An Empirical Study on Hyperparameter Optimization for Fine-Tuning Pre-trained Language Models

Xueqing Liu
Stevens Institute of Technology
xueqing.liu@stevens.edu

Chi Wang
Microsoft Research
wang.chi@microsoft.com

Abstract

The performance of fine-tuning pre-trained language models largely depends on the hyperparameter configuration. In this paper, we investigate the performance of modern hyperparameter optimization methods (HPO) on fine-tuning pre-trained language models. First, we study and report three HPO algorithms' performances on fine-tuning two state-of-the-art language models on the GLUE dataset. We find that using the same time budget, HPO often fails to outperform grid search due to two reasons: insufficient time budget and overfitting. We propose two general strategies and an experimental procedure to systematically troubleshoot HPO's failure cases. By applying the procedure, we observe that HPO can succeed with more appropriate settings in the search space and time budget; however, in certain cases overfitting remains. Finally, we make suggestions for future work. Our implementation can be found in https://github.com/microsoft/FLAML/tree/main/flaml/nlp/.

1 Introduction

In the recent years, deep learning and pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Clark et al., 2020; He et al., 2021) have achieved great success in the NLP community. It has now become a common practice for researchers and practitioners to fine-tune pre-trained language models in down-stream NLP tasks. For example, the HuggingFace transformers library (Wolf et al., 2020) was ranked No.1 among the most starred NLP libraries on GitHub using Python[1].

Same as other deep learning models, the performance of fine-tuning pre-trained language models largely depends on the hyperparameter configuration. A different setting in the hyperparameters may cause a significant drop in the performance, turning a state-of-the-art model into a poor model. Methods for tuning hyperparameters can be categorized as (1) traditional approaches such as manual tuning and grid search, and (2) automated HPO methods such as random search and Bayesian optimization (BO). Manual tuning often requires a large amount of manual efforts; whereas grid search often suffers from lower efficiency due to the exponential increase in time cost with the number of hyperparameters. Automated HPO methods were proposed to overcome these disadvantages. Recently, automated HPO methods also become increasingly popular in the NLP community (Zhang and Duh, 2020; Dodge et al., 2019). For example, Bayesian optimization (BO) (Zhang and Duh, 2020) and Population-based Training (Jaderberg et al., 2017) both prove to be helpful for improving the performance of the transformer model (Vaswani et al., 2017) for neural machine translation. The HuggingFace library has also added native supports for HPO in a recent update (version 3.1.0, Aug 2020).

With improved supports, users can now easily access a variety of HPO methods and apply them to their fine-tuning tasks. However, the effectiveness of this step is less understood. To bridge this gap, in this paper, we propose an experimental study for fine-tuning pre-trained language models using the HuggingFace library. This study is motivated by the following research questions: First, can automated HPO methods outperform traditional tuning method such as grid search? Second, on which NLP tasks do HPO methods work better? Third, if HPO does not work well, how to troubleshoot the problem and improve its performance?

To answer these questions, we start from a simple initial study (Section 4) by examining the performance of three HPO methods on two state-of-the-art language models on the GLUE dataset. The

[1] https://github.com/EvanLi/Github-Ranking/blob/master/Top100/Python.md

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2286–2300
August 1–6, 2021.

time budget for HPO in the initial study is set to be the same as grid search. Results of the initial study show that HPO often fails to match grid search's performance. The reasons for HPO's failures are two folds: first, the same budget as grid search may be too small for HPO; second, HPO overfits the task. With these observations, we propose two general strategies for troubleshooting the failure cases in HPO as well as an overall experimental procedure (Figure 1). By applying the procedure (Section 5), we find that by controlling overfitting with reduced search space and using a larger time budget, HPO has outperformed grid search in more cases. However, the overfitting problem still exists in certain tasks even when we only search for the learning rate and batch size. Finally, we make suggestions for future work (Section 7).

