enough. In high-quality studio environments for example, it is necessary that artists can easily refine predictions to a very high level without having to resort to manually touching up the prediction on a pixel-wise level. In our approach, we aim to solve both of these issues by estimating not only the alpha value, but also the foreground and background colors of the input image and allowing more user interaction into the prediction process. We model our network as a

recurrent inference machine (RIM) [94] that sits atop existing matting methods and is trained fully end-to-end with reconstruction and Wasserstein GAN losses [95, 96]. Our contributions are therefore:

- A fully automatic algorithm that can be added to any existing alpha prediction algorithm and that estimates the foreground and background colors for a given alpha, essentially solving the inverse compositing problem.
- Due to the recurrent nature of our algorithm, we allow users to update intermediate predictions to further guide the algorithm, which can lead to much better predictions through only a minor amount of manual work.
- We show through experiments that our color predictions substantially surpass the state-of-the-art. Especially our foreground color prediction leads to faithful new composites.
- Our method is small and lightweight and can process even high resolution images very fast.

While our network predicts both foreground and background colors, as well as the alpha, it is explicitly not an inpainting method aiming to remove the foreground object or reconstruct the background. We strictly focus on foreground color prediction.

4.2 Method

Our method aims to solve the inverse compositing problem by simultaneously estimating the alpha matte, as well as the foreground and background colors of a given image. Instead of trying to solve

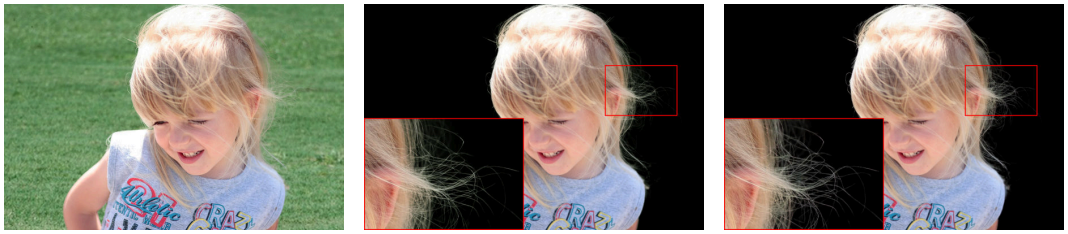

Figure 4.1: Comparison of the ground truth and our predictions. To the left: The foreground extracted from the input image using the ground-truth alpha and composited onto a black background. To the right: The foreground extracted from our predicted foreground using the predicted alpha and composited onto a black background. As can be seen, even when using the ground-truth alpha, the green of the old background shines through in the new composition. This is not the case in our composition.

either of these problems from scratch, we rely on an initial guess for the solution of the alpha and leverage the correlation of the three problems to solve for the foreground and background colors in the process. As a result of this, our method can be seen as a post-processing method that can be used on any of the many methods that aim to predict the alpha matte of an image. Our method estimates the foreground and background colors, slightly refines the alpha, and additionally gives users the ability to easily refine the results further with easy manual user interaction as described in Section 4.3.

To achieve this, we design a recurrent inference machine [94] to solve the inverse of the forward model given in Equation 1.1. Traditionally, one way to achieve this would be to optimize the maximum a posteriori (MAP) solution, given a likelihood and prior as has been done by Bayesian Matting [6] in the past:

$$\arg\max_{F,B,\alpha} P(F,B,\alpha|I) = L(I|F,B,\alpha) + L(F) + L(B) + L(\alpha), \tag{4.1}$$

where $L(\cdot)$ is the *log likelihood* $L(\cdot) = logP(\cdot)$, α is the alpha matte and F, B, I are the foreground, background and observed image colors

respectively.

In contrast, a RIM as proposed by Putzky et al. [94] is a recurrent neural network (RNN) that is able to learn the iterative inference of the problem and implicitly learns the prior and inference procedure in the model parameters. Each state of the RIM includes the current solution, a hidden memory state and the gradient of the likelihood to the problem, which gives information about the generative process and indicates how good the current solution is. Given an observation $\mathbf{y}$ and a previous solution $\mathbf{x}_{t-1}$, the RIM calculates the gradient of the log-likelihood $\nabla L(\mathbf{y}|\mathbf{x}_{t-1})$ as an additional input to the network and predicts an update step $\Delta\mathbf{x}_t$ such that

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta\mathbf{x}_t, \tag{4.2}$$

as can be seen in Figure 4.2. In this paper, the image I is the observation $\mathbf{y}$ and the foreground, background and alpha F, B, α form the current solution $\mathbf{x}_t$.

The log-likelihood in our case is modeled by the difference between the color in the observed image and the color that would result from the composition of the predicted foreground, background, and alpha [6]:

$$L(\mathbf{y}|\mathbf{x}) = L(I|F, B, \alpha) = -\frac{||I - \alpha F - (1-\alpha)B||^2}{\sigma^2}. \tag{4.3}$$

This corresponds to a Gaussian probability distribution centered around $\bar{C} = \alpha F + (1-\alpha)B$ with a standard deviation of σ. From this, the gra-

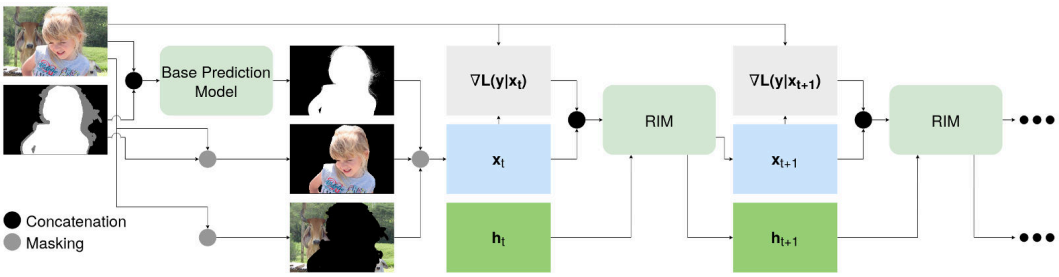

Figure 4.2: The Overall system. First, we do an initial alpha prediction using any prediction model. Usually, these need a trimap as additional input, but this is not required by our approach. Next, we create initial foreground and background predictions, using either the trimap or the initial alpha prediction as a mask for the input image. All three initial predictions get concatenated to form the 7-channel $\mathbf{x}_t$. Given the original image $\mathbf{y}$ and $\mathbf{x}_t$, we calculate the gradient of the log-likelihood $\nabla L(\mathbf{y}|\mathbf{x}_t)$, which we concatenate to $\mathbf{x}_t$ to form the input of the RIM. The RIM then calculates and applies an update step $\Delta\mathbf{x}_{t+1}$ for $\mathbf{x}_t$ to form $\mathbf{x}_{t+1}$. We continue this process for a maximum of 5 iterations in our experiments.

dient of this log-likelihood is given by:

$$\begin{aligned}\nabla L(\mathbf{y}|\mathbf{x}) = \nabla L(I|F,B,\alpha) &= \left[\frac{\partial L}{\partial F}, \frac{\partial L}{\partial B}, \frac{\partial L}{\partial \alpha}\right]^T \\ &= \begin{bmatrix} 2\alpha(I - \alpha F + B - \alpha B) \\ (-2 + 2\alpha)(I - \alpha F + B - \alpha B) \\ \sum_{R,G,B} (2F + 2B)(I - \alpha F + B - \alpha B) \end{bmatrix} * \frac{1}{\sigma^2}.\end{aligned} \tag{4.4}$$

Since $\frac{\partial L}{\partial \alpha}$ is a sum across all three RGB channels, we have to sum over all three channels. As stated previously, our method serves as a post-processing for other alpha prediction methods. As such, we use the output of whichever alpha prediction method we use as a base as the initial guess for the alpha. For the initial foreground and background predictions, we use the original input image in all areas where the trimap gives known foreground/background regions respectively and zeros otherwise. If the base method does not need to use a trimap, we simply use the regions where the predicted alpha is purely foreground or purely background as mask.

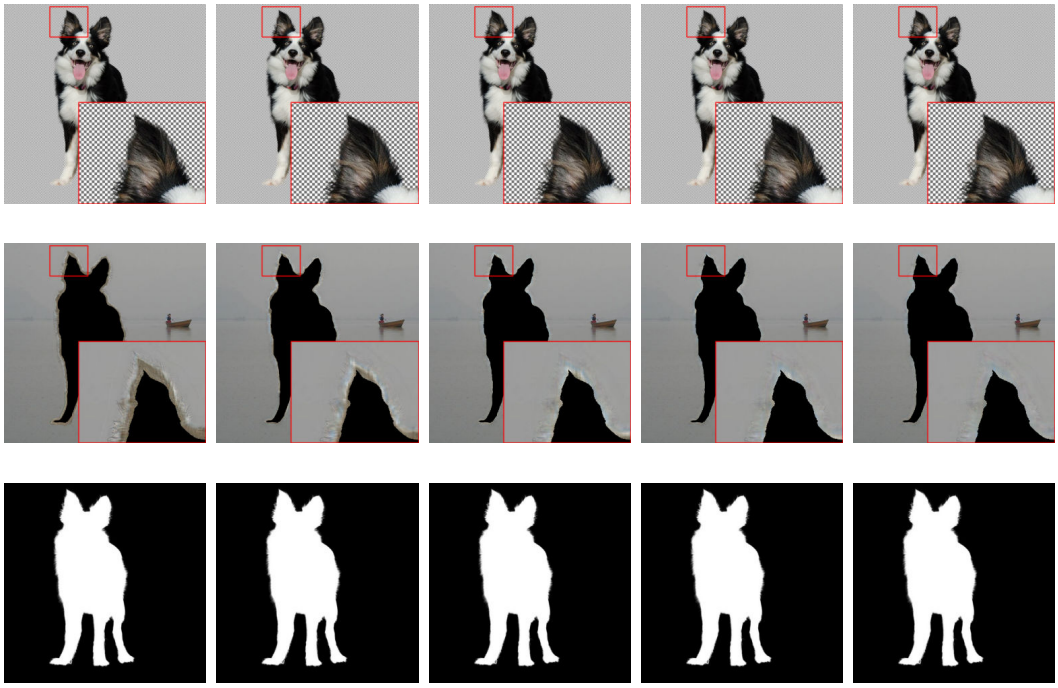

Figure 4.3: Our prediction over $t = 1, \ldots, 5$. While our alpha prediction barely changes from the initial input, both the foreground and background predictions get consecutively more accurate, as seen especially in the background prediction.

The RIM then predicts updates for the current solution at every iteration, that consecutively lead to an ideal solution for all predictions. A loss is calculated for every iteration with the final loss being defined as:

$$\mathcal{L}^{total} = \sum_{t=1}^{T} w_t \mathcal{L}(\mathbf{x}_t, \mathbf{x}_{target}), \tag{4.5}$$

where T is the total number of iterations, w_t is a positive weighting factor and $\mathbf{x}_{target}$ is the ground-truth. In this paper we set $T = 5$ and $w_t = 1$ for all iterations, which we experimentally found to achieve the best results. Further details of our loss function are given in Section 4.2.5. A visualization of the iterative process can be seen in Figure 4.3.

4.2.1 Spectral normalization

Spectral normalization [97] has originally been proposed for use in the discriminator of generative adversarial networks and has been adapted widely [98, 99, 100], especially in image synthesis tasks and even outside of GANs [101, 98, 61]. Spectral normalization controls the Lipschitz constant of a function f by constraining the spectral norm of each layer $g : \mathbf{h}_{in} \rightarrow \mathbf{h}_{out}$. The Lipschitz norm $||g||_{Lip}$ is defined as equal to $\sup_{\mathbf{h}}|||\nabla g(\mathbf{h})|||$, where $|||(A)|||$ is the spectral norm of matrix A, which is the L_2 matrix norm of A.

$$|||(A)||| := \max_{\mathbf{h}:\mathbf{h}\neq 0} \frac{||A\mathbf{h}||_2}{||\mathbf{h}||_2} = \max_{||\mathbf{h}||_2 \leq 1} ||A\mathbf{h}||_2, \tag{4.6}$$

which is equivalent to the largest singular value to A. For a linear layer $g(\mathbf{h}) = W\mathbf{h}$ the norm is given by $||g||_{Lip} = \sup_{\mathbf{h}}|||\nabla g(\mathbf{h})||| = \sup_{\mathbf{h}}|||W||| = |||W|||$. Spectral normalization normalizes the spectral norm of the weight matrix W so that it satisfies the Lipschitz constraint $|||W||| = 1$:

$$\bar{W}_{SN}(W) := W/|||W|||. \tag{4.7}$$

To estimate $|||W|||$, the power iteration method [102, 103] is used, which is computationally very cheap.

4.2.2 Gated Recurrent Units

Gated recurrent units (GRU) have been proposed by Cho et al. [105] as an alternative to long short-term memory units (LSTM) [106] that are simpler to compute and implement. A graphical depiction of the unit can be seen in Figure 4.4. Similar to the LSTM, the GRU consists of several gates through which the input flows. As a recurrent unit, the unit activates through several timesteps. The input

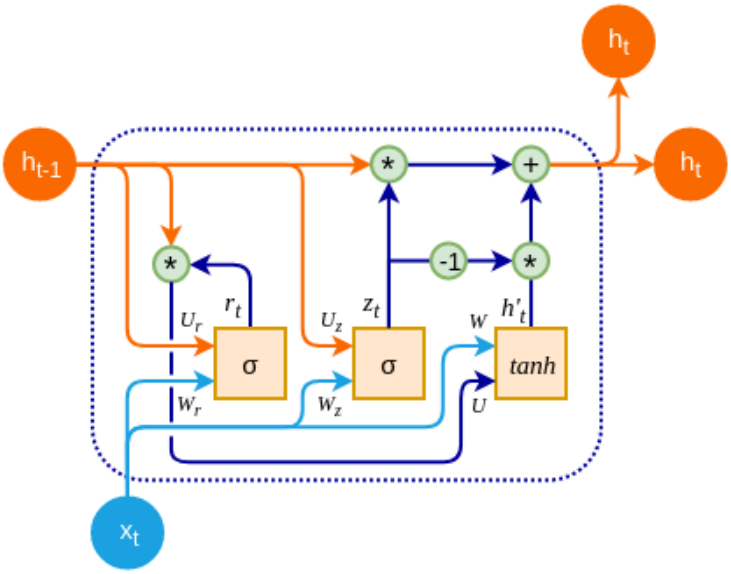

Figure 4.4: Gated Recurrent Unit. Image taken from [104].

at timestep t consists of a feature vector $\mathbf{x}_t$ and the hidden state from the previous iteration $\mathbf{h}_{t-1}$. The initial hidden state for $t = 0$ is defined as $\mathbf{h}_0 = 0$. The *reset gate* $\mathbf{r}_t$ for time step t is calculated with:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}), \tag{4.8}$$

where σ is the logistic sigmoid function and $\mathbf{W}_r$ and $\mathbf{U}_r$ are learned weight matrices respectively. The *update gate* $\mathbf{z}_t$ is similarly computed by:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}). \tag{4.9}$$

The actual activation $\mathbf{h}_t$ is then computed by:

$$\mathbf{h}_t = \mathbf{z}_t \mathbf{h}_{t-1} + (1 - \mathbf{z}_t)\hat{\mathbf{h}}_t, \tag{4.10}$$

where

$$\hat{\mathbf{h}}_t = \phi(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1})), \tag{4.11}$$

and where ϕ is the hyperbolic tangent function and $\odot$ the hadamard product. Through this formulation, previous hidden states can be

forgotten if the reset gate is close to 0 and the new hidden state is forced to reset with the current input only, which allows the unit to drop unnecessary information. The update gate on the other hand, decides how much information from the previous hidden state carries over to the new one, which allows the unit to remember long-term information. Since each of these units have independent reset and update gates, they can capture information at different time scales. Although GRUs have been developed for machine translation and worked on input *vectors*, it is possible to transform these units to convolutional GRUs by replacing the matrix multiplications with convolutions.

