

Hyperparameter Optimisation Methods For Transformer Neural Nets

Patrick Haughey¹, Douglas Leith²

¹ Trinity College Dublin, Ireland haughepa@tcd.ie

² Trinity College Dublin, Ireland doug.leith@tcd.ie

Abstract. Previous studies comparing hyperparameter optimisation (HPO) algorithms are becoming stale, with the result that machine learning practitioners have little guidance as to which HPO methods are preferable for transformer models with text data. The present study is a step towards addressing this gap. It compares five popular HPO methods, under fair and reproducible conditions, evaluating their performance for hyperparameter tuning of a transformer model trained on text data. To the best of our knowledge, this is the first apples-to-apples comparison of these HPO methods, and also the first HPO comparison study using transformer neural nets with text data.

Keywords: Hyperparameter optimisation · Transformers

1 Introduction

In this paper we compare five state-of-the-art hyperparameter optimisation (HPO) algorithms: Random Search, Bayesian Optimisation (BayesOpt), Asynchronous Successive Halving (ASHA), Bayesian Optimisation with Hyperband (BOHB) and Population-Based Training (PBT). These algorithms are implemented in widely used libraries such as RayTune [13] and so are amongst the most readily available and commonly used HPO methods. We evaluate their performance for hyperparameter tuning of a transformer model trained on text data.

To the best of our knowledge, the present study is the first apples-to-apples comparison of these five HPO methods, and also the first HPO comparison study using transformer neural nets with text data³.

Almost all machine learning models have so-called hyperparameters whose values must be selected in order to train and use the model. Hyperparameters include stochastic gradient descent (SGD) parameters that affect model training, e.g. learning rate, momentum, weight decay, batch size, and also model structural parameters e.g. number of transformer heads and layers. It is often computationally expensive to carry out multiple model training runs with different combinations of hyperparameter values, and so the need is for HPO methods that efficiently (i.e. with few training runs) find a reasonably good choice of hyperparameter values. Often, a mixture of experience, heuristics and trial and

³ Previous HPO comparison studies mostly predate transformers and so study e.g. convolutional networks with image data.

error is used to select the hyperparameter values. Nevertheless, since the choice of hyperparameters can have a substantial impact on model performance, there is also great interest in automating the process of hyperparameter selection.

Given a machine learning model/algorithm A_λ with a number of hyperparameters $\lambda \in \Lambda = \prod_{n=1}^N \Lambda_n$ (the domains Λ_n can be continuous, integer, binary, or categorical, and could also potentially have a conditional structure), the hyperparameter optimisation (HPO) task is to solve [6]:

$$\lambda^* \in \arg \min_{\lambda \in \Lambda} \mathbb{E}_{(D_{\text{train}}, D_{\text{valid}}) \sim \mathcal{S}(D)} [V(L, A_\lambda, D_{\text{train}}, D_{\text{valid}})].$$

where $V(L, A_\lambda, D_{\text{train}}, D_{\text{valid}})$ is a method of evaluating performance that trains A_λ on dataset D_{train} and then evaluates the loss function L on dataset D_{valid} , with $\mathcal{S}(D)$ the distribution of $(D_{\text{train}}, D_{\text{valid}})$ over dataset D . That is, the goal of hyperparameter optimisation is to find a configuration λ^* within the given search space Λ , that minimises the loss expected by the learning model/algorithm when evaluated on unseen data.

Due to the lack of published studies, practitioners have little guidance as to which HPO methods are preferable for transformer models with text data, especially as the hyperparameter search space increases. The present study is a first step towards addressing this gap. It compares popular HPO methods under fair and reproducible conditions.

While it is not possible to draw definitive conclusions from a single study, our measurements indicate that: (i) Random search is consistently slower than the other algorithms, (ii) early stopping of model training runs is important for efficient search, and (iii) ASHA, BayesOpt with ASHA and PBT exhibit competitive performance and so are probably a good starting point for practitioners seeking to use HPO algorithms with transformer models.

2 Related Work

2.1 Hyperparameter Optimisation Algorithms

Many automated optimisation methods have been developed that target the HPO task. We briefly summarise the main methods here.