The main contributions of this work are:

- We empirically study the performance of three HPO methods on two pre-trained language models and on the GLUE benchmark;
- We design an experimental procedure which proves useful to systematically troubleshoot the failures in HPO for fine-tuning;
- We report and analyze the execution results of the experimental procedure, which sheds light on future work;

2 Definition of HPO on Language Model Fine-Tuning

Given a pre-trained language model, a fine-tuning task, and a dataset containing $D_{train}, D_{val}, D_{test}$, the goal of a hyperparameter optimization algorithm is to find a hyperparameter configuration $\mathbf{c}$, so that when being trained under configuration $\mathbf{c}$, the model's performance on a validation set D_{val} is optimized. Formally, the goal of HPO is to find

$$\mathbf{c}^* = \arg\max_{\mathbf{c}\in\mathcal{S}} f(\mathbf{c}, D_{train}, D_{val})$$

where $\mathcal{S}$ is called the *search space* of the HPO algorithm, i.e., the domain where the hyperparameter values can be chosen from. The function $f(\cdot,\cdot,\cdot)$ is called the evaluation protocol of HPO, which is defined by the specific downstream task. For example, many tasks in GLUE define f as the validation accuracy. If a task has multiple protocols, we fix f as one of them[2]. After finding $\mathbf{c}^*$, the performance of HPO will be evaluated using the performance of the model trained with $\mathbf{c}^*$ on the *test* set D_{test}.

To fairly compare the performances of different HPO algorithms, the above optimization problem is defined with a constraint in the maximum running time of the HPO algorithm, which we call the *time budget* for the algorithm, denoted as B. Under budget B, the HPO algorithm can try a number of configurations $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n$. The process of fine-tuning with configuration $\mathbf{c}_i$ is called a *trial*. Finally, we call the process of running an HPO algorithm A once *one HPO run*.

3 Factors of the Study

In this paper, we conduct an empirical study to answer the research questions in Section 1. First, can automated HPO methods outperform grid search? The answer to this question depends on multiple factors, i.e., the NLP task on which HPO and grid search are evaluated, the pre-trained language model for fine tuning, the time budget, the search space for grid search and HPO algorithm, and the choice of HPO algorithm. To provide a comprehensive answer, we need to enumerate multiple settings for these factors. However, it is infeasible to enumerate all possible settings for each factor. For instance, there exist unlimited choices for the search space. To accomplish our research within reasonable computational resources[3], for each factor, we only explore the most straight-foward settings. For example, the search space for grid search is set as the default grid configuration recommended for fine-tuning (Table 1), and the search space for HPO is set as a straightforward relaxation of the grid configuration. We explain the settings for each factor in details below.

[2]There are 3 GLUE tasks with multiple validation scores: MRPC, STS-B, and QQP (not studied). For MRPC we optimize the validation accuracy, and for STS-B we optimize the Pearson score on the validation set.

[3]Our experiments were run on two GPU servers, server 1 is equipped with 4xV100 GPUs (32GB), and server 2 is a DGX server equipped with 8xV100 GPUs (16GB). To avoid incomparable comparisons, all experiments on QNLI and MNLI are run exclusively on server 2, and all other experiments are run exclusively on server 1. To speed up the training, we use fp16 in all our experiments. To guarantee the comparability between different HPO methods, all trials are allocated exactly 1 GPU and 1 CPU. As a result, all trials are executed in the single-GPU mode and there never exist two trials sharing the same GPU.

Hyperparameter	Electra-grid	Electra-HPO	RoBERTa-grid	RoBERTa-HPO
learning rate	{3e-5,1e-4,1.5e-4}	log((2.99e-5,1.51e-4))	{1e-5,2e-5,3e-5}	(0.99e-5,3.01e-5)
warmup ratio	0.1	(0, 0.2)	0.06	(0, 0.12)
attention dropout	0.1	(0, 0.2)	0.1	(0, 0.2)
hidden dropout	0.1	(0, 0.2)	0.1	(0, 0.2)
weight decay	0	(0, 0.3)	0.1	(0, 0.3)
batch size	32	{16, 32, 64}	{16, 32}	{16, 32, 64}
epochs	10 for RTE/STS-B, 3 for other	10 for RTE/STS-B, 3 for other	10	10

Table 1: The search space for grid search and HPO methods in this paper. For grid search, we adopt the search spaces from the Electra (Clark et al., 2020) and RoBERTa (Liu et al., 2019) paper. For each model, we expand the grid search space to a larger, simple search space for HPO.

NLP Tasks. To study HPO's performance on multiple NLP tasks, we use the 9 tasks from the GLUE (General Language Understanding Evaluation) benchmark (Wang et al., 2018).

Time Budget. We focus on a low-resource scenario in this paper. To compare the performance of grid search vs. HPO, we first allocate the same time budget to HPO as grid search in our initial comparative study (Section 4). If HPO does not outperform grid search, we increase the time budget for HPO. We require that each HPO run takes no more than 8 GPU hours with the NVIDIA Tesla V100 GPU under our setting. We prune a task if the time for grid search exceeds two hours. A complete list of the time used for each remaining task can be found in Table 2.