4.2.3 Network architecture

We base our network architecture on the network proposed by Putzky et al. [94]. The full input of the network consists of the current solution $\mathbf{x}_t = [F, B, \alpha]^T$ and the gradient of the log-likelihood $\nabla L(I|F, B, \alpha)$, resulting in a full resolution feature map of 14 channels in total. The first convolution layer of the network downsamples the input by a factor of 2 and is followed by a GRU. Following that, a transposed convolution layer upsamples the resulting feature maps back to the original resolution, again followed by a GRU. A final convolution layer serves as output layer and reduces the number of channels back to 7. With the exception of the last one, all convolution layers use spectral normalization [97] and a tanh nonlinearity. The number of feature maps for the inner two convolutions is 32 respectively and the number of feature maps for the hidden states of the GRUs is 128. All convolution kernels are 3×3. We choose spectral normalization instead of the popular batch normalization due to the formers success in generative adversarial networks and the latters

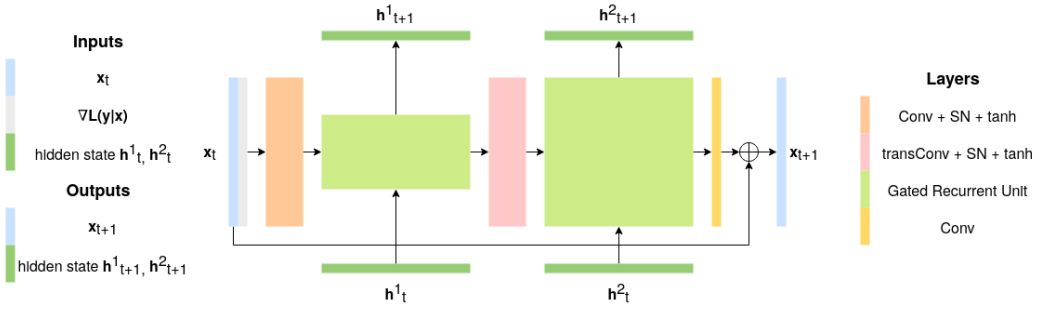

Figure 4.5: Network architecture. For each iteration, a candidate solution $\mathbf{x}_t$ serves as input, is concatenated with the gradient $\nabla L(\mathbf{y}|\mathbf{x})$ and put through the network to predict an update step, which is added to $\mathbf{x}_t$. Convolutions are normalized through spectral normalization (SN) and use tanh as activation function.

problems when applied to recurrent networks. These problems are due to the nature of batch normalization: It tries to normalize the input of each layer and learns the characteristics of the input during training. In a recurrent network, however, the same layer is visited multiple times and the input characteristics can be different in each iteration, which can make the batch normalization layer difficult or impossible to train. Spectral normalization, on the other hand, normalizes the weights of the layer, side-stepping this issue entirely. The full network structure can be seen in Figure 4.5.

Our network only contains 1155680 parameters, which easily fits even on mobile devices. However, the network operates on full resolution feature maps and propagates full resolution hidden states to the next iteration, which may use a lot of memory. This is not an issue, however, since the receptive field of the network is only 11×11. Due to this, even very high resolution images can be processed in smaller tiles with an overlap between tiles of only 11 pixels.

4.2.4 Spatial tiling

If a high resolution image is too large to wholly fit into GPU memory, it is possible to process the image on the CPU instead. However, this can be quite slow. Another possibility is to tile the high resolution image into smaller sub-images and process them one at a time. In this approach, the receptive field of the network is important. This can be visualized in Figure 4.6. In a) the image is naively tiled into 4 sub-images that directly border on each other. The receptive field of size 5×5 of an example network is shown around the red pixel in the upper left tile. Due to the naive tiling, the receptive field would not see the actual input pixels in the 5×2 marked region to the right of the pixel. Instead this area would be filled through some sort of padding. As the sliding window of the convolution moves across the image, a 2 pixels wide border on each sub-image would generate values that do not match the values that would have been generated if the image was processed in full, due to the difference of actual image pixels and padded input pixels. This pattern of border artefacts can be seen in 4.6 b). To avoid this, the high resolution image needs to be tiled in such a way, that there exists an overlap between tiles the size of the receptive field of the image as seen in 4.6 c). Due to this, the pixels in the border region can be safely discarded for each sub-image, since each pixel will exist in a non-border region of one sub-image. Since the amount of overlap between sub-images corresponds to the size of the receptive field, larger receptive fields require more sub-images to process, which in turn slows down the speed of the method. Since our method has a receptive field of only 11×11, the method needs barely any extra tiles over the naive tiling method to faithfully process the whole image. Pseudocode of the tiling is presented in algorithm 1.

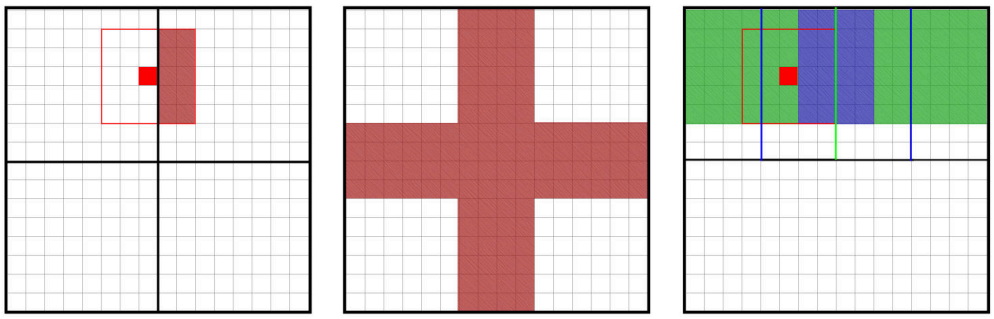

Figure 4.6: a) Image naively tiled into 4 sub-images. The 5×5 receptive field around the red pixel is shown by the red lines. The 5×2 shaded region in the receptive field is getting padded due to the tiling and therefore results in a wrong output of the red pixel. b) The pattern of wrong outputs in this example. c) The first row of the correct tiling pattern, with 3 tiles per row instead of 2 and a $\lfloor \text{RF}/2 \rfloor * 2 = 4$ pixel overlap between tiles. The green and blue shaded regions designate the area kept per tile respectively. As can be seen around the red pixel, all pixels within the receptive field properly exist in the tile. Since the outer $\lfloor \text{RF}/2 \rfloor = 2$ pixels from the border in each tile are discarded, no wrong results due to padding can appear.

4.2.5 Loss Function

Inspired by the success of generative inpainting networks [56, 107], we use the WGAN-GP loss [95, 96], combined with a L_1-based reconstruction loss to train our network.

Following previous methods [8, 55, 61, 54, 57], we define the reconstruction loss only over the unknown region of the image. This leads to the reconstruction loss at iteration t:

$$\mathcal{L}_1 = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} |\mathbf{x}_{i,t} - \mathbf{x}_{i,target}|, \tag{4.12}$$

where $\mathcal{U}$ is the unknown region of the trimap. Note that we do not calculate the loss independently for the alpha, foreground and background predictions. $\mathbf{x}_{target}$ is defined as the ground-truth foreground, background and alpha concatenated to a 7-channel fea-

Algorithm 1 Tiling a high-resolution image.

function TILING($\mathbf{y}, \mathbf{x}, \mathbf{h}$)
$\mathbf{y}$ - observed image, $\mathbf{x}$ - candidate solution, $\mathbf{h}$ - hidden state
RF - receptive field size, TS - tile size, Net - prediction network
  if $height(\mathbf{x}) < TS$ **and** $width(\mathbf{x}) < TS$ **then**
    return $Net(\mathbf{y}, \mathbf{x}, \mathbf{h})$
  else
    $VT = \left\lceil \frac{height(\mathbf{x})}{TS-RF+1} \right\rceil$ ▷ Number of vertical tiles
    $HT = \left\lceil \frac{width(\mathbf{x})}{TS-RF+1} \right\rceil$ ▷ Number of horizontal tiles
    $TP = LP = \lfloor \mathrm{RF}/2 \rfloor$ ▷ Top padding and left padding
    $BP = VT * (TS - RF + 1) + TP - height(\mathbf{x})$ ▷ Bottom padding
    $RP = HT * (TS - RF + 1) + LP - width(\mathbf{x})$ ▷ Right padding
    $\mathbf{x} = \mathbf{x}_{out} = Pad(\mathbf{x}, LP, RP, TP, BP)$ ▷ Pad $\mathbf{x}$
    $\mathbf{y} = Pad(\mathbf{y}, LP, RP, TP, BP)$ ▷ Pad $\mathbf{y}$
    $\mathbf{h} = \mathbf{h}_{out} = Pad(\mathbf{h}, LP, RP, TP, BP)$ ▷ Pad $\mathbf{h}$
    for every $tile$ in $VT \times HT$ **do**
      $\mathbf{x}_{tile} = Extract(\mathbf{x}, tile)$ ▷ Extract tile from $\mathbf{x}$
      $\mathbf{y}_{tile} = Extract(\mathbf{y}, tile)$ ▷ Extract tile from $\mathbf{y}$
      $\mathbf{h}_{tile} = Extract(\mathbf{h}, tile)$ ▷ Extract tile from $\mathbf{h}$
      $\mathbf{x}_{tile}, \mathbf{h}_{tile} = Net(\mathbf{y}_{tile}, \mathbf{x}_{tile}, \mathbf{h}_{tile})$
      $\mathbf{x}_{out} = Insert(\mathbf{x}_{out}, \mathbf{x}_{tile})$ ▷ Insert $\mathbf{x}_{tile}$ into $\mathbf{x}_{out}$
      $\mathbf{h}_{out} = Insert(\mathbf{h}_{out}, \mathbf{h}_{tile})$ ▷ Insert $\mathbf{h}_{tile}$ into $\mathbf{h}_{out}$
    end for
    $\mathbf{x}_{out} = Depad(\mathbf{x}_{out})$ ▷ Remove padding
    $\mathbf{h}_{out} = Depad(\mathbf{h}_{out})$ ▷ Remove padding
    return $\mathbf{x}_{out}, \mathbf{h}_{out}$
  end if
end function

ture map and $\mathbf{x}_t$ as the update step predicted by the RIM at iteration t added to the previous prediction: $\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta\mathbf{x}_t$.

We further add the WGAN-GP loss to train the RIM with adversarial gradients and improve prediction accuracy. In generative adversarial networks (GANs), a generator G is trained to output generated data $\mathbb{P}_g$ similar to the real data distribution $\mathbb{P}_r$, whereas a competing discriminator (or *critic* in the case of WGAN) aims to distinguish data sampled from real and generated data distributions. WGAN uses the *Wasserstein-1* distance $\mathcal{W}(\mathbb{P}_r, \mathbb{P}_g)$ to compare real and generated data distributions. The WGAN loss is constructed using the

Kantorovich-Rubinstein duality:

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})] - \mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})], \tag{4.13}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions, $\mathbb{P}_g$ is the model distribution generated by the RIM and D is the critic network. To enforce the 1-Lipschitz constraint, Gulrajani et al. [96] proposed a gradient penalty term, since a 1-Lipschitz function only has gradients with a norm of at most 1 everywhere. This gradient penalty term

$$\lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(||\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})||_2 - 1)^2] \tag{4.14}$$

is added to the WGAN loss for random samples $\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}$. In our method, we are only interested in predictions of the unknown region $\mathcal{U}$. Consequently the gradient penalty should only be applied to pixels inside that region. Therefore we follow Yu et al. [56] by multiplying the gradients with a mask $\mathbf{m}$ of the unknown region. From this, the final WGAN-GP loss at iteration t follows as:

$$\mathcal{L}_{WGAN} = \mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(||\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}}) \odot \mathbf{m}||_2 - 1)^2]. \tag{4.15}$$

The balancing term λ is set to 10 in our approach. As critic we use SN-PatchGAN which has provided fast and stable training in inpainting tasks [107]. Instead of using the whole 7-channel feature map $\mathbf{x}$ as input to the critic, we composite a new image out of the foreground, alpha and a new randomly selected background, similar to Lutz et al. [8]. We calculate $\mathcal{L}_1$ and $\mathcal{L}_{WGAN}$ at every iteration t and add them. Finally, we sum them up for the final loss term in Equation 4.5.

4.2.6 Training details

We train our network on the publicly available matting dataset published by Xu et al. [54]. This dataset contains 431 unique foreground images and corresponding alpha mattes. They further released another 50 unique images that can be composited with predefined backgrounds to create the Composition-1k testing dataset. To generate the initial alpha predictions during training, we use GCA-Matting [61] as a base since they achieve the best performance on the Composition-1k testing set. To produce the initial foreground and background predictions, we use the given trimap to mask the input image in their respective known regions. Afterwards, all images are normalized to the range of $[-1, 1]$. Since the training dataset is small, data augmentation is essential to properly train a neural network and we follow the improved data augmentation strategy of Li et al. [61]. This strategy initially selects two random foreground images with a probability of 0.5 and combines them to create a new foreground object with corresponding alpha. They follow this by resizing the image to 640×640 with a probability of 0.25 to generate some images that contain nearly the whole image instead of random patches. Following this, a random affine transformation is applied to the image, which consists of random rotation, scaling, shearing and flipping. Afterwards, a trimap is generated by random dilation and a 512×512 patch is cropped from the image, centered around a random pixel in the unknown region of the trimap. Then, the image is converted to HSV space and random jitter is applied to hue, saturation and value. Finally, a random background image is selected from MS COCO [78] and the input image is composited. Naturally, all applicable transformations are applied to the foreground image, alpha and trimap to keep them matching.

We use the Adam optimizer [108] with a fixed learning rate of 10^{-4} and train for 100000 iterations.

4.3 Manual editing

One of the main objectives of our method is to give users the optional ability to more directly influence the prediction process. Due to the recurrent nature of our method, this is easily achievable. During any step of the prediction, users can directly manually update any one of the foreground, background or alpha predictions through image editing software. Since all three predictions are fundamentally linked, changes in one of them can propagate to the other two. For example, a user may manually inpaint part of the background that the network struggles with and which may be easier to adjust for the user than directly changing either the foreground colors or the alpha. This change propagates to the foreground prediction and can lead to a much better foreground in the end. A detailed example of this can be seen in Figure 4.7. To make sure that the network recognizes any direct modification of the predictions by the user, we additionally set the hidden states at the corresponding locations to 0. The whole process is intuitive and can easily be implemented as part of any image processing software.