Grid search is a classical automated HPO method [3, 20]. Each hyperparameter is given a specific range, which is then uniformly divided into sample points. The model performance is evaluated for all combinations of hyperparameters on the resulting grid. Unfortunately, the computational burden scales combinatorially with the number of hyperparameters, making grid search impractical for all but a small number of hyperparameters.

Random search [3] repeatedly samples from the set of possible combinations of hyperparameter values, keeping note of the best performing combination. It was proposed as a practical alternative to grid search by [3].

Bayesian Optimisation (BayesOpt) [5, 19, 23] samples from the set of possible combinations of hyperparameter values and evaluates their performance. It fits a function approximation to the measured performance values. It uses

this model to pick a new combination of hyperparameter values to test, e.g. by selecting a combination where the model prediction is most uncertain, updates the model and repeats. The main choices to be made are (i) the type of model to fit, typically using either a kernel model with a gaussian or Matern kernel or a Tree-structured Parzen Estimator (TPE) [2] although other alternatives have been proposed e.g. FABOLAS [9], SMAC [14], Spearmint [21] and (ii) the acquisition strategy used to select the next hyperparameter values to test and which must balance exploration against exploitation. Common acquisition strategies include upper confidence bound (UCB), expected improvement (EI), probability of improvement (POI).

Population-Based Training (PBT) [10], jointly optimises model weights and hyperparameters by keeping a population of N models with different model weights and hyperparameters. PBT iterates over the models, updating the model weights by gradient descent (asynchronously and in parallel) but also periodically updating the model weights and hyperparameters by borrowing these values from models in the population that are observed to perform well.

The Successive Halving Algorithm (SHA) [8] aims to allocate more resources to promising trials and by stopping poorly performing trials early. It starts with an active set of n combinations of hyperparameter values to evaluate. Models are trained with these in parallel for a specified period r , then the results sorted by performance achieved. The bottom 50% are pruned from the active set, and the process repeats. While rarely used itself for HPO, it is used as the foundation for a number of HPO methods, including Hyperband, ASHA and BOHB.

Hyperband [12] runs multiple copies of SHA in parallel each with a different combination of (n, r) to hedge against a poor choice of these values. For each copy of SHA the initial active set of n combinations of hyperparameter values is selected randomly.

Asynchronous Successive Halving (ASHA) was introduced by [11] and is an asynchronous parallelisation of Hyperband/SHA. ASHA modifies SHA to avoid synchronisation, with pruning taking place asynchronously.

Bayesian Optimisation Hyperband (BOHB) [4] extends Hyperband’s random choice of hyperparameter values with Bayesian optimisation, switching between Bayesian-based exploration and random search.

Other HPO methods proposed in the literature include CMA-ES [16] and Particle Swarm Optimisation (PSO) [15].

2.2 Evaluation of HPO Algorithms

Table 1 summarises the previous studies that compare the performance of HPO algorithms. Of these, [4, 7, 11, 12, 18, 23] evaluate the HPO algorithms on neural nets. However, these studies pre-date transformers and so are mostly confined to MLPs and convolutional neural nets on image data. We identified only one paper that evaluates HPO with transformer neural nets, namely [7]. However, this paper contains no detailed results on methods other than PBT, making direct comparison impossible.

Papers	HPO Algorithms Evaluated
[3, 22]	Random search, Grid search
[23]	Grid search, BayesOpt
[24]	Grid search, Random search, BayesOpt, Hyperband, BOHB, PSO
[18]	Random Search, BayesOpt, BOHB, Hyperband, CMA-ES
[12]	Random search, BayesOpt, Hyperband
[4]	Random Search, BayesOpt, Hyperband, BOHB
[11]	Random Search, Hyperband, BOHB, PBT, ASHA, SHA

Table 1: Previous studies comparing the performance of HPO algorithms

3 Experimental Setup

3.1 Hardware & Software

Performance evaluations were carried out on a server equipped with four RTX 4090 (24GB) GPUs, EPYC 9354P CPU (32 Cores/64 Threads) and 188GB RAM. The RayTune library [13] was used as it is widely used and implements all the algorithms that we investigate. All software used can be found in the GitHub repository https://github.com/Paddyjh/How_Best_To_Optimise_HP_ML.