NLP task	Electra	epoch	RoBERTa	epoch
WNLI	420	3	660	10
RTE	1000	10	720	10
MRPC	420	3	720	10
CoLA	420	3	1200	10
STS-B	1200	10	1000	10
SST	1200	3	**7800**	-
QNLI	1800	3	-	
QQP	**7800**	3	-	-
MNLI	6600	3	-	-

Table 2: The running time of grid search for each task (in seconds) and the corresponding number of epochs.

Pre-trained Language Models. In this paper, we focus on two pre-trained language models: the Electra-base model (Clark et al., 2020), and the RoBERTa-base model (Liu et al., 2019). Electra and RoBERTa are among the best-performing models on the leaderboard of GLUE as of Jan 2021[4]. Another reason for choosing the two models is that they both provide a simple search space for grid search, and we find it helpful to design our HPO search space on top of them. We use both models' implementations from the transformers library (Wolf et al., 2020) (version = 3.4.0). Among all the different sizes of RoBERTa and Electra (large, base, small), we choose the base size, because large models do not fit into our 2-hour budget[5]. With the 2-hour time constraint, we prune tasks where grid search takes longer than two hours. For Electra, QQP is pruned, whereas for RoBERTa, SST, QNLI, QQP, MNLI are pruned.

Search Space for Grid Search and HPO. It is generally difficult to design an HPO search space from scratch. In our problem, this difficulty is further amplified with the limited computational resources. Fortunately, most papers on pre-trained language models recommend one or a few hyperparameter configurations for fine-tuning. We use them as the configurations for grid search. For HPO, the performance depends on the search space choice, e.g., it takes more resources to explore a large space than a smaller space close to the best configuration. Due to the time budget limits, we focus on a small space surrounding the recommended grid search space, as shown

[4] www.gluebenchmark.com

[5] Our empirical observation shows that the large models take 1.5 to 2 times the running time of the base models.

in Table 1.[6] More specifically, we convert the *learning rate*, *warmup ratio*, *attention dropout*, and *hidden dropout* to a continuous space by expanding the grid space. For *weight decay*, since the recommended configuration is 0, we follow Ray Tune's search space and set the HPO space to (0, 0.3) (Kamsetty, 2020). For *epoch number*, most existing work uses an integer value between 3 and 10 (Clark et al., 2020; Liu et al., 2019; Dai et al., 2020), resulting in a large range of space we can possibly search. To reduce the exploration required for HPO, we skip expanding the search space for epoch number and fix it to the grid configuration.

HPO Algorithms. We compare the performance between grid search and three HPO algorithms: random search (Bergstra and Bengio, 2012), asynchronous successive halving (ASHA) (Li et al., 2020), and Bayesian Optimization (Akiba et al., 2019)+ASHA. We use all HPO methods' implementations from the Ray Tune library (Liaw et al., 2018) (version 1.2.0). We use BO (with TPE sampler) together with the ASHA pruner, because with the small time budget, BO without the pruner reduces to random search. As fine-tuning in NLP usually outputs the checkpoint with the *best* validation accuracy, we also let the HPO methods output the *best* checkpoint of the best trial. This choice is explained in more details in Appendix A.1.

4 Experiment #1: Comparative Study using 1GST

As the performance of HPO depends on the time budget, to compare between grid search and HPO, we first conduct an initial study by setting the time budget of HPO to the same as grid search. For the rest of this paper, we use aGST to denote that the time budget=$a\times$the running time for grid search. Table 3 shows the experimental results on Electra and RoBERTa using 1GST. For each (HPO method, NLP task) pair, we repeat the randomized experiments 3 times and report the average scores. We analyze the results in Section 4.1.

[6]The grid search spaces in Table 1 are from Table 7 of Electra and Table 10 of RoBERTa. For Electra, we fix the hyperparameters for Adam; we skip the layer-wise learning rate decay because it is not supported by the HuggingFace library. While Electra's original search space for learning rate is [3e-5, 5e-5, 1e-4, 1.5e-4], we have skipped the learning rate 5e-5 in our experiment.