4.4 Results

We evaluate our proposed method on the Composition-1k dataset [54], with regards to the alpha matte prediction, as well as the foreground and background color prediction. The Composition-1k dataset

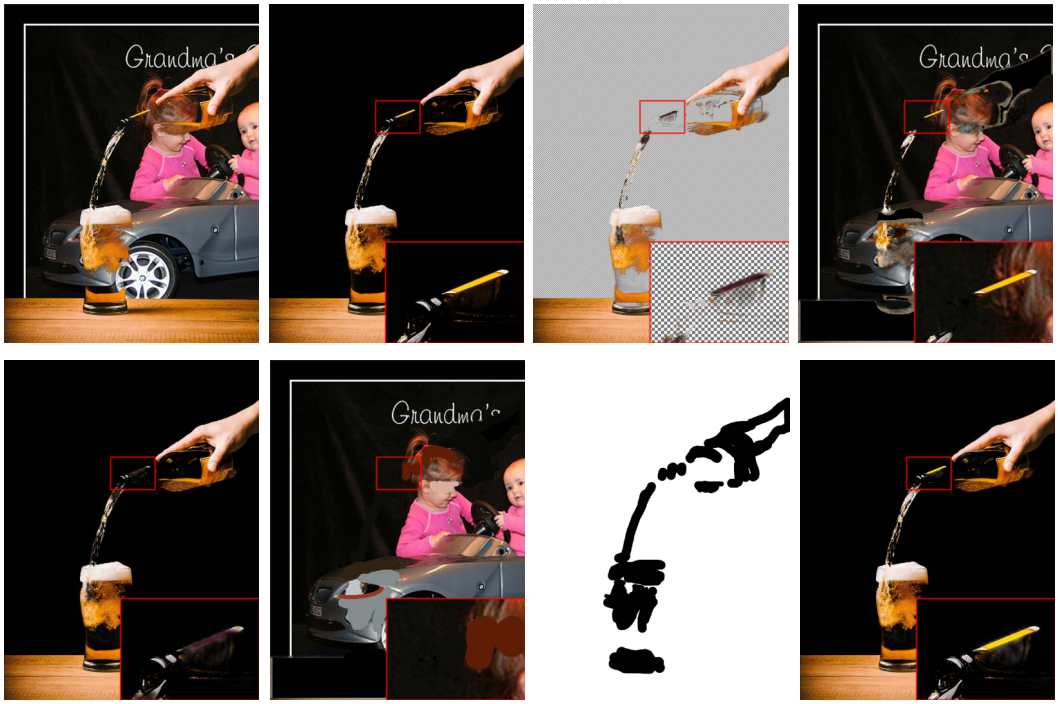

Figure 4.7: Visualization of the manual editing process. Top row from left to right: Input image, ground-truth composition, predicted foreground, predicted background. Bottom row from left to right: Resulting composition, very rough edit to the background, corresponding editing mask and the composition resulting from manual edit. As can be seen, the color predictions in this example are bad and incorrectly color parts of the background with foreground colors and vice versa. After the rough manual edit to the background, the algorithm recovers the correct foreground color.

consists of 50 unique foreground images that have been composited into new images using 20 predefined backgrounds each, resulting in 1000 testing images in total. To generate our results, we use several different published methods for alpha prediction and use them as initial guess for our method. Using these predictions, we then generate foreground colors, background colors and alphas over 5 iterations in total, which we experimentally found to lead to good convergence while keeping computation time low.

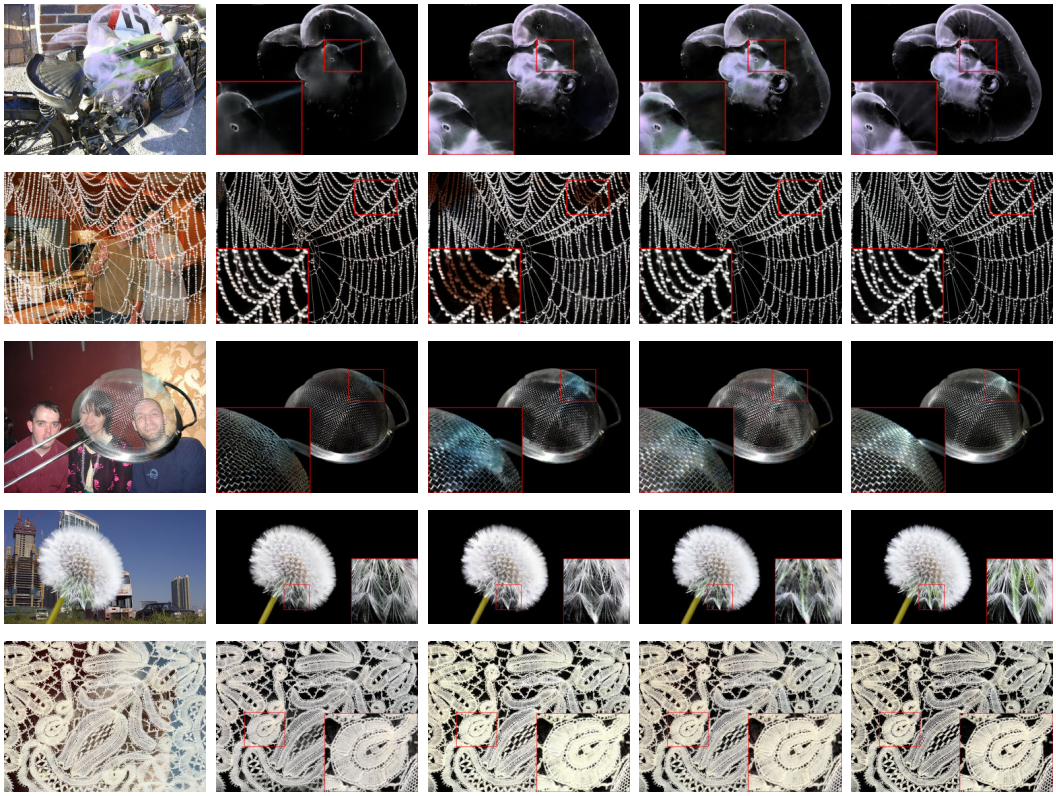

Figure 4.8: Visual comparison on the Composition-1k dataset. From left to right: Input image, Compositions from Context-Aware [60], Samplenet [55], Ours, Ground-truth. As can be seen in the highlighted regions, Context-Aware predicts comparatively plain colors. Samplenet predicts good foreground colors, but sometimes introduces patches of wrong color to the foreground. Our method leads to compositions closest to the ground-truth overall.

4.4.1 Foreground and background color prediction

Due to the composited nature of the Composition-1k dataset, we have the ground-truth background and foreground colors available. Therefore, we can calculate metrics for prediction on this dataset. As metrics, we choose the MSE and SAD of $\alpha \times F$ and $(1-\alpha) \times B$ as introduced in [109]. However, due to the nature of the dataset, which can contain wrong foreground colors in all pixels where the alpha is 0, we multiply by the ground-truth alpha. This also disentangles the performance of all evaluated methods from their performance in predicting the alpha. We further only consider pixels in the unknown trimap area.

We compare our results against Closed-form Matting [47], Context-Aware Matting [60] and Samplenet [55]. Closed-form Matting can predict the alpha matte by solving a sparse linear system of equations. As a second step and given an alpha as input, it is possible to solve this equation for the foreground and background colors instead of the alpha. For fair comparisons, we use the superior alpha predictions from GCA-Matting [61] as input. Context-Aware Matting predicts the foreground and alpha simultaneously through a deep neural network. By contrast, Samplenet does a full background, foreground and alpha prediction sequentially using 3 networks. Additionally, we can use the input image as a baseline comparison. We use the same initial alpha predictions for our method as in the evaluation of the alpha prediction, as are shown in Table 4.1.

As can be seen in the table, we achieve the overall best results for the color prediction by a wide margin, especially in the prediction of the foreground colors. The background predictions of Samplenet are marginally better than ours according to the SAD, but it performs worse in foreground prediction. We can also observe that the quality of the initial alpha prediction has a large impact on the resulting quality of the foreground and background predictions. The better the initial alpha prediction is, the better our method performs. However, even alpha predictions from the outdated KNN-Matting lead to foreground colors that are almost as good as the colors naively taken from the input image and noticeable better background colors. We also outperform the Closed-form solution for the foreground and background colors, even when given the GCA-Matting alpha prediction as input. Furthermore, the Closed-form solution is quite slow in comparison to modern deep learning methods, which is non-optimal for any interactive application. Vi-

	Foreground		Background	
Methods	SAD	MSE (10^4)	SAD	MSE (10^4)
Input Image	58.32	26.49	57.90	26.12
KNN [92] + Ours	59.91	49.25	36.77	15.19
AlphaGAN [8] + Ours	44.27	27.90	40.12	16.93
IF [50] + Ours	37.93	31.98	29.49	15.82
GCA [61] + CF [47]	31.98	23.15	29.40	10.69
Context-Aware [60]	46.93	18.02	-	-
SampleNet [55]	42.68	29.26	**24.59**	7.99
GCA [61] + Ours	**28.32**	**12.10**	25.07	**5.97**

Table 4.1: Quantitative results of the foreground and background prediction on the Composition-1k dataset. Best results are emphasized in bold. Note that not all images could be predicted for KNN Matting and Information-flow Matting due to trimaps incompatible with these methods.

sual comparisons can be seen in Figure 4.8 and in the appendix B.2.1.

4.4.2 Color prediction over several iterations

Our method is a recurrent neural network and predicts update steps to the previous solution over several iterations to output new solutions. We compare our results for the color predictions in Table 4.2. As can be seen, our predictions get consecutively better over iterations until they saturate after $t = 5$.

4.4.3 Manual editing

To show the impact of the manual editing process, we compare the fully automatic output of our method to results we get when making only small edits during the process. For this, we take 5 of the images of the Composition-1k dataset where the automatic prediction generates sub-par results and spend less than a minute each on manually improving the intermediate alpha predictions. As can be seen in Table 4.3, even small edits focusing on the foreground

massively improve the color prediction. Naturally, these edits could be done on the alpha predictions of other methods as well, however, only our method interlinks the alpha and color predictions in a way to propagate the changes from the alpha to the foreground and background color predictions automatically. In other methods the edits to the alpha would have to be replicated for the color predictions as well, increasing the amount of work. Please refer to the appendix B.2.2 for the images and edits.

4.4.4 Alpha matte prediction

For the alpha matte prediction, we compare on the commonly used evaluation metrics [53]. The full results of the evaluation are shown in Table 4.5. As can be seen from the table, the alpha predictions of our method barely differ from those of the initial alpha prediction. We slightly improve on the input alpha prediction, but are unable to significantly refine them even if the initial alpha predictions are bad. Some visual results of our alpha predictions using GCA Matting [61] as input can be seen in Figure 4.8.

4.4.5 User study

To further evaluate the quality of our work, we conduct a user study comparing the foreground color prediction of our method with Samplenet [55] and Context-Aware Matting [60]. Following the approach of [60], we take all 31 images of the real-world image dataset from Xu et al. [54] and use the predicted alpha and foreground colors to composite a new image with a plain background. To ensure any differences between matting results are only due to the color predictions, we use the predicted alpha from Samplenet as initial guess

for our method in the comparisons with Samplenet and similar for our comparisons with Context-Aware Matting.

We recruited 20 participants for our user study for each of the comparisons. Each participant was presented with a short training session where the results of two methods was shown and the differences explained. This was done to help people with no prior matting experience spot the subtle differences in results.

Each participant conducted 31 trials corresponding to all the images of the real world dataset. In each trial the original image was shown at the top with the composited results of the methods at the bottom. The results were shown one at a time and the participant could use buttons or the arrow keys to switch the bottom image between the result images. By being able to rapidly switch between images, it is possible to recognize even subtle differences in the foreground colors. The participants were asked to choose the result which they found more accurate and realistic.

We calculated the preference rate of the participants of our results and show the mean preference rate and standard deviation in Table 4.4. As can be seen, the majority of participants preferred our results to those of Samplenet. However, when comparing to Context-Aware Matting, the results show no preference of one over the other. Some examples of our study can be seen in the appendix B.2.3. As can be seen, the color differences between our results and Context-Aware Matting are very minor, which explains the responses. However, our method still significantly outperforms Context-Aware Matting numerically in the Composition-1k dataset and offers the option to further improve the results interactively.

	Foreground		Background	
Iteration	SAD	MSE (10^4)	SAD	MSE (10^4)
1	42.06	16.34	35.17	10.67
2	31.18	12.87	27.57	7.20
3	29.18	12.24	25.83	6.33
4	28.41	**12.05**	25.24	6.05
5	**28.32**	12.10	**25.07**	**5.97**

Table 4.2: Our results for foreground and background color prediction over iterations using the GCA alpha prediction as initial input. Best results are emphasized in bold. We do not see improvements after 5 iterations.

	Foreground		Background	
	SAD	MSE (10^4)	SAD	MSE (10^4)
Pre-edit	161.50	143.26	60.39	17.25
Post-edit	81.32	22.748	60.21	17.15

Table 4.3: Our results for foreground and background color prediction for selected examples before and after manual editing.

4.4.6 Limitations

The goal of this work is to introduce a lightweight method that can be used with any alpha prediction network to estimate the foreground and background colors that lead to compelling new composites. However, we do not refine the input alpha to a significant amount due to the small capacity of our network. In future work, it may be desirable to explore an updated network architecture that is able to further refine inadequate initial alpha predictions.

Further, as opposed to Samplenet, our method does not predict good background colors in areas where the background can not, or only barely, be seen in the image. This has no impact on new compositions and we do not claim to fully inpaint the background after the foreground has been removed. However, certain applications may find a fully inpainted background desirable, which we can not provide.

Ours vs	Mean preference rate	Std
Context-Aware [60]	48.87%	0.15
Samplenet [55]	64.84%	0.19

Table 4.4: Results of the user study on the real world dataset [54]. Given the pairwise choice between foreground objects extracted using our method and the state of the art, our results were about as often chosen as Context-Aware Matting [60] and significantly more often than Samplenet Matting [55].

Methods	MSE	SAD	Grad	Conn
KNN [92]	0.078	112.60	67.68	113.47
KNN [92] + Ours	0.078	112.81	67.75	113.52
IF [50]	0.066	75.41	63.39	75.48
IF [50] + Ours	0.066	75.47	63.38	75.48
AlphaGAN [8]	0.031	68.71	50.97	70.42
AlphaGAN [8] + Ours	0.031	68.72	50.97	70.40
Deep Image Matting [54]	0.014	50.4	31.0	50.8
IndexNet [59]	0.013	45.8	25.9	43.7
VDRN [110]	0.011	45.3	30.0	45.6
AdaMatting [57]	0.010	41.7	16.8	–
SampleNet [55]	0.010	46.79	22.50	45.64
SampleNet [55] + Ours	0.010	46.82	22.51	45.66
GCA [61]	0.009	**35.28**	16.92	32.53
GCA [61] + Ours	**0.009**	35.31	**16.91**	**32.53**

Table 4.5: Quantitative results of the alpha prediction on the Composition-1k dataset. Best results are emphasized in bold. Note that not all image could be predicted for KNN Matting and Information-flow Matting due to trimaps incompatible with these methods.

4.5 Conclusion

In this chapter, we propose a novel method to estimate foreground and background colors given an initial alpha prediction. Our method is lightweight and can easily be used on top of any other alpha prediction method. We show that even initial alpha predictions that do not satisfy high-quality standards generate color predictions that are quantitatively better than the colors directly taken from the input image. We show through quantitative and qualitative evaluation that our method substantially outperforms the state-of-the-

art in foreground and background color estimation. Further, the recurrent nature of our method allows users to manually edit parts of the candidate solutions with ease, which can propagate further and lead to better final predictions. We show that very rough edits to the background candidate solution can lead to a significantly better final foreground solution through minimal effort.