3.2 Model and Dataset

We use a GPT decoder-only transformer model. The model inputs are token indices, each token is converted to an embedding vector and an absolute positional embedding is added element-wise. The current sequence of tokens is processed by n_layer transformer blocks. Each transformer block first applies a LayerNorm, followed by multi-head self-attention using a causal mask and then an additional LayerNorm with a position-wise MLP. Residual connections wrap around the sublayers. Dropout is applied to the attention weights, the output projection of the multi-head attention and the last linear layer of the neural network.

The loss function used is the mean token-level cross-entropy loss. During training, the model takes in a set of inputs and targets. For each position, the target token represents the next token in the sequence, and the loss is calculated by comparing the predicted distribution over the vocabulary to this target. AdamW [17] is used for training.

The default values of the model hyperparameters are as follows: batch size 64, block size 256, embedding size 256, number of heads 4, number of layers n_layer 4, dropout rate 0.2, AdamW learning rate α 10^{-3} , β_1 0.9, β_2 0.999, weight decay 0.01, ϵ 10^{-8} .

The dataset used was the Tiny Shakespeare dataset [1], which comprises roughly 40,000 lines and 1,000,000 characters of text taken from various works of Shakespeare. The data is split into training and validation datasets, with 90% of the data is used for training and 10% being held out to evaluate the model.

Small number of hyperparameters	Larger number of hyperparameters
AdamW α $[10^{-6}-0.1]$, β_1 $[0.1-0.999]$, β_2 $[0.1-0.9999]$, weight decay $[10^{-8}-0.01]$, ϵ $[10^{-10}, 10^{-3}]$	AdamW α $[10^{-6}-0.1]$, β_1 $[0.1-0.999]$, β_2 $[0.1-0.9999]$, weight decay $[10^{-8}-0.01]$, ϵ $[10^{-10}, 10^{-3}]$, transformer block size $[64 - 384]$, batch size $[16 - 96]$, number of layers $[1 - 6]$, number of heads $[1 - 6]$, embedding size, dropout $[0.1 - 0.5]$

Table 2: Names and value ranges of hyperparameters studied

3.3 Hyperparameters

We evaluate HPO performance for two sets of hyperparameters. The first set is relatively small, with five AdamW hyperparameters. The second is larger, with the five AdamW hyperparameters and also six model hyperparameters. See Table 2 for details, including the range of values explored for each hyperparameter. These ranges were determined by the memory limits of the RTX4090 cards used.

3.4 Training Budget

Model training was limited to a maximum of 10,000 iterations. To determine this value, the model was trained using default hyperparameter values and was found to reach a minimum validation loss at about iteration 5000, and this was then doubled in order to allow for hyperparameter configurations that may take longer to converge. Each HPO algorithm run for the tests with a small number of hyperparameters were limited to a time budget of 40 minutes. This budget was decided after running all HPO methods and observing that none showed a decrease in validation score after roughly 35 minutes. For the larger number of hyperparameters tests, the HPO time budget was doubled to 80 minutes.

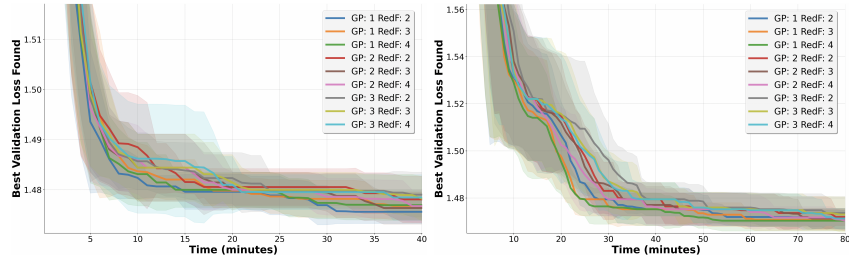
A trial refers to a single training run on the model, with a given hyperparameter configuration. Each HPO algorithm was allowed unlimited trial generation but the maximum number of trials allowed to run in parallel is capped at four, with each trial allocated 1 GPU and 16 CPU cores (the hardware had 4 GPUs and 64 CPU cores) so that each was allocated to one of the four RTX 4090 GPUs and had an equal CPU share.