4.1 Analysis of the Initial Results

Electra. By comparing the performance of grid search and HPO in Table 3 we can make the following findings. First, HPO fails to match grid search's validation accuracy in the following tasks: RTE, STS-B, SST and QNLI. In certain tasks such as QNLI and RTE, grid search outperforms HPO by a large margin. Considering the fact that grid search space is a subspace of the HPO space, this result shows that with the same time budget as grid search (i.e., approximately 3 to 4 trials), it is difficult to find a configuration which works better than the recommended configurations. Indeed, with 3 to 4 trials, it is difficult to explore the search space. Although ASHA and BO+ASHA both search for more trials by leveraging early stopping (Li et al., 2020), the trial numbers are still limited (the average trial numbers for experiments in Table 3 can be found in Table 6 of the appendix). Second, among the tasks where HPO outperforms grid search's validation accuracy, there are 2 tasks (WNLI, MRPC) where the test accuracy of HPO is lower than grid search. As a result, the HPO algorithm overfits the validation dataset. Overfitting in HPO generally happens when the accuracy is optimized on a limited number of validation data points and cannot generalize to unseen test data (Feurer and Hutter, 2019). (Zhang et al., 2021) also found that fine-tuning pre-trained language models is prone to overfitting when the number of trials is large, though they do not compare HPO and grid search. Finally, by searching for more trials, ASHA and BO+ASHA slightly outperform random search in the validation accuracy, but their test accuracy is often outperformed by random search.

RoBERTa. By observing RoBERTa's results from Table 3, we can see that the average validation accuracy of HPO outperforms grid search in all tasks except for CoLA. It may look like HPO is more effective; however, most of the individual runs in Table 3 overfit. As a result, HPO for fine-tuning RoBERTa is also prone to overfitting compared with grid search. The complete lists of the overfitting cases in Table 3 can be found in Table 8 and Table 9 of Appendix A.3.

4.2 A General Experimental Procedure for Troubleshooting HPO Failures

Since Table 3 shows HPO cannot outperform grid search using 1GST, and is prone to overfitting, we propose two general strategies to improve HPO's

	WNLI	RTE	MRPC	CoLA	STS-B	SST	QNLI	MNLI
Electra-base, validation								
grid	56.3	**84.1**	92.3/89.2	67.2	**91.5/91.4**	**95.1**	**93.5**	88.6
RS	56.8	82.2	**93.0/90.4**	68.8	90.1/90.2	94.7	93.0	88.9
RS+ASHA	57.2	80.3	93.0/90.3	67.9	91.4/91.3	94.9	93.1	88.6
BO+ASHA	**58.2**	82.6	**93.1/90.4**	**69.4**	91.5/91.3	94.7	93.1	**89.2**
Electra-base, test								
grid	**65.1**	**76.8**	**91.1/87.9**	58.5	**89.7/89.2**	**95.7**	**93.5**	88.3
RS	64.4	75.6	90.7/87.5	63.0	88.0/87.6	95.1	93.0	**88.7**
RS+ASHA	62.6	74.1	90.6/87.3	61.2	89.5/89.1	94.9	92.9	88.5
BO+ASHA	61.6	75.1	90.7/87.4	**64.1**	89.7/89.1	94.8	93.0	**88.7**

	WNLI	RTE	MRPC	CoLA	STS-B
RoBERTa-base, validation					
grid	56.3	79.8	93.1/90.4	**65.1**	91.2/90.8
RS	**57.8**	80.4	93.3/90.7	64.1	91.2/90.9
RS+ASHA	57.3	**80.8**	93.4/90.8	64.5	91.2/90.9
BO+ASHA	56.3	80.3	**93.7/91.4**	64.5	**91.3/91.0**
RoBERTa-base, test					
grid	**65.1**	73.9	90.5/87.1	**61.7**	89.3/88.4
RS	64.9	73.5	90.1/86.7	59.1	**89.3/88.6**
RS+ASHA	**65.1**	**74.1**	**90.6/87.3**	59.4	89.1/88.3
BO+ASHA	**65.1**	73.3	90.4/87.2	60.1	89.1/88.4

Table 3: Results of the initial comparative study on Electra (top) and RoBERTa (bottom) by varying the GLUE task and HPO method while fixing the search space and time budget. For each (HPO method, task), we rerun the experiment 3 times and report the average.

performance. First, we increase the time budget for HPO so that HPO can exploit the space with more trials. Second, to control overfitting, we propose to reduce the search space. More specifically, we propose to fix the values of certain hyperparameters to the default values in the grid configuration (Table 3). The reason is that overfitting can be related to certain hyperparameter settings of the model. For example, it was shown in ULMFit (Howard and Ruder, 2018) that using a non-zero warmup step number can help reduce overfitting. Intuitively, a larger search space is more prone to overfitting. For example, by using a warmup search space = (0, 0.2), the warmup steps in the best trial found by HPO may be much smaller or larger than the steps used by grid search. Other hyperparameters which are related to overfitting of fine-tuning include the learning rate (Smith and Le, 2017), batch size (Smith et al., 2017), and the dropout rates (Srivastava et al., 2014; Loshchilov and Hutter, 2019, 2018).