Chapter 5

Video matting

In this chapter the third contribution of this thesis is presented. First, an introduction motivating the project is given. Afterwards our method is explained in detail and some concepts related to this chapter are introduced. Following this, evaluations on the video matting task are done. Finally, a conclusion is given, discussing the limitations of current methods.

5.1 Motivation

In recent years, natural image matting algorithms have reached an astonishing performance, mainly due to leveraging neural networks. This was possible due to the release of a publicly available matting dataset that while small, was of sufficient size to train neural networks for the natural image matting task in a supervised manner. In fact, the current state-of-the-art in natural image matting outperforms the classical approaches to such a degree, that they reach a higher performance than classical video matting methods on videos, even though they are trained only on single images. However, this does not mean that they perform flawlessly. Temporal inconsistencies such as flickering are naturally prevalent in approaches that were never designed to handle such issues. However, properly labeled video sequences for video matting are scarce and seem insufficient to properly train networks for this task. Nevertheless, video matting is an important problem and in this chapter we investigate ways to add temporal consistency to natural image matting methods. In addition to evaluations on existing methods to apply temporal consistency, we design a neural network for the video matting task and train it in a way that leverages the single image matting performance of modern algorithms while also introducing temporal consistency to reduce flickering. While our method does not achieve higher performance than the state-of-the-art on the videomatting.com benchmark, it does show much promise in reducing temporal artifacts as we will show through visual examples in Section 5.3.

5.2 Method

Our method uses 3D convolutions to enforce temporal consistency across frames in a video sequence. Due to the scarcity of available video matting data, we designed our approach to enhance existing natural image matting algorithms instead of designing a new network from the ground up.

5.2.1 Data augmentation

To train our network, we use the publicly available video matting dataset from videomatting.com [80]. As has been described in Section 2.5.1, the training set contains 3 sequences, containing 150, 285 and 150 frames respectively with both ground-truth alphas and foreground colors available. To train our network, we make use of heavy data augmentation. Following the augmentation strategies of previous natural image matting papers [55, 61], each frame in a training sequence is augmented as follows: First, a random affine transformation consisting of random rotation, scaling, shearing and flipping is applied. Afterwards, a trimap is generated by random dilation and a 256×256 patch is cropped from the image, centered around a random pixel in the unknown region of the trimap. Then, the image is converted to HSV space and random jitter is applied to hue, saturation and value. Naturally all random parameters are consistent for each frame in a training sequence to generate coherent training inputs. Finally each frame is composited onto a new background to generate unique sequences. With a probability of 0.5, a random image from MS COCO [78] is selected to serve as background for the whole sequence. Otherwise, consecutive frames from a random video of the DAVIS dataset [111] are selected to simu-

late training sequences with moving background and foreground. During training we chose a sequence length of 10 to give the network enough temporal context to learn.

5.2.2 Channel-separated convolutions

In traditional convolutions all input channels are connected to all output channels. This allows for the most amount of channel interactions within the convolutions, but also the most amount of parameters and floating-point operations. To reduce the number of parameters and operations, it is possible to group the convolution filters into subsets, where each filter is only connected to the channels in its group. An extreme version of this are depthwise convolutions, where each channel is in its own group. This version has the least amount of parameters and operations, but also no channel interactions (See Figure 5.1 respectively). Networks such as Xception [33] and Mobilenet [82] are some of the earliest networks that have used depthwise convolutions in the past. Since depthwise convolutions have no channel interaction at all, 1×1 pointwise convolutions are often used in addition directly afterwards to reintroduce interaction. This combination is called depthwise-separable convolutions. In 3D convolutions, the convolution kernel encompasses the spatiotemporal dimensions and the same concept of grouping the convolution filters applies here. Instead of calling convolutions with maximal groups depthwise-separated convolutions, however, Tran et al. [112] introduce the term *channel-separated* to avoid confusion, since the term *depth* could apply to both the channel and temporal dimensions.

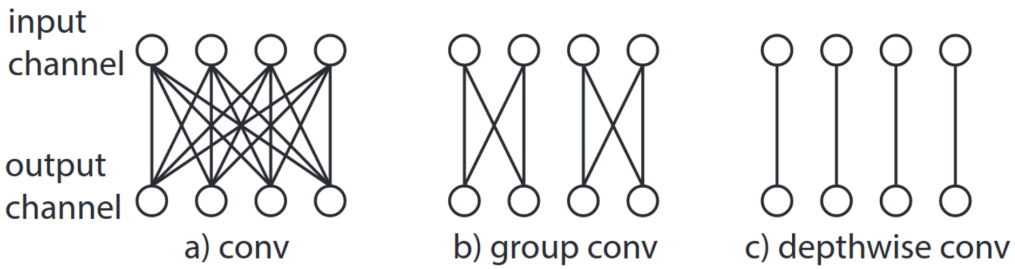

Figure 5.1: Convolutions can be grouped, which each group having only access to their corresponding channels. a) Traditional convolution with one group. All input channels are connected to all output channels. b) Group convolution with 2 groups. c) Depthwise convolution with each channel in its own group. Image taken from [112].

5.2.3 Network architecture

The baseline for our method is GCA-Matting [61]. GCA-Matting is a relatively small network that has reached very good performance for natural image matting. It is an encoder-decoder network with skip-connections and additional guided contextual attention modules. However, as with other natural image matting methods, video matting is not directly considered and the method therefore suffers from temporal inconsistencies. To alleviate this problem, we enhance the network by adding 3D convolutions between the encoder and decoder of the network. These 3D convolutions operate over several frames of the input sequence simultaneously and can therefore learn to spot and remove temporal inconsistencies. We construct our network in 3 variations:

- **v1**: This baseline version adds 2 3D convolutions after the final stage of the encoder. Both convolutions are followed by 3D batch normalization [34] and ReLU activation functions respectively. Both convolutions have a kernel size of $3 \times 3 \times 3$ and add a total of 14158848 parameters to the network. This is a massive amount, especially considering that the baseline GCA-Matting network only has 25269144 parameters in total.

- **v2**: To reduce the number of parameters added from the 3D convolutions and enhance the performance, we update the network from **v1** in two ways. First, we change the sequence of two 3D convolutions into a residual block as in [113]. Second, we change the default 3D convolutions to channel-separated convolutions [112]. This version only adds 29696 new parameters to the network.

- **v3**: In the final version of our network, we add channel-separated convolutions to all connections between the encoder and the decoder of GCA-Matting. In each case, the 3D convolutions are contained within a residual block, as in **v2**. This version adds 59392 new parameters to the network.

Since the aim is to leverage the performance of GCA-Matting on single images, we do not change any details of either the encoder or decoder. This means we can use the weights of the network trained on the dataset released by Xu et al. [54] without any modifications. In all three variants of our network, the receptive field over the temporal dimension is 5. Therefore, the network needs to process at least 5 frames of a video sequence at a time during evaluation.

5.2.4 Training pipeline

In our networks, the only parts containing 3D convolutions can be found in the connections between the encoder and the decoder. The desired input shape of tensors for 3D convolutions is $[B, C, T, H, W]$, for the batch, channel, temporal, height and width dimensions respectively. Since we kept the encoder and decoder of the network deliberately unchanged from when it was trained on single images, these parts of the network only accept inputs in the shape of $[B, C, H, W]$.

Naturally, this means we need to reshape the input tensors appropriately between parts of the network. For the encoder and decoder, all frames are processed individually and the temporal dimension is shuffled into the batch dimension. For the temporal part of the network, we shuffle the temporal dimension back out of the batch dimension.

During training, we use a combination of two loss functions. The first is the reconstruction loss over the unknown region of the video sequence and is defined as:

$$\mathcal{L}_1 = \frac{1}{T}\sum_{t}^{T}\frac{1}{|\mathcal{U}_t|}\sum_{i\in\mathcal{U}_t}|\alpha_{i,t} - \hat{\alpha}_{i,t}|, \tag{5.1}$$

where T denotes the number of frames in the sequence, $\mathcal{U}_t$ is the unknown region of the trimap for frame t and $\alpha_{i,t}$ and $\hat{\alpha}_{i,t}$ are the predicted and ground-truth alpha values at pixel i and frame t respectively.

The second loss function is the dtSSD loss used by Shahrian et al. [114] and defined as:

$$\mathcal{L}_{dtSSD} = \frac{1}{T}\sum_{t}^{T}\sqrt{\sum_{i}(\frac{\mathrm{d}\alpha_{i,t}}{\mathrm{d}t} - \frac{\mathrm{d}\hat{\alpha}_{i,t}}{\mathrm{d}t})^2}. \tag{5.2}$$

We combine these losses using a weighting factor w, such that the final loss in our network is defined as:

$$\mathcal{L} = \mathcal{L}_1 + w * \mathcal{L}_{dtSSD}. \tag{5.3}$$

In our experiments we used $w = 100$ to balance both terms.

We use the weights of a GCA-Matting network trained on single images to initialize the weights of the encoder and decoder respectively and train the network for 50000 iterations on the available video

matting data using the data augmentation described in 5.2.1. We use the Adam optimizer [108] with default parameters and an initial learning rate of 4×10^{-4}, which is reduced during training by cosine decay [115].

During evaluation, at least 5 frames of the input sequence need to be processed at once due to the receptive field of the 3D convolutions. Unfortunately, for larger resolution video sequences, these might not fit onto GPU memory. For our network, however, it is possible to process every frame individually in the encoder, then process 5 at a time in the temporal part and finally process them individually again in the decoder, therefore reducing the amount of needed space on the GPU by storing parts of the output briefly in main memory and sacrificing a bit of speed.

5.3 Results

In this section, we evaluate the state-of-the-art on the video matting task. Since methods designed for natural image matting of individual images are currently outperforming previous video matting methods without any additional processing, we compare different methods to add temporal consistency as a post-processing step to these methods, as well as a baseline method and our method proposed in Section 5.2.

5.3.1 Comparisons

The methods we compare against are as follows:

Baseline: As baseline comparison, we use GCA-Matting [61] without any additional processing.

OFD: Ke et al. [3] proposed a one frame delay processing step that

can be used for any matting algorithm. They recognize that temporal inconsistency in matting algorithms, usually appears in the form of flickering. To alleviate this, they use the preceding and following frames in a sequence to remove inconsistencies in the current frame. A pixel is designated as flickering if it meets the two following conditions $\mathcal{C}$:

1. $|\alpha_{i,t-1} - \alpha_{i,t+1}| \leq \epsilon$
2. $|\alpha_{i,t} - \alpha_{i,t-1}| > \epsilon$ and $|\alpha_{i,t} - \alpha_{i,t+1}| > \epsilon$

These conditions state that if a pixel has close alpha values in $t-1$ and $t+1$, but is very different between t and $t-1$ and t and $t+1$ a flicker appears. In this case the value of $\alpha_{i,t}$ is replaced:

$$\alpha_{i,t} = \begin{cases} (\alpha_{i,t-1} + \alpha_{i,t+1})/2 & \text{if } \mathcal{C}, \\ \alpha_{i,t} & \text{otherwise.} \end{cases} \tag{5.4}$$

In this comparison, we use the OFD step on each of the outputs of the baseline method.

Blind video consistency: Lei et al. [77] propose a blind video consistency method that can be applied to any computer vision task and which presents the current state-of-the-art in blind video consistency (See Section 2.4). We apply this method on each of the processed outputs from the baseline method to remove temporal artifacts.

5.3.2 Quantitative results on videomatting.com

The videomatting.com benchmark [80] provides 10 video sequences (With 3 variants of trimaps each) that can be submitted online. After submission quantitative metrics will be calculated using the hid-

den ground-truth alphas. These metrics are described in detail in their paper, but we will briefly state their formulation:

- $\mathbf{SSDA} = \frac{1}{T}\sum_t^T \sqrt{\sum_i (\alpha_{i,t} - \hat{\alpha}_{i,t})^2}$,
- $\mathbf{dtSSD} = \frac{1}{T}\sum_t^T \sqrt{\sum_i (\frac{\mathrm{d}\alpha_{i,t}}{\mathrm{d}t} - \frac{\mathrm{d}\hat{\alpha}_{i,t}}{\mathrm{d}t})^2}$,
- $\mathbf{MESSDdt} = \frac{1}{T}\sum_t^T \sum_i |(\alpha_{i,t} - \hat{\alpha}_{i,t})^2 - (\alpha_{i+v_i,t+1} - \hat{\alpha}_{i+v_i,t+1})^2|$,

with T denoting the total number of frames in the sequence, $\alpha_{i,t}$ and $\hat{\alpha}_{i,t}$ the predicted and ground-truth alpha values at pixel i on frame t and v_i the motion vector at pixel i.
We compare the 3 variants of our method described in Section 5.2.3 to the methods described in Section 5.3.1. All submissions to video-matting.com need to be RGBA images and we use the foreground prediction method presented in chapter 4 to predict the foreground RGB colors from all comparisons. Due to this, we can compare against OFD twice. In OFD[1] we apply the step directly on the alpha predictions of the baseline before foreground color predictions. In OFD[2], we apply the step on the RGBA images after the foreground colors have been calculated. We also applied the blind video consistency method to the baseline. However, the method completely fails for some of the sequences and therefore, we could not submit those results to the benchmark.