3.5 Performance Metrics

HPO algorithm performance is evaluated using (i) the mean and standard deviation validation loss vs wall clock time, (ii) snapshots of the standard deviation of mean validation loss at the first, second and third $\frac{1}{3}$ of the HPO test run time (referred to as the early, middle and later stages of the optimisation), (iii) the distribution of number of trials and of training iterations allocated per trial (allowing the impact of early termination of trials, e.g. by ASHA, to be assessed).

HPO	Configuration Parameters
ASHA	Grace Period (GP), Reduction Factor (RF)
Bayesopt	Kappa (K), Kind (KD), Xi
BOHB	Bandwidth Factor (BF), Number of Samples (NS), Random Fraction (RandF), Reduction Factor (RedF), Top N Percent (TNP)
PBT	Burn-in (BI), Perturbation Factor (PF), Perturbations (PT), Quantile Factor (QF), Resampling Probability (RP)

Table 3: Summary of HPO algorithm configuration parameters



(a) Small number of hyperparameters (b) Larger number of hyperparameters

Fig. 1: ASHA configuration tests

4 Sensitivity of Performance on HPO Configuration

With the exception of Random Search, all of the hyperparameter optimisation algorithms studied themselves have parameters that need to be selected before an algorithm can be used, see Table 3. In this section we briefly evaluate the impact of the choice of algorithm parameters (which we refer to as an algorithm *configuration*) on performance. Since it is impractical to exhaustively evaluate all combinations of parameter values, for each algorithm we select nine configurations that span a broad range of parameter values. As we will see, we find that performance is relatively insensitive to the configuration and so more extensive evaluation does not seem warranted.

Figure 1 plots the mean and standard deviation of the best-so-far validation loss against wall clock time of ASHA for various configurations and averaged across 10 seeds. It can be seen that the choice of configuration appears to have a limited effect on the convergence rate and final validation loss achieved by the algorithm (the standard deviation intervals largely overlap for all configurations). That said, aggressive early stopping appears to be preferable, with all Grace Period = 1 configurations being in the top 50% of results. Additionally, a higher Reduction Factor (RF) appeared to be preferable with a larger number of hyperparameters, as an RF of 4 appears in the top 2 configurations and an RF of 3 appears in the third-best configuration.

Figure 2 shows the performance of the BayesOpt algorithm for various configurations and averaged across 10 seeds. The Kappa parameter value is only used when Kind=UCB while parameter Xi is only used when Kind equals POI or EI. It can be seen that the configuration parameters have minimal effect on the optimiser’s performance when there is a small number of hyperparameters,