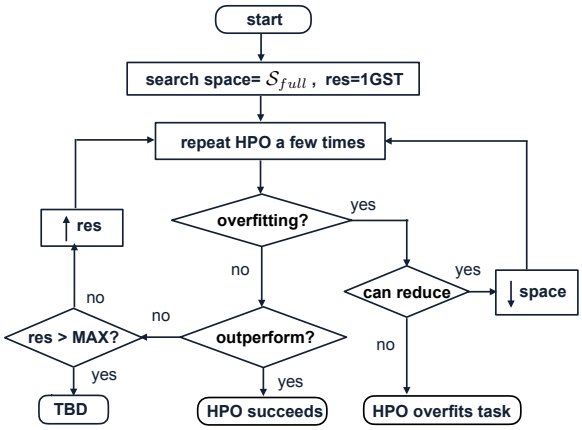
Our proposed procedure for troubleshooting HPO failures is depicted in Figure 1. Starting from the full search space and 1GST, we test the HPO algorithm for a few times. If any overfitting is observed, we reduce the search space and go back to testing the HPO algorithm again. On the other hand, if no overfitting is observed and HPO also does not outperform grid search, we increase the time budget and also go back to testing the HPO algorithm again. We continue this procedure until any of the following conditions is met: first, HPO successfully outperforms grid search; second, the search space cannot be further reduced, thus HPO overfits the task; third, the time budget cannot be further increased under a user-specified threshold, thus whether HPO can outperform grid search is to be determined for this specific task.

5 Experiment #2: Troubleshooting HPO

In this section, we evaluate the effectiveness of our proposed procedure in Figure 1. To apply the procedure, we need to further consolidate two components: first, what time budget should we use; second, *which hyperparameter* to fix for reducing the search space. For the first component, we use a

relatively small list for time budget options {1GST, 4GST}. For the second component, it is difficult to guarantee to reduce overfitting by fixing a specific hyperparameter to its grid search values. When choosing the hyperparameter to fix, we refer to the configurations of the best trials which cause the HPO results to overfit.

Figure 1: A general experimental procedure for troubleshooting HPO failure cases.



5.1 Choosing the Hyperparameter to Fix

Electra. To decide which hyperparameter to fix, we examine the best trial's configuration for the overfitting HPO runs (compared with the grid search performance). If there is a pattern in a certain hyperparameter of all these configurations (e.g., warmup ratio below 0.1 for Electra), by fixing such hyperparameters to the values of grid search, we can exclude the other values which may be related to overfitting. We apply this analytical strategy to the initial Electra results in Table 3. Among the 72 runs, 9 runs overfit compared with grid search. For each run, we list the hyperparameter configurations of the best trial in Table 8 of Appendix A.3. For Electra, we have skipped showing weight decay in Table 8, because the HPO configuration is never smaller than the grid configuration, thus does not affect the result of the analysis. For comparative purpose, we also list the hyperparameter values of the best trial in grid search. To improve the readability of Table 8, we use 4 different colors (defined in Appendix A.3) to denote the comparison between values of the best trial in HPO and values of the best trial in grid search.

From Table 8, we observe that the *warmup ratio*s are often significantly lower than 0.1. We skip the analysis on learning rate because its search space (log((2.99e-5,1.51e-4))) cannot be further reduced without losing coverage of the grid configurations or continuity; we also skip weight decay because any trial's value cannot be smaller than 0. Following this empirical observation, we hypothesize that fixing the warmup ratio to 0.1 can help reduce overfitting in Electra. We use $\mathcal{S}_{full}$ to denote the original search space and $\mathcal{S}_{-wr}$ to denote the search space by fixing the warmup ratio to 0.1. If HPO overfits in both $\mathcal{S}_{full}$ and $\mathcal{S}_{-wr}$, the procedure will reduce the search space to the minimal continuous space $\mathcal{S}_{min}$ containing the grid search space, which searches for the learning rate only.

RoBERTa. We apply the same analytical strategy to the RoBERTa results in Table 3 and show the hyperparameters of the best trials in Table 9. For RoBERTa