The results on the benchmark for the *medium* trimap can be seen in Table 5.1. The results for the *narrow* and *wide* trimaps mirror those shown in the table. Visual results can be seen in Figures 5.2, 5.3 and 5.4. As can be seen, none of the methods perform better on average than the baseline method, which also outperforms the previous state-of-the-art on the benchmark. In some sequences, the one frame delay trick slightly outperforms the baseline, but only to

	Rank	City	Rain	Concert	Flowers	Snow	Slava	Vitaliy	Artem	Juneau	Woods
					SSDA						
Baseline	2.0[1]	21.80[3]	**18.65**[1]	15.80[3]	**42.85**[1]	**10.33**[1]	**17.07**[1]	**26.20**[1]	**23.14**[1]	40.87[4]	46.55[4]
OFD[1]	2.7[2]	21.68[2]	43.22[2]	**15.50**[1]	43.54[2]	12.41[2]	17.34[3]	35.69[2]	23.37[3]	47.20[5]	50.90[5]
OFD[2]	3.1[3]	**21.62**[1]	51.28[3]	15.60[2]	43.76[3]	12.45[3]	17.33[2]	35.86[3]	23.36[2]	47.24[6]	50.96[6]
v2	4.0[4]	104.15[5]	58.18[4]	112.12[6]	219.34[4]	37.76[5]	21.44[4]	46.72[4]	55.99[6]	**23.74**[1]	**28.21**[1]
v3	4.4[5]	99.63[4]	58.19[5]	109.13[5]	223.26[5]	37.05[4]	21.86[5]	65.84[6]	44.02[4]	25.87[3]	30.34[3]
v1	4.8[6]	110.02[6]	63.37[6]	100.48[4]	231.13[6]	40.43[6]	24.95[6]	59.56[5]	52.82[5]	24.44[2]	28.90[2]
					dtSSD						
Baseline	2.0	14.13[3]	**21.64**[1]	11.65[3]	**29.30**[1]	**10.41**[1]	**14.55**[1]	**28.77**[1]	**16.90**[1]	38.95[4]	36.63[4]
OFD[1]	3.1	12.93[2]	61.06[2]	10.31[2]	31.27[2]	14.16[2]	15.07[3]	45.91[4]	17.31[3]	51.74[5]	46.88[6]
OFD[2]	3.3	**12.90**[1]	73.17[5]	**10.30**[1]	31.30[3]	14.23[3]	15.05[2]	46.10[5]	17.30[2]	51.75[6]	46.84[5]
v2	3.9	25.96[5]	71.53[3]	25.13[6]	65.42[5]	27.39[5]	15.91[5]	41.34[2]	24.31[6]	**24.14**[1]	**25.09**[1]
v3	4.0	22.92[4]	72.14[4]	23.57[5]	60.33[4]	26.61[4]	15.43[4]	46.52[6]	20.99[4]	25.36[3]	25.20[2]
v1	4.7	28.40[6]	78.58[6]	23.16[4]	66.59[6]	28.60[6]	16.40[6]	44.79[3]	23.11[5]	24.78[2]	25.69[3]
					MESSDdt						
Baseline	2.0	0.33[3]	**0.32**[1]	0.25[3]	**1.23**[1]	**0.10**[1]	**0.21**[1]	**0.48**[1]	**0.35**[1]	1.38[4]	1.73[4]
OFD[1]	2.5	**0.30**[1]	2.94[2]	**0.20**[1]	1.28[2]	0.17[2]	0.22[2]	1.41[3]	0.36[2]	2.08[5]	2.30[5]
OFD[2]	3.6	0.30[2]	4.31[4]	0.21[2]	1.28[3]	0.17[3]	0.22[3]	1.44[4]	0.36[3]	2.08[6]	2.30[6]
v3	4.0	1.84[4]	4.28[3]	2.83[5]	9.69[4]	0.53[4]	0.25[4]	1.79[6]	0.79[4]	0.61[3]	0.77[3]
v2	4.2	2.16[5]	4.34[5]	2.96[6]	10.78[6]	0.54[5]	0.26[5]	1.18[2]	1.13[6]	**0.54**[1]	**0.72**[1]
v1	4.7	2.33[6]	5.03[6]	2.62[4]	10.49[5]	0.61[6]	0.32[6]	1.74[5]	1.07[5]	0.55[2]	0.75[2]

Table 5.1: SSDA, dtSSD and MESSDdt results on the videomatting.com benchmark for *medium* trimaps. Best results are shown in bold, ranking in superscript.

a slight degree. In most sequences, OFD reduces the performance of the baseline. Our proposed video matting method can also not compete with the baseline on average. This is mostly due to the fact that fine details in the alpha matte are often lost as can be seen most prominently in the *City* sequence shown in Figure 5.2. However, our method shows great promise in the *Juneau* (see Figure 5.4) and *Woods* sequences, where it vastly outperforms the baseline. In comparison to our method, the baseline shows parts of the background in the alpha, which can be seen in the hair on the right. Our method completely avoids this issue. We attribute this performance mostly to the lack of available training data and the training methodology. Clearly, in this case the network overfit on the available video data and lost too much of the trained knowledge from the baseline. We show in Section 5.3.4, how an updated training schedule can lead to better results with less details lost.

The results on the videomatting.com also show that **v2** and **v3** of our network implementation consistently outperform **v1**, even though

these variants contain several orders of magnitude less parameters. Clearly, using channel-separated convolutions is beneficial in the video matting task.

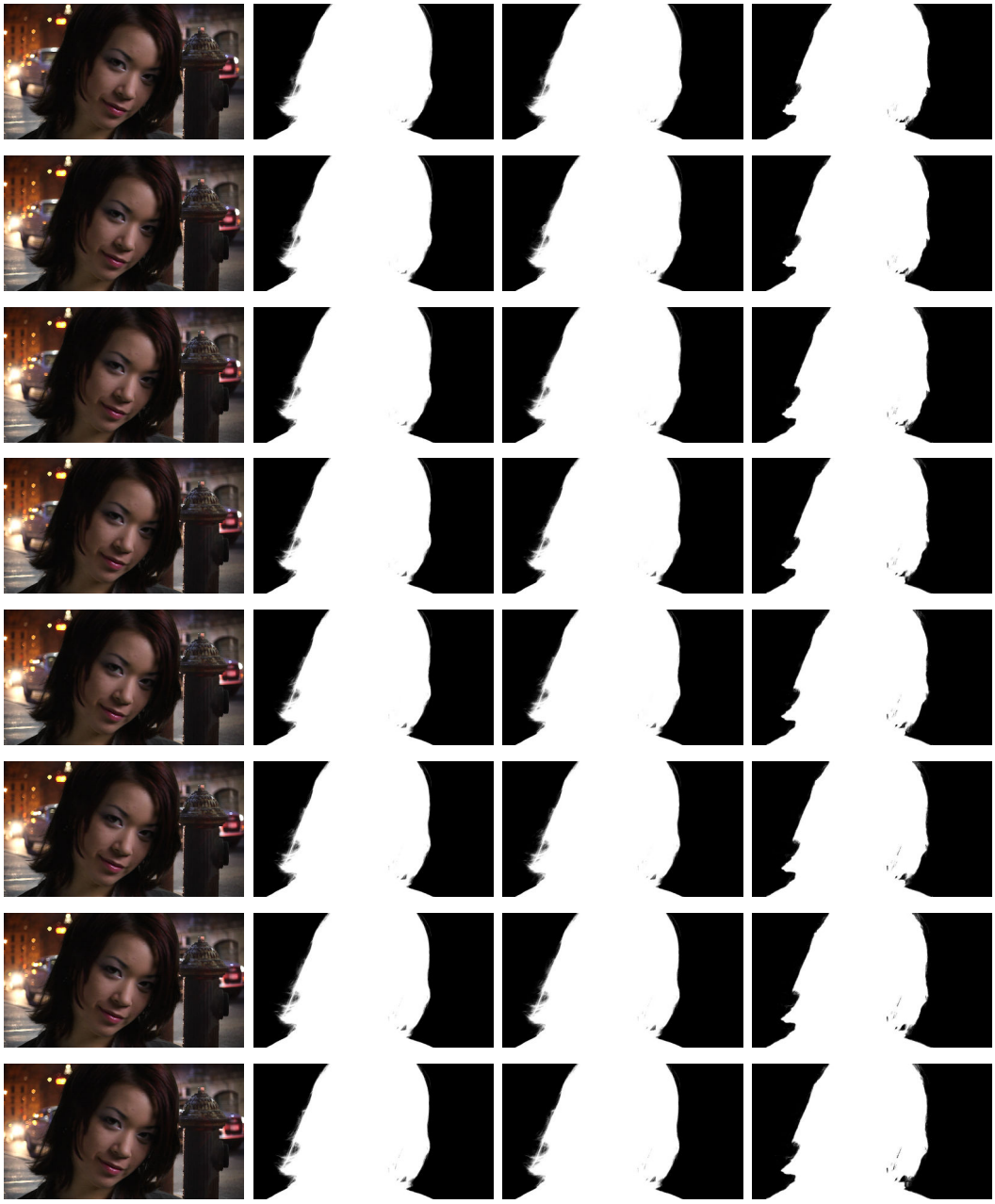

Figure 5.2: Visual comparison on the videomatting.com benchmark on frames 5 to 12 of the *City* sequence using *medium* trimaps. From left to right: Input frame, baseline, OFD[1] and v3 results.

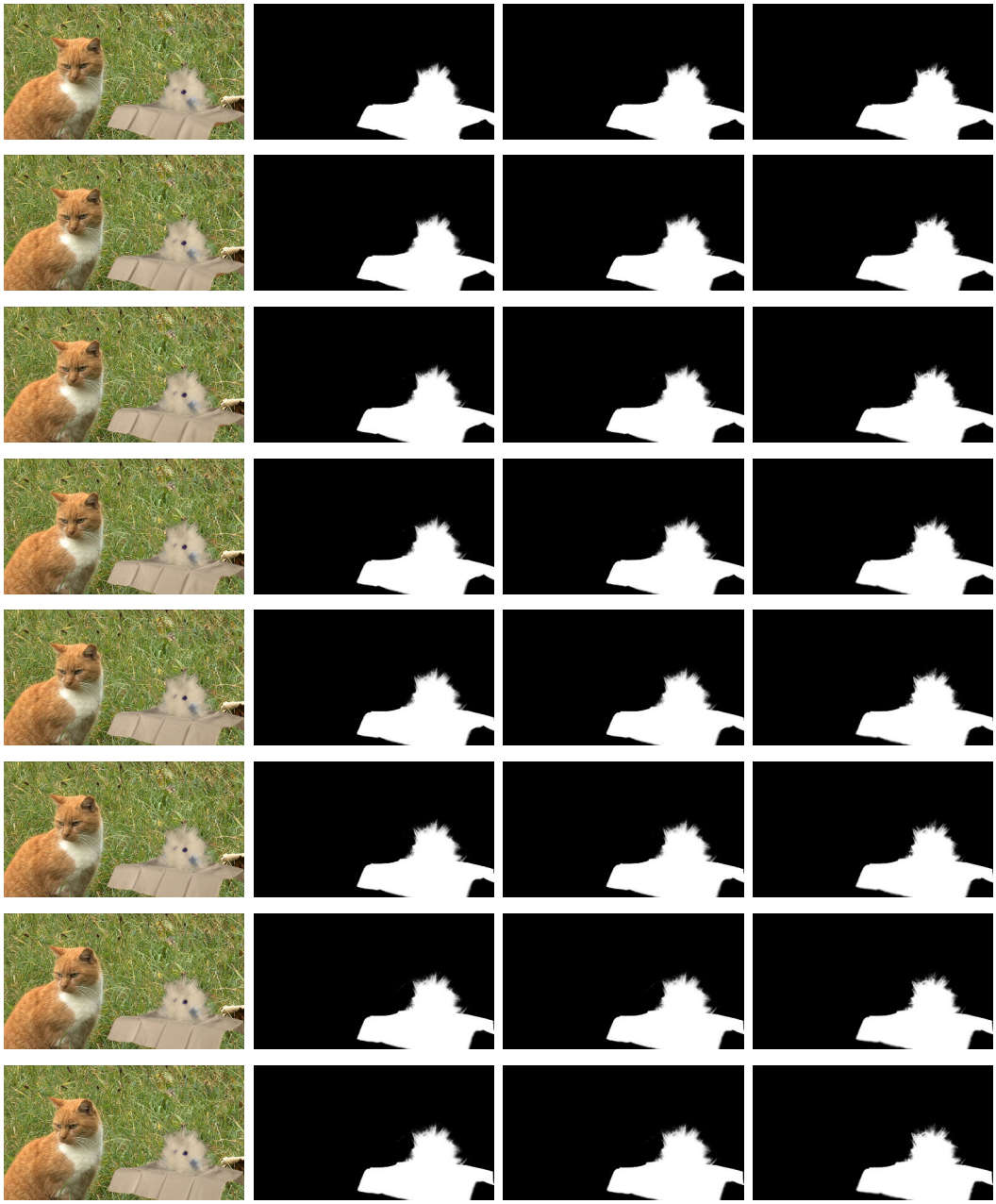

Figure 5.3: Visual comparison on the videomatting.com benchmark on frames 5 to 12 of the *Slava* sequence using *medium* trimaps. From left to right: Input frame, baseline, OFD[1] and v3 results.

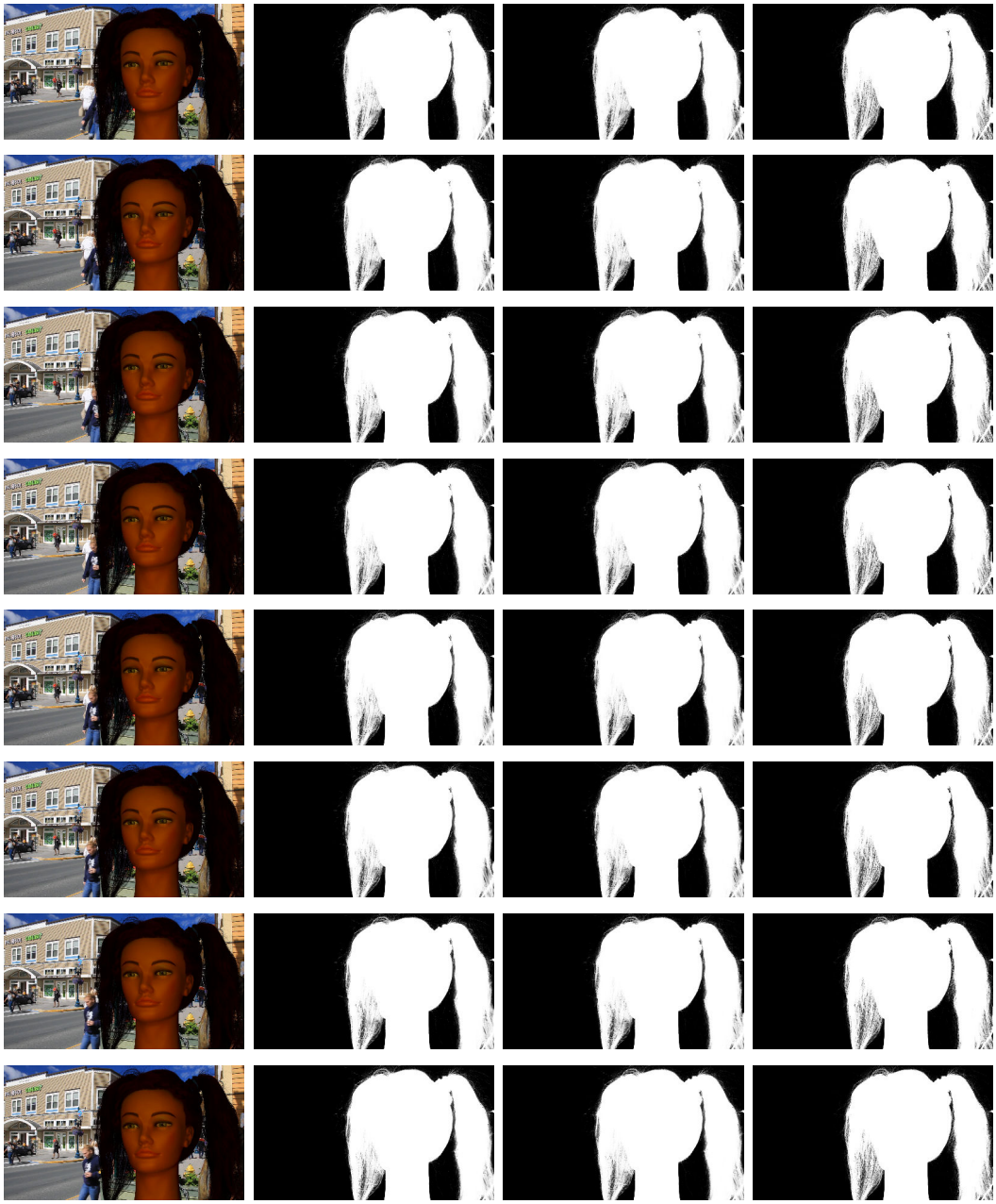

Figure 5.4: Visual comparison on the videomatting.com benchmark on frames 5 to 12 of the *Juneau* sequence using *medium* trimaps. From left to right: Input frame, baseline, OFD[1] and v3 results.