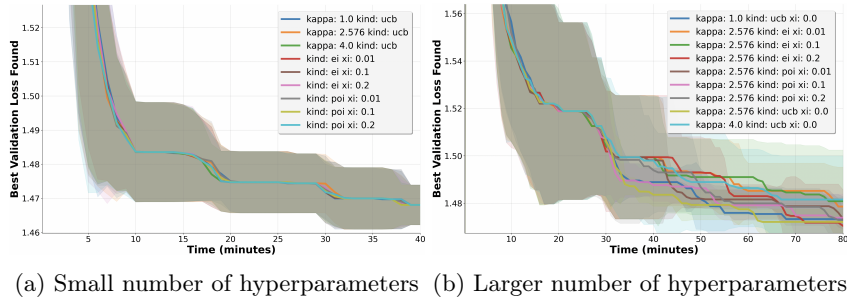


Fig. 2: BayesOpt configuration

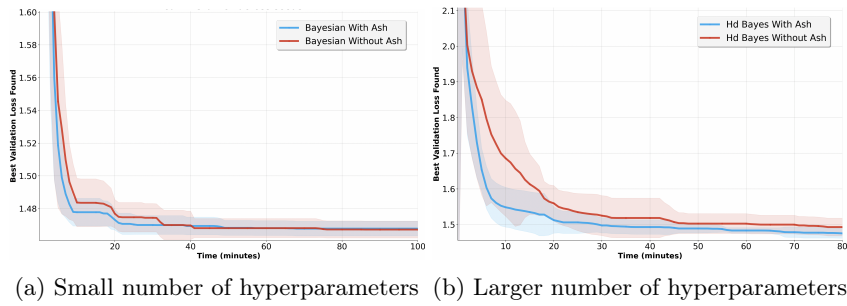


Fig. 3: BayesOpt with and without ASHA

but have a more substantial effect for a larger number of hyperparameters. The UCB acquisition function with $\text{Kappa}=4$ and the EI acquisition function with $\text{Xi}=0.1$ or $\text{Xi}=0.01$ have relatively poor performance. An EI activation function with $\text{Xi}=0.2$ yields the lowest mean final validation loss although UCB with $\text{Kappa}=1$ or 2.576 or POI with $\text{Xi}=0.01$, 0.1 or 0.2 also perform similarly.

Figure 3 shows the performance of BayesOpt using ASHA and the standard implementation without ASHA. It can be seen that using ASHA consistently yields faster convergence and so focus on BayesOpt with ASHA in our evaluation below. Inspection of the number of trials (model training runs) carried out shows that BayesOpt with ASHA performs roughly $5\times$ more trials than vanilla BayesOpt, presumably due to the early termination of less promising trials by ASHA.

Figure 4 shows the performance of the BOHB algorithm for various configurations and averaged across 10 seeds. It can be seen that the standard deviation intervals largely overlap for almost all configurations and so the choice of configuration appears to have a limited effect on the convergence rate. The notable exception in Figure 4(a) is with $\text{BF}=2$, $\text{NS}=32$, $\text{RandF}=0.5$, $\text{RedF}=4$, $\text{TNP}=20$ which shows high standard deviation in the initial stages.

Figure 5 shows the performance of the PBT algorithm. It can be seen from Figure 5(a) that the choice of configuration parameters affects the standard deviation in the initial stages with a small number hyperparameters, but the

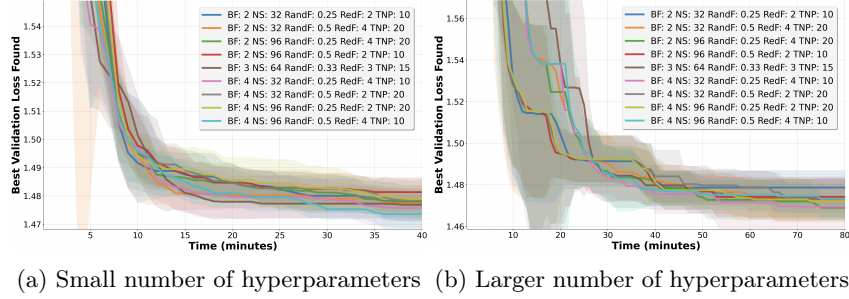


Fig. 4: BOHB configuration tests

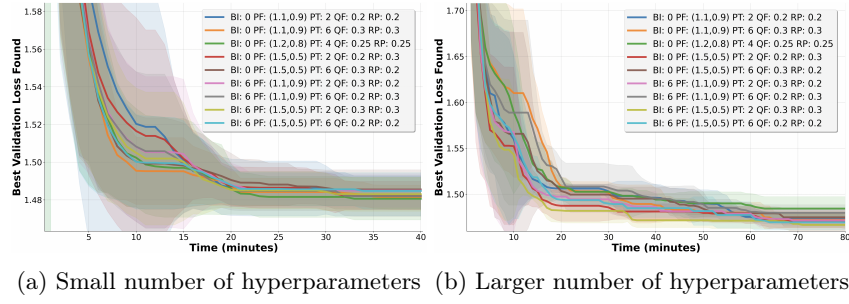


Fig. 5: Population-Based configuration tests

mean validation loss is much the same for all choices of configuration parameters. In Figure 5(b) the choice of configuration parameters has quite a strong effect on initial convergence of the mean validation loss (but little effect on the standard deviation), but a smaller effect on the final mean validation loss.

5 Comparison of Best HPO Method Configurations

In this section we compare the performance of five HPO methods (Random, ASHA, Bayesopt with ASHA, BOHB, PBT). The best configuration found for each method from the previous section is used, see Table 4, and each method is run across 10 seeds and evaluated on both a low and a high-dimensional hyperparameter search space. Performance is evaluated using (i) the mean and

HPO	Configuration Parameters Values	
	Small #hyperparameters	Large #hyperparameters
ASHA	GP=1, RF=2	GP=1, RF=4
Bayesopt	K=4, KD=UCB	Xi=0.2, KD=EI
BOHB	BF=4, NS=96, RandF=0.5, RedF=4, TNP=10	BF=2, NS=96, RandF=0.25, RedF=4, TNP=20
PBT	BI=0, PF=(1.2, 0.8), PT=4, QF=0.25, RP=0.25	BI=6, PF=(1.5, 0.5), PT=2, QF=0.3, RP=0.30

Table 4: HPO algorithm selected configuration parameters

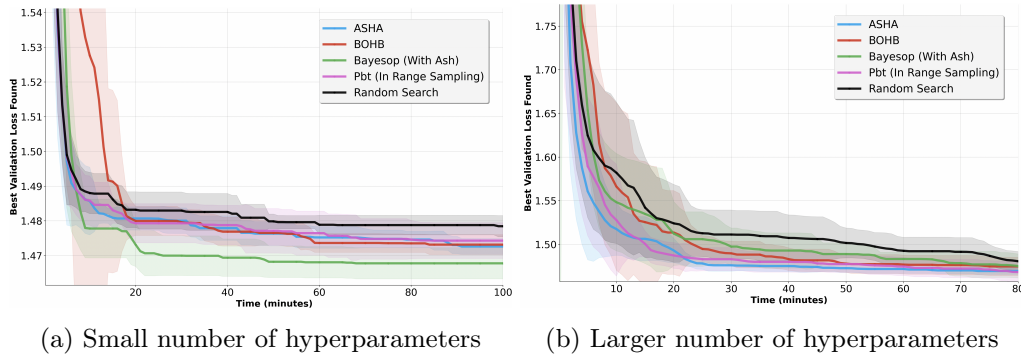


Fig. 6: Measured mean and standard deviation validation loss vs real-world wall clock time for the hyperparameter optimisation algorithms studied

standard deviation validation loss against real-world wall clock time, (ii) standard deviation of mean validation loss at early, middle and later stages of the optimisation, and (iii) the distribution of number of trials per optimiser and of training iterations allocated per trial.

Figure 6 plots the measured mean and standard deviation of the best-so-far validation loss against real-world wall clock time. Ideally, we would like the validation loss to quickly decrease to a low value, and to do this consistently i.e. with low standard deviation. We can make the following observations:

1. All of the HPO algorithms studied are successful at eventually finding a set of hyperparameter values that yield low validation loss. The algorithms have similar levels of standard deviation in performance, with the exception of BOHB which has somewhat greater standard deviation than the other algorithms, especially in the early stages.
2. Random Search is consistently slower than the other algorithms at finding hyperparameter values yielding a low validation loss. This is perhaps unsurprising given its simplicity, but it is notable because previous studies have suggested that it is a superior approach.
3. The number of hyperparameters affects which HPO algorithm performs best. It can be seen in Figure 6(a) that BayesOpt with ASHA achieves the lowest loss of all algorithms very rapidly, and continues to outperform the other optimisers until completion. In Figure 6(b), ASHA and PBT show similar performance and dominate the other algorithms.
4. Early stopping of training runs appears to be important for achieving competitive performance. BayesOpt with ASHA dominates the other algorithms in Figure 6(a), while ASHA shows good performance in Figure 6(b). That said, it is worth noting that PBT is competitive with ASHA in Figure 6(b). PBT allows trials to run for longer but can change configurations mid-trial.