5.3.3 Blind video consistency

As stated previously, the blind video consistency method sometimes fails completely on some test cases and could not be submitted for quantitative results. However, even on the sequences where it

works, the results are worse than the baseline as can be seen in Figure 5.5. In the *City* sequence, the results the blind video consistency method are overly smooth and the method does not retain any of the sharp alphas in the hair of the foreground object. In the *Juneau* sequence, the method fails to remove parts of the background showing in the alpha as can be seen in the hair on the left and additionally introduces grid-like artifacts in the top-left of the hair.

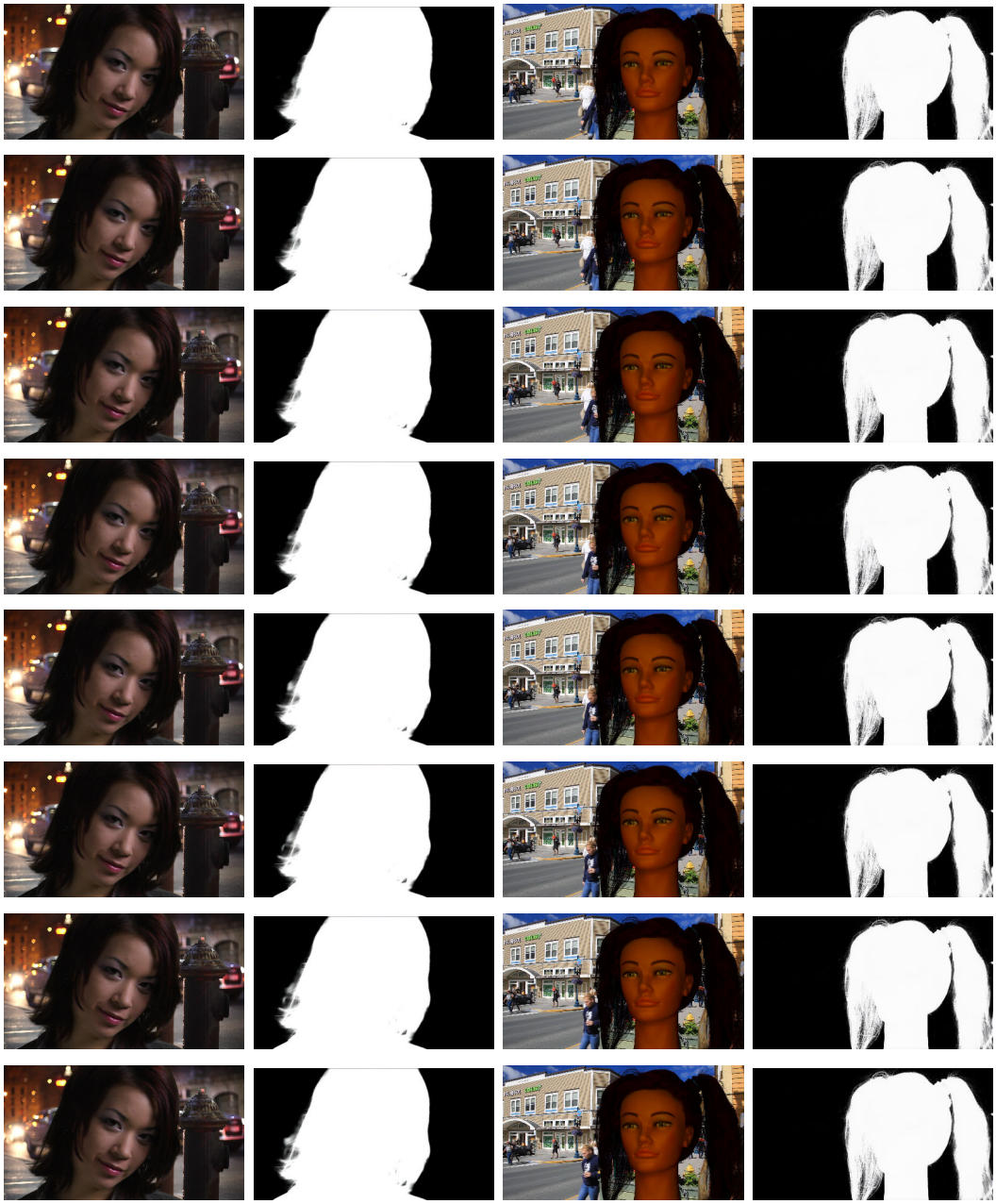

Figure 5.5: Visual comparison of the blind video consistency method [77] proposed by Lei et al. on frames 5 to 12 of the *City* and *Juneau* sequences.

5.3.4 Updated training schedule

As can be seen on the results of the videomatting.com benchmark and especially in Figure 5.2, our network tends to loose a lot of the finer details of the alpha matte compared to the baseline. To alleviate this problem, we update our training schedule to train on both

the video matting data and also on the matting dataset released by Xu et al. [54]. Every training iteration, a new batch is randomly sampled from either the video data with a sequence length of 10, or from the single image dataset with a sequence length of 1 and a 10 times larger batch size. Due to this, the network is less likely to overfit on the video data and able to keep more of the finer details in the alpha matte. A comparison of this is shown in Figure 5.6 on the *City* and *Juneau* sequences. As can be seen, the updated training schedule leads to more details in the *City* sequence and simultaneously keeps the great performance our method already showed on the *Juneau* sequence. Clearly, it is beneficial to implement the training in a way to learn from both video sequences and single images.

Figure 5.6: Visual comparison between the original and updated training schedule on frames 5 to 12 of the *City* and *Juneau* sequences.

5.4 Conclusion

In this chapter, we evaluate methods to introduce temporal consistency in video matting methods. Due to the power of convolutional neural networks in computer vision tasks, deep neural networks

trained on single images for the natural image matting task outperform classical approaches, even those that have been explicitly designed for the video matting task. Even so, these new CNNs suffer from temporal artifacts that can decrease the perceptual quality of the results to a large degree. We investigate several state-of-the-art approaches to introduce temporal consistency as a post-processing step in video matting results, as well as a neural network trained on the video matting task. We show that blind video consistency methods are not suitable for video matting and degrade the results or fail completely. Other filtering approaches to remove flickering also tend to decrease the matting quality. On average, our method does not outperform the baseline method, but it shows very promising results in several of the test sequences and we believe further work in this direction can lead to better results than the state-of-the-art.

Chapter 6

Conclusions and Outlook

Natural image matting is still an ongoing research problem and important for many practical applications for high-quality and casual creatives. In this thesis, methods were proposed to further develop the state-of-the-art in natural image and video matting, as well as the related problem of foreground color prediction. This chapter serves as summary and conclusion of this thesis and is organized as follows: First, all contributions made in this thesis are revisited and summarized as a whole. Afterwards a look forward is given, with potential further research outlined.

6.1 Conclusion

Natural image matting is a very difficult problem to solve. Even small differences between predicted alpha mattes can have a large impact on the perceived quality of the alpha. This is especially true in high-frequency regions such as hair, where the alpha is expected to show sharp edges and overly smooth predictions can have a large negative impact on the quality. In chapter 3, we present a novel deep convolutional neural network and training pipeline to rectify this issue. We model our method as the first generative adversarial network for natural image matting, which helps train the network predict alphas that lead to well made compositions and visually appealing results. We beat the then state-of-the-art in the perceptually motivated gradient metric for natural image matting on the alphamatting.com benchmark and show qualitatively that our alpha predictions work especially well in high-frequency regions.

For high-quality compositions, however, predicting excellent alpha mattes is often not good enough if the goal is to create new compositions out of the foreground object. This is due to the fact that the colors in the original RGB image in the transparent regions are always a mix between the true foreground and background colors. To create faithful compositions, it is therefore necessary to also predict the true foreground colors alongside the alpha matte. In chapter 4, we present a novel algorithm to predict the foreground colors of an object. Given an initial alpha prediction, we create a recurrent inference machine to essentially solve the inverse composition problem and divide the input RGB image into alpha, foreground and background. We show quantitatively and qualitatively that our method outperforms the state-of-the-art in foreground color pre-

diction. Furthermore, our method is well suited to predict colors even from very large resolution images that may not fit onto GPU memory in their entirety. Finally, our method also allows user interaction, by allowing the direct alteration of candidate solutions, which can propagate these changes forwards.

For artists and casual creatives, video is just as important a medium as single images. Naturally, natural image matting is therefore also often applied to video sequences. Modern neural network methods trained for single images have shown to drastically outperform classical approaches and even classical video matting methods. Due to the fact that they have been designed for single images only, however, these approaches lack temporal consistency across a video sequence. In chapter 5, we evaluate several approaches to adding temporal consistency to single image natural image matting methods. We also present the first deep convolutional neural network for video matting. We introduce a way to leverage a network trained on single images for video matting by adding 3D convolutions within the network. While our approach does not outperform the state-of-the-art, we show promising results in some of the test sequences and believe that our method can serve as groundwork for further research in this area.

These three chapters correspond to and answer the research objectives stated in 1.3. The success of our proposed method in chapter 3 and the manifold successive methods answer the first research objective of this thesis. Our method proposed in chapter 4 answers the second question and shows how high-quality compositions can be created. We investigate the third question in chapter 5 and lay

the groundwork for a convolutional neural network to solve the video matting task.

6.2 Future Work

Many of the recent state-of-the-art in natural image matting have experimented on using a variety of different loss functions. However, it seems that in general the L1-loss is sufficient to properly train networks for good alpha predictions. In chapter 3, we introduce the first generative adversarial network for natural image matting. However, there have been many advances using GANs in other computer vision tasks in recent years and it would be interesting to investigate new GAN techniques and improved training pipelines for the natural image matting task.

In chapter 4, we introduce a recurrent inference machine for foreground color prediction. The recurrent nature of the algorithm allows for some manual edition of candidate solutions that is not given in traditional networks. Nevertheless, there are certainly improvements to be made regarding the user interaction to allow artists to achieve the ultra high-quality results that they are used to through chroma keying. Finding a way to iteratively improve alpha matting and foreground color estimation results without relying on changing the results on a pixel by pixel basis and to potentially propagate edits through subsequent frames in a video sequence would go a long way to make natural image matting suitable for high-quality studio environments.

The biggest problem for video matting is the lack of a large dataset.

The available dataset only contains 3 video sequences available for training, which is not good enough to fully train a network for the video matting task. One way to go forward in this would be to create a new, larger, video matting dataset that contains enough sequences. A different solution would be to find a way to re-frame the training as a weakly supervised training and make use of other existing video sequences such as semantic video segmentation. Nevertheless, there are certainly ways to improve the performance of video matting and the task would benefit from future research.

6.3 Perspectives

In this dissertation we explore deep learning techniques for natural image matting. We have proposed several new methods to predict high-quality alpha mattes, as well as foreground and background colors. Combined, these methods can be used to extract the foreground object from images and composite realistic looking new images on new backgrounds. This is not the limit of their use cases, however, and several image processing tasks can benefit from the proposed methods as well. Many video conference software allow for the blurring of the background for example. However, this is often done through semantic segmentation and often introduces artifacts around the border of the person. Natural image matting methods such as the ones proposed in this work could be used to generate much higher quality results. Instead of blurring the background, many other effects could also be applied, such as recoloring of the background. For this to properly work, however, the foreground and background colors of the image are necessary, which

can be achieved through the method proposed in this work. Naturally, hand in hand with any technological advancement comes the danger of misusing it. So called *deep fakes* are images that have been automatically altered, most often by replacing the face of the subject of the image. One can imagine several ways in which technology such as this can be misused by propagating misinformation or slander. The methods in this work are designed to help artists, professional or otherwise, manipulate images. The easier such methods are to use and the better their results look, the easier it is to misuse them as well. While this is not a problem that can be solved by scientists, it is maybe something to keep in mind.

Appendix A

Abbreviations

Short Term	Expanded Term
ASPP	Atrous Spatial Pyramid Pooling
CGI	Computer Generated Imagery
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
FLOPs	Floating-point Operations Per second
FPS	Frames Per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
RGB	Red Green Blue
SAD	Sum of Absolute Differences
SSIM	Structural Similarity
VFX	Visual Effects

Appendix B

Additional results

This appendix serves to showcase additional results for chapter 3 and 4.

B.1 Additional results for chapter 3

This section contains some additional results for chapter 3.

B.1.1 Additional comparison results on the Composition-1k test dataset

We show further comparisons on the Composition-1k test dataset in Figure B.1 and B.2.

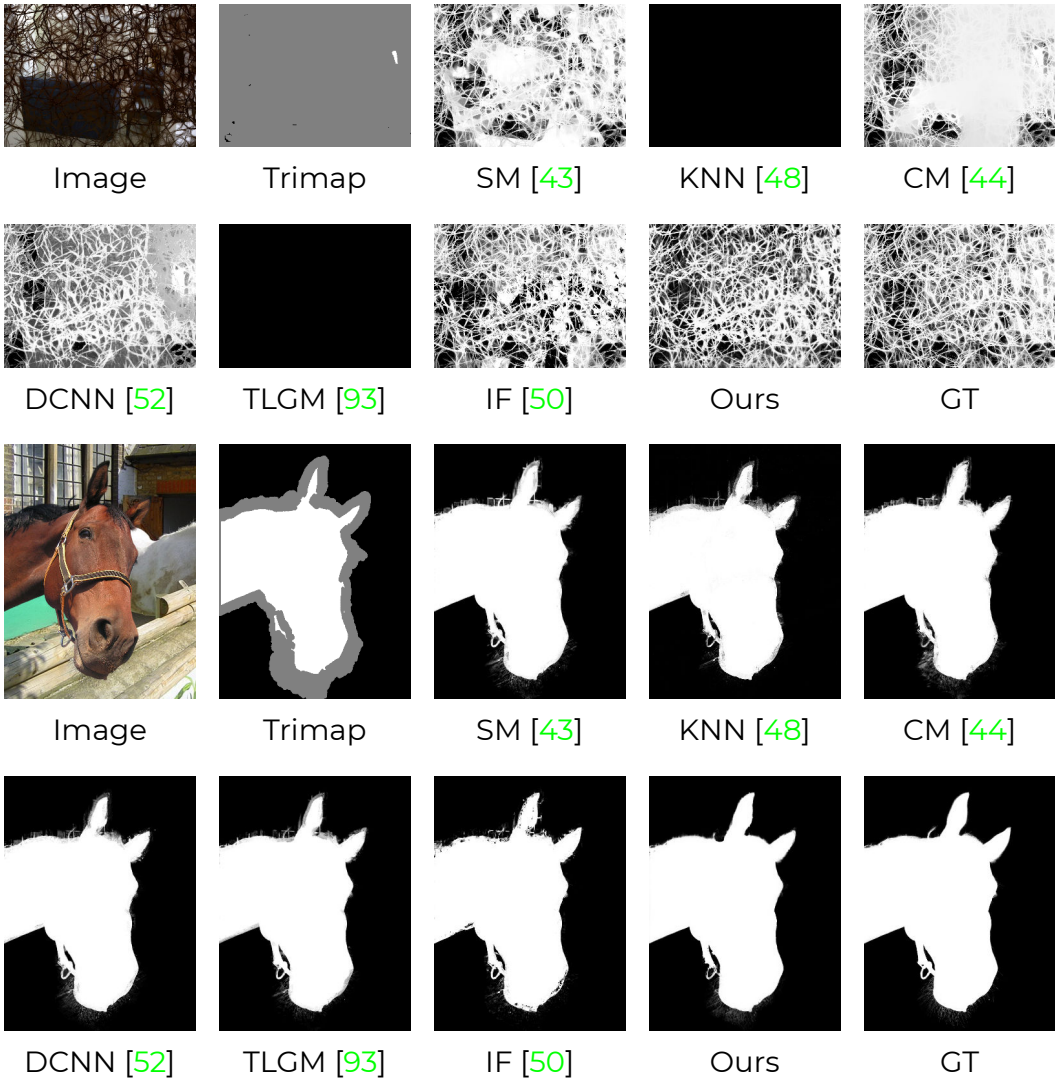

Figure B.1: Comparison results on the Composition-1k test dataset.

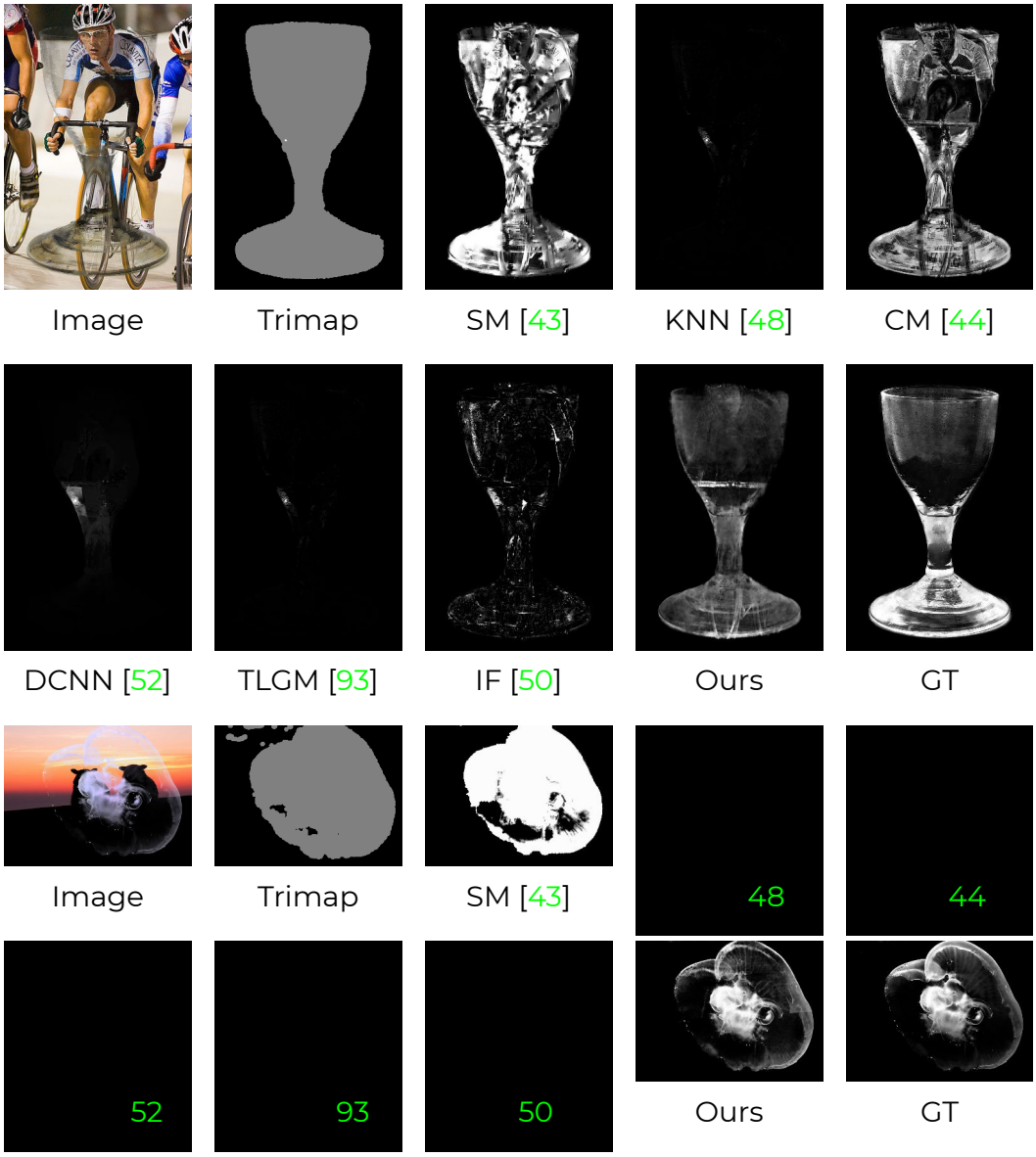

Figure B.2: Comparison results on the Composition-1k test dataset.

B.2 Additional results for chapter 4

This section contains some additional results for chapter 4.

B.2.1 Additional comparison results on the Composition-1k test dataset

We show further comparisons of our method to Context-Aware Matting [60] and Samplenet [55] in Figures B.3 and B.4

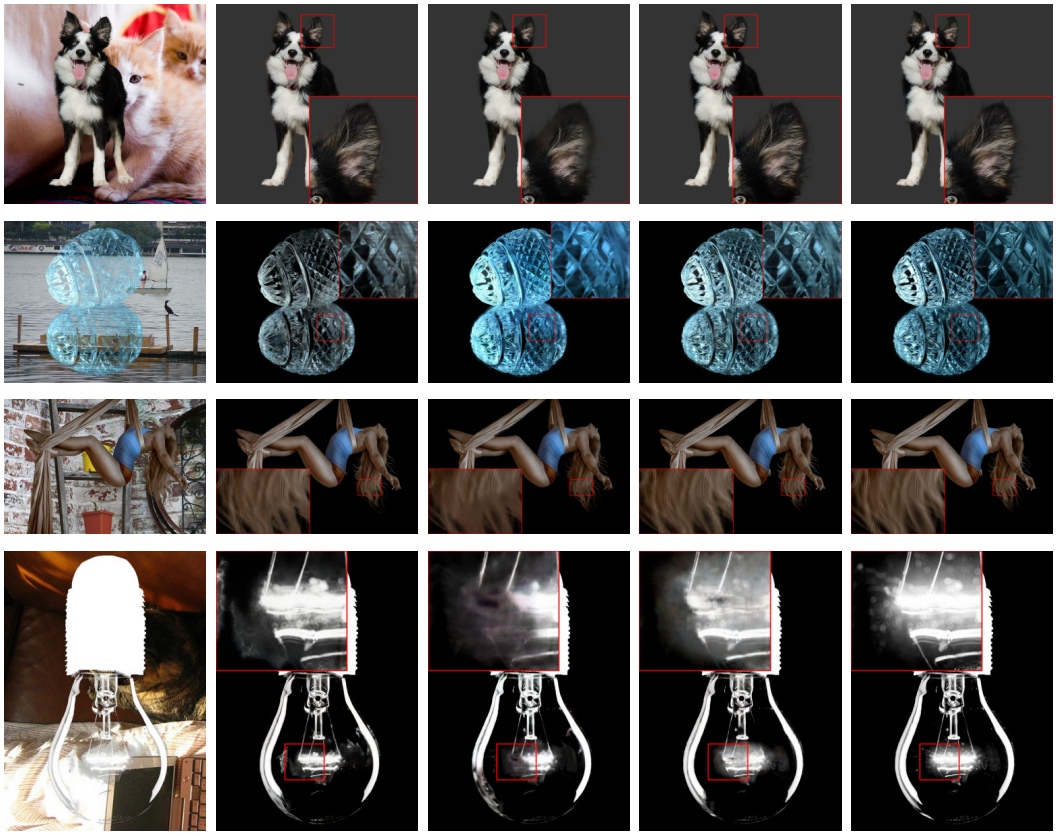

Figure B.3: Visual comparison on the Composition-1k dataset. From left to right: Input image, Compositions from Context-Aware Matting [60], Samplenet [55], Ours, Ground-truth.

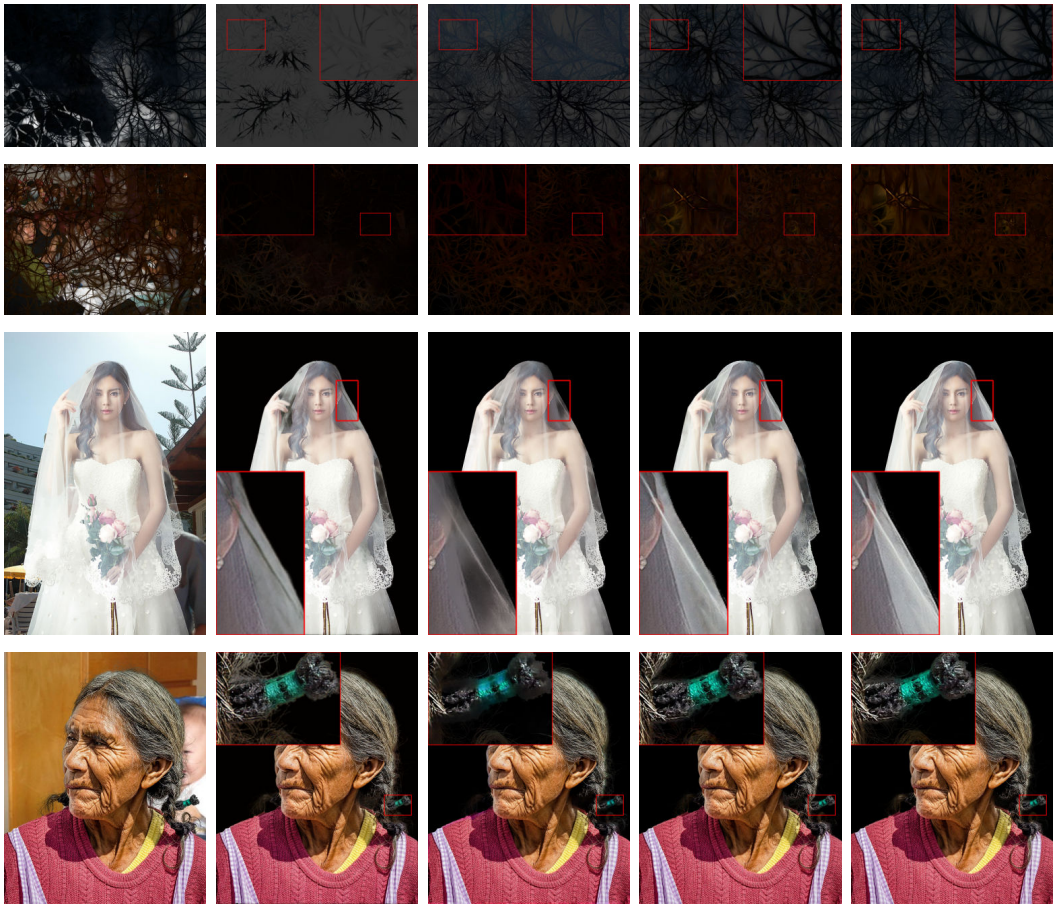

Figure B.4: Visual comparison on the Composition-1k dataset. From left to right: Input image, Compositions from Context-Aware Matting [60], Samplenet [55], Ours, Ground-truth.

B.2.2 Visualization of the manual editing

We show the automatic predictions, the manual edits done and the refined results in Figure B.5.

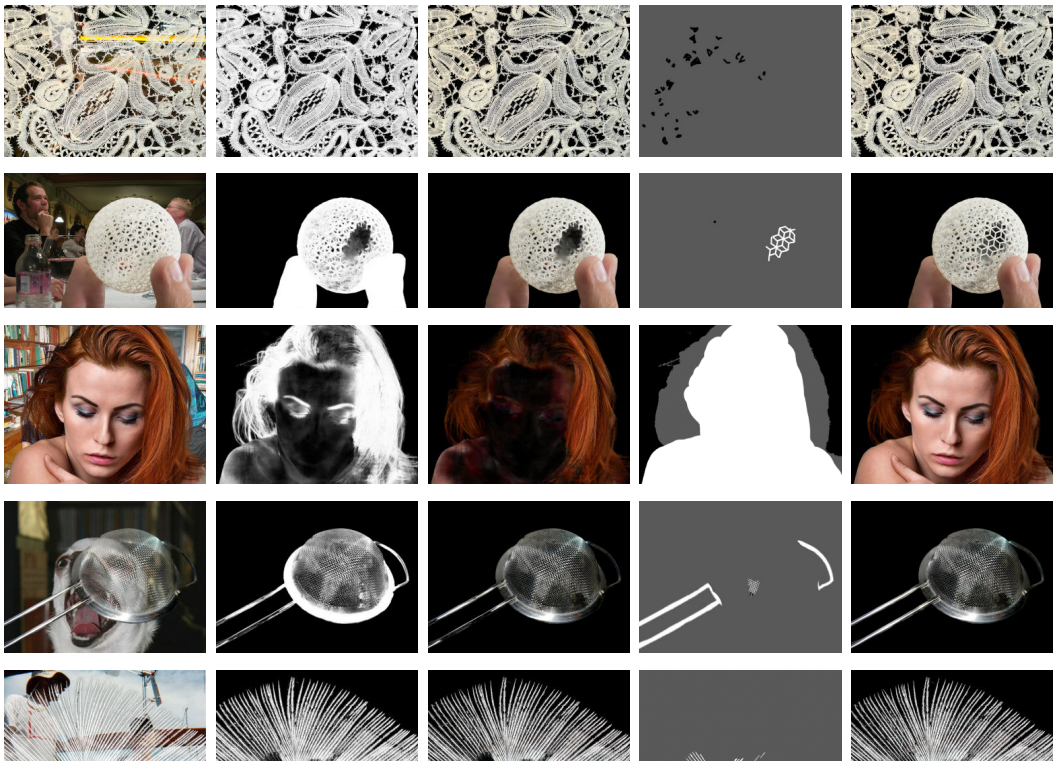

Figure B.5: Visualization of the manual editing process. From left to right: Input image, predicted alpha, composition from automatically predicted alpha and foreground, editing mask for the alpha, composition from the updated alpha and newly predicted foreground.
As can be seen, the faulty automatic alpha predictions lead to unappealing compositions. However, a small amount of manual editing is sufficient to recover foreground color predictions that lead, alongside the new alpha, to much better compositions.
Please note that the amount of manual editing was deliberately kept low.

B.2.3 User study results

We show some example images from the user study we conducted in Figure B.6.

Figure B.6: Example images from the user study. From left to right: Input image, Context-Aware Matting [60], our result with the alpha prediction from Context-Aware as input, Samplenet [55], our result with the alpha prediction from Samplenet as input.
As can be seen in the comparison with Samplenet, the color predictions from Samplenet are somewhat smoothed out on the edges, which can lead to more unappealing compositions. For the comparison with Context-Aware Matting, the differences in color predictions are more difficult to spot.

Bibliography

[1] B. Zhu, Y. Chen, J. Wang, S. Liu, B. Zhang, and M. Tang, "Fast deep matting for portrait animation on mobile phone," in *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017* (Q. Liu, R. Lienhart, H. Wang, S. K. Chen, S. Boll, Y. P. Chen, G. Friedland, J. Li, and S. Yan, eds.), pp. 297–305, ACM, 2017.