To gain more insight into the data shown in Figure 6, Figure 7 summarises the validation loss standard deviation at early, middle and late stages of the

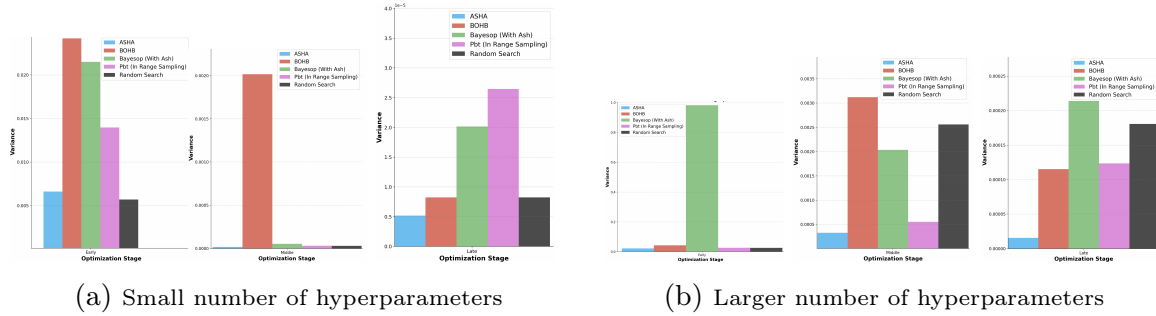


Fig. 7: Optimiser variance at Early, Middle, Late stage of optimisation

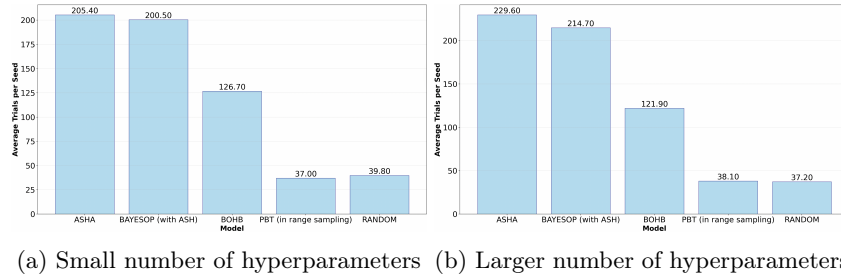


Fig. 8: Trials allocated per optimiser

optimisation. It can be seen that ASHA consistently exhibits a relatively low standard deviation at all stages and for both small and larger numbers of hyperparameters. BayesOpt with ASHA generally exhibits a relatively high standard deviation i.e. the quality of the hyperparameters found is generally more variable than for the other algorithms.

Drilling down further, Figure 8 shows the number of trials (training runs) carried out by each algorithm, while Figure 9 shows the duration of the training runs. It can be seen from Figure 8 that PBT and Random Search sampled from a far smaller number of trials than ASHA, Bayes Opt with ASHA and BOHB. Consistent with this, Figure 9 shows that ASHA, Bayes Opt with ASHA and BOHB tend to terminate training runs earlier than PBT and Random Search. That is, the aggressive early termination of trials by ASHA, Bayes Opt with ASHA and BOHB allows these algorithms to explore a larger search space. While a larger number of trials can be expected to help performance, judicious choice of trials is also important since although BayesOpt with ASHA and BOHB have a higher number of trials than PBT, Figure 6 shows that PBT tends to match or outperform these algorithms.