[2] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia, "Deep automatic portrait matting," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9905 of *Lecture Notes in Computer Science*, pp. 92–107, Springer, 2016.

[3] Z. Ke, K. Li, Y. Zhou, Q. Wu, X. Mao, Q. Yan, and R. W. Lau, "Is a green screen really necessary for real-time portrait matting?," *ArXiv*, vol. abs/2011.11961, 2020.

[4] "Green screen: It can break a man.: Movies without cgi." https://knowyourmeme.com/photos/1015016-movies-without-cgi, Feb 2018. Accessed: 2020-12-30.

[5] J. Wang and M. F. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *10th IEEE*

International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China, pp. 936–943, IEEE Computer Society, 2005.

[6] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski, "A bayesian approach to digital matting," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, pp. 264–271, IEEE Computer Society, 2001.

[7] Q. Zhu, L. Shao, X. Li, and L. Wang, "Targeting accurate object extraction from an image: A comprehensive study of natural image matting," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 2, pp. 185–207, 2015.

[8] S. Lutz, K. Amplianitis, and A. Smolic, "Alphagan: Generative adversarial networks for natural image matting," in *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, p. 259, BMVA Press, 2018.

[9] S. Lutz and A. Smolic, "Foreground color prediction through inverse compositing," in *Winter Conference on Applications of Computer Vision 2021 (WACV 2021)*, 2021.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[11] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, “Salnet360: Saliency maps for omni-directional images with CNN,” *Signal Process. Image Commun.*, vol. 69, pp. 26–34, 2018.

[12] S. Lutz, M. Davey, and A. Smolic., “Deep convolutional neural networks for estimating lens distortion parameters,” in *Irish Machine Vision and Image Processing Conference (IMVIP) 2019*, 2019.

[13] X. Zheng, T. Chalasani, K. Ghosal, S. Lutz, and A. Smolic, “Stada: Style transfer as data augmentation,” in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2019, Volume 4: VISAPP, Prague, Czech Republic, February 25-27, 2019* (A. Trémeau, G. M. Farinella, and J. Braz, eds.), pp. 107–114, SciTePress, 2019.

[14] M. Hudon, S. Lutz, R. Pagés, and A. Smolic, “Augmenting hand-drawn art with global illumination effects through surface inflation,” in *The 16th ACM SIGGRAPH European Conference on Visual Media Production*, 2019.

[15] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*. Adaptive computation and machine learning, MIT Press, 2016.

[16] O. E. David and N. S. Netanyahu, “Deeppainter: Painter classification using deep convolutional autoencoders,” in *Artificial Neural Networks and Machine Learning - ICANN 2016 - 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II* (A. E. P. Villa, P. Masulli, and A. J. P. Rivero, eds.), vol. 9887 of *Lecture Notes in Computer Science*, pp. 20–28, Springer, 2016.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1106–1114, 2012.

[19] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, “A review on deep learning techniques applied to semantic segmentation,” *CoRR*, vol. abs/1704.06857, 2017.

[20] 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 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016.

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269, IEEE Computer Society, 2017.

[22] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I* (D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, 2014.

[23] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3431–3440, IEEE Computer Society, 2015.

[24] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 2852–2860, 2012.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 1–9, IEEE Computer Society, 2015.

[26] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[27] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III* (N. Navab, J. Hornegger, W. M. W. III, and A. F. Frangi, eds.), vol. 9351 of *Lecture Notes in Computer Science*, pp. 234–241, Springer, 2015.

[28] S. Zhou, J. Wu, Y. Wu, and X. Zhou, “Exploiting local structures with the kronecker layer in convolutional networks,” *CoRR*, vol. abs/1512.09194, 2015.

[29] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[30] L. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *CoRR*, vol. abs/1706.05587, 2017.

[31] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), vol. 11211 of *Lecture Notes in Computer Science*, pp. 833–851, Springer, 2018.

[32] Y. Guo, Y. Li, L. Wang, and T. Rosing, “Depthwise convolution is all you need for learning multiple visual domains,” in

The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 8368–8375, AAAI Press, 2019.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1800–1807, IEEE Computer Society, 2017.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org, 2015.

[35] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5967–5976, IEEE Computer Society, 2017.

[36] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9908 of *Lecture Notes in Computer Science*, pp. 318–335, Springer, 2016.

[37] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[38] J. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2242–2251, IEEE Computer Society, 2017.

[39] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906 of *Lecture Notes in Computer Science*, pp. 694–711, Springer, 2016.

[40] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *CoRR*, vol. abs/1607.08022, 2016.

[41] O. Juan and R. Keriven, “Trimap segmentation for fast and user-friendly alpha matting,” in *Variational, Geometric, and Level Set Methods in Computer Vision, Third International Workshop, VLSM 2005, Beijing, China, October 16, 2005, Proceedings* (N. Paragios, O. D. Faugeras, T. Chan, and C. Schnörr, eds.), vol. 3752 of *Lecture Notes in Computer Science*, pp. 186–197, Springer, 2005.

[42] H. Chang, Q. Yang, and C. Pan, “An iterative bayesian approach for digital matting,” in *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*, pp. 122–125, IEEE Computer Society, 2006.

[43] E. S. L. Gastal and M. M. Oliveira, “Shared sampling for real-time alpha matting,” *Comput. Graph. Forum*, vol. 29, no. 2, pp. 575–584, 2010.

[44] E. Shahrian, D. Rajan, B. L. Price, and S. Cohen, “Improving image matting using comprehensive sampling sets,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pp. 636–643, IEEE Computer Society, 2013.

[45] X. Feng, X. Liang, and Z. Zhang, “A cluster sampling method for image matting via sparse coding,” in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906 of *Lecture Notes in Computer Science*, pp. 204–219, Springer, 2016.

[46] L. Karacan, A. Erdem, and E. Erdem, “Alpha matting with kl-divergence-based sparse sampling,” *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4523–4536, 2017.

[47] A. Levin, D. Lischinski, and Y. Weiss, “A closed-form solution to natural image matting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 228–242, 2008.

[48] Q. Chen, D. Li, and C. Tang, “KNN matting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2175–2188, 2013.

[49] X. Chen, D. Zou, Q. Zhao, and P. Tan, “Manifold preserving edit propagation,” *ACM Trans. Graph.*, vol. 31, no. 6, pp. 132:1–132:7, 2012.

[50] Y. Aksoy, T. O. Aydin, and M. Pollefeys, “Designing effective inter-pixel information flow for natural image matting,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 228–236, IEEE Computer Society, 2017.

[51] X. Chen, D. Zou, S. Z. Zhou, Q. Zhao, and P. Tan, “Image matting with local and nonlocal smooth priors,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pp. 1902–1907, IEEE Computer Society, 2013.

[52] D. Cho, Y. Tai, and I. Kweon, “Natural image matting using deep convolutional neural networks,” in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906 of *Lecture Notes in Computer Science*, pp. 626–643, Springer, 2016.

[53] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, “A perceptually motivated online benchmark for image matting,” in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 1826–1833, IEEE Computer Society, 2009.

[54] N. Xu, B. L. Price, S. Cohen, and T. S. Huang, “Deep image matting,” in *2017 IEEE Conference on Computer Vision and*

Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 311–320, IEEE Computer Society, 2017.

[55] J. Tang, Y. Aksoy, C. Öztireli, M. H. Gross, and T. O. Aydin, "Learning-based sampling for natural image matting," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 3055–3063, Computer Vision Foundation / IEEE, 2019.

[56] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 5505–5514, IEEE Computer Society, 2018.

[57] S. Cai, X. Zhang, H. Fan, H. Huang, J. Liu, J. Liu, J. Liu, J. Wang, and J. Sun, "Disentangled image matting," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 8818–8827, IEEE, 2019.

[58] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 802–810, 2015.

[59] H. Lu, Y. Dai, C. Shen, and S. Xu, "Indices matter: Learning to index for deep image matting," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Ko-*

rea (South), October 27 - November 2, 2019, pp. 3265–3274, IEEE, 2019.

[60] Q. Hou and F. Liu, "Context-aware image matting for simultaneous foreground and alpha estimation," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4129–4138, IEEE, 2019.

[61] Y. Li and H. Lu, "Natural image matting via guided contextual attention," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 11450–11457, AAAI Press, 2020.

[62] Y. Zhang, L. Gong, L. Fan, P. Ren, Q. Huang, H. Bao, and W. Xu, "A late fusion CNN for digital matting," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 7469–7478, Computer Vision Foundation / IEEE, 2019.

[63] Y. Qiao, Y. Liu, X. Yang, D. Zhou, M. Xu, Q. Zhang, and X. Wei, "Attention-guided hierarchical structure aggregation for image matting," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 13673–13682, IEEE, 2020.

[64] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern*

Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 5987–5995, IEEE Computer Society, 2017.

[65] Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski, "Video matting of complex scenes," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 243–248, 2002.

[66] D. Li, Q. Chen, and C. Tang, "Motion-aware KNN laplacian for video matting," in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pp. 3599–3606, IEEE Computer Society, 2013.

[67] M. Sindeev, A. Konushin, and C. Rother, "Alpha-flow for video matting," in *Computer Vision - ACCV 2012 - 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part III* (K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, eds.), vol. 7726 of *Lecture Notes in Computer Science*, pp. 438–452, Springer, 2012.

[68] J. Johnson, E. S. Varnousfaderani, H. Cholakkal, and D. Rajan, "Sparse coding for alpha matting," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3032–3043, 2016.

[69] D. Zou, X. Chen, G. Cao, and X. Wang, "Video matting via sparse and low-rank representation," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1564–1572, IEEE Computer Society, 2015.

[70] I. Choi, M. Lee, and Y. Tai, "Video matting using multi-frame nonlocal matting laplacian," in *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI* (A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.),

vol. 7577 of *Lecture Notes in Computer Science*, pp. 540–553, Springer, 2012.

[71] D. Zou, X. Chen, G. Cao, and X. Wang, "Unsupervised video matting via sparse and low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1501–1514, 2020.

[72] G. Cao, J. Li, Z. He, and X. Chen, "Divide and conquer: A self-adaptive approach for high-resolution image matting," in *2016 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 24–30, 2016.

[73] G. Cao, J. Li, X. Chen, and Z. He, "Patch-based self-adaptive matting for high-resolution image and video," *Vis. Comput.*, vol. 35, no. 1, pp. 133–147, 2019.

[74] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video snapcut: robust video object cutout using localized classifiers," *ACM Trans. Graph.*, vol. 28, no. 3, p. 70, 2009.

[75] T. Wang, M. Liu, J. Zhu, N. Yakovenko, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 1152–1164, 2018.

[76] A. Mallya, T. Wang, K. Sapra, and M. Liu, "World-consistent video-to-video synthesis," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VIII* (A. Vedaldi, H. Bischof, T. Brox, and

J. Frahm, eds.), vol. 12353 of *Lecture Notes in Computer Science*, pp. 359–378, Springer, 2020.

[77] C. Lei, Y. Xing, and Q. Chen, "Blind video temporal consistency via deep video prior," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[78] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V* (D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8693 of *Lecture Notes in Computer Science*, pp. 740–755, Springer, 2014.

[79] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[80] M. Erofeev, Y. Gitman, D. S. Vatolin, A. Fedorov, and J. Wang, "Perceptually motivated benchmark for video matting," in *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015* (X. Xie, M. W. Jones, and G. K. L. Tam, eds.), pp. 99.1–99.12, BMVA Press, 2015.

[81] A. Levinshtein, C. Chang, E. Phung, I. Kezele, W. Guo, and P. Aarabi, "Real-time deep hair matting on mobile devices," in *15th Conference on Computer and Robot Vision, CRV 2018, Toronto, ON, Canada, May 8-10, 2018*, pp. 1–7, IEEE Computer Society, 2018.

[82] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[83] S. Seo, S. Choi, M. Kersner, B. Shin, H. Yoon, H. Byun, and S. Ha, "Towards real-time automatic portrait matting on mobile devices," *CoRR*, vol. abs/1904.03816, 2019.

[84] J. Liu, Y. Yao, W. Hou, M. Cui, X. Xie, C. Zhang, and X. Hua, "Boosting semantic human matting with coarse annotations," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 8560–8569, IEEE, 2020.

[85] G. Chen, K. Han, and K. K. Wong, "Tom-net: Learning transparent object matting from a single image," *CoRR*, vol. abs/1803.04636, 2018.

[86] D. Shin and Y. Chen, "Deep garment image matting for a virtual try-on system," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pp. 3141–3144, IEEE, 2019.

[87] S. Sengupta, V. Jayaram, B. Curless, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Background matting: The world is your green screen," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 2288–2297, IEEE, 2020.

[88] Z. Zou, W. Li, T. Shi, Z. Shi, and J. Ye, "Generative adversarial training for weakly supervised cloud matting," in *2019*

IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pp. 201–210, IEEE, 2019.

[89] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[90] S. Hong, S. Kwak, and B. Han, "Weakly supervised learning with deep convolutional neural networks for semantic segmentation: Understanding semantic layout of images with minimum human supervision," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 39–49, 2017.

[91] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, 2014.

[92] Q. Chen, D. Li, and C. Tang, "KNN matting," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 869–876, IEEE Computer Society, 2012.

[93] C. Li, P. Wang, X. Zhu, and H. Pi, "Three-layer graph framework with the sumd feature for alpha matting," *Comput. Vis. Image Underst.*, vol. 162, pp. 34–45, 2017.

[94] P. Putzky and M. Welling, “Recurrent inference machines for solving inverse problems,” *CoRR*, vol. abs/1706.04008, 2017.

[95] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *CoRR*, vol. abs/1701.07875, 2017.

[96] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5767–5777, 2017.

[97] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[98] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 7354–7363, PMLR, 2019.

[99] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Free-form image inpainting with gated convolution,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4470–4479, IEEE, 2019.

[100] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo, "Foreground-aware image inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 5840–5848, Computer Vision Foundation / IEEE, 2019.

[101] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[102] G. H. Golub and H. A. Van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1-2, pp. 35–65, 2000.

[103] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *CoRR*, vol. abs/1705.10941, 2017.

[104] I. Vasilev, *Advanced Deep Learning with Python*. Packt, Dec 2019.

[105] K. Cho, B. van Merrienboer, Ç. Gülçehre, 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (A. Moschitti, B. Pang, and W. Daelemans, eds.), pp. 1724–1734, ACL, 2014.

[106] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[107] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," *CoRR*, vol. abs/1806.03589, 2018.

[108] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[109] B. L. Price, B. S. Morse, and S. Cohen, "Simultaneous foreground, background, and alpha estimation for image matting," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pp. 2157–2164, IEEE Computer Society, 2010.

[110] H. Tang, Y. Huang, M. Jing, Y. Fan, and X. Zeng, "Very deep residual network for image matting," in *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pp. 4255–4259, IEEE, 2019.

[111] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. H. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 724–732, IEEE Computer Society, 2016.

[112] 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*

2019, Seoul, Korea (South), October 27 - November 2, 2019, pp. 5551–5560, IEEE, 2019.

[113] 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 Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 6450–6459, IEEE Computer Society, 2018.

[114] E. Shahrian and D. Rajan, "Weighted color and texture sample selection for image matting," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 718–725, IEEE Computer Society, 2012.

[115] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.