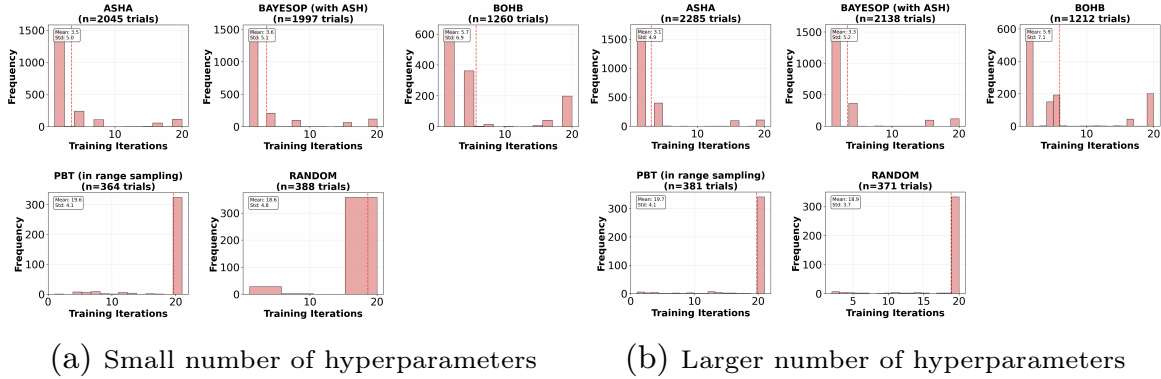


Fig. 9: Iterations per trial for each optimiser

6 Summary and Conclusions

We present a direct performance comparison of five state-of-the-art hyperparameter optimisation algorithms: Random Search, Bayesian Optimisation, Asynchronous Successive Halving, Bayesian Optimisation with Hyperband and Population-Based Training. While it is not possible to draw definitive conclusions from a single study, our measurements indicate that: (i) Random search is consistently slower than the other algorithms, (ii) early stopping of model training runs is important for efficient search, and (iii) ASHA, BayesOpt with ASHA and PBT exhibit competitive performance and so are probably a good starting point for practitioners seeking to use HPO algorithms with Transformer models.

References

1. Tiny shakespeare dataset (2023), https://huggingface.co/datasets/karpathy/tiny_shakespeare
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Proceedings NIPS. pp. 2546–2554 (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (Feb 2012)
4. Falkner, S., Klein, A., Hutter, F.: BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In: Proc ICML. pp. 1437–1446 (2018)
5. Galuzzi, B.G., Giordani, I., Candelieri, A., Perego, R., Archetti, F.: Hyperparameter optimization for recommender systems through Bayesian optimization. Computational Management Science **17**(4), 495–515 (2020)
6. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning: Methods, Systems, Challenges (2019)
7. Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., Kavukcuoglu, K.: Population Based Training of Neural Networks (2017), arXiv:1711.09846 [cs]
8. Karnin, Z., Koren, T., Somekh, O.: Almost Optimal Exploration in Multi-Armed Bandits. In: Proceedings ICML. vol. 28, pp. 1238–1246 (2013)

9. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Proc AISTATS. pp. 528–536 (2017)
10. Li, A., Spyra, O., Perel, S., Dalibard, V., Jaderberg, M., Gu, C., Budden, D., Harley, T., Gupta, P.: A generalized framework for population based training. In: Proc KDD. p. 1791–1799 (2019)
11. Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-tzur, J., Hardt, M., Recht, B., Talwalkar, A.: A System for Massively Parallel Hyperparameter Tuning. Proc Machine Learning and Systems **2**, 230–246 (2020)
12. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. J. Machine Learn. Res. **18**(185), 1–52 (2018)
13. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A Research Platform for Distributed Model Selection and Training (2018), <http://arxiv.org/abs/1807.05118>
14. Lindauer, M., Eggenberger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: Smac3: A versatile bayesian optimization package for hyperparameter optimization. J. Machine Learn. Res. **23**(54), 1–9 (2022)
15. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proc Genetic and Evolutionary Computation Conf. p. 481–488 (2017)
16. Loshchilov, I., Hutter, F.: Cma-es for hyperparameter optimization of deep neural networks. arXiv preprint arXiv:1604.07269 (2016)
17. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization (2019), <http://arxiv.org/abs/1711.05101>
18. Motz, M., Krauß, J., Schmitt, R.H.: Benchmarking of hyperparameter optimization techniques for machine learning applications in production. Adv. in Ind. & Manufacturing Eng. **5**, 100099 (2022)
19. Nguyen, V.: Bayesian Optimization for Accelerating Hyper-Parameter Tuning. In: Proc AIKE. pp. 302–305 (2019)
20. Shekhar, S., Bansode, A., Salim, A.: A Comparative study of Hyper-Parameter Optimization Tools. In: Proc CSDE. pp. 1–6 (2021)
21. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proc NIPS. p. 2951–2959 (2012)
22. Subaşı, N.: Comprehensive Analysis of Grid and Randomized Search on Dataset Performance. European J. Eng and Applied Sciences **7**(2), 77–83 (2024)
23. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. J. Electronic Science & Tech. **17**(1), 26–40 (2019)
24. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing **415**, 295–316 (2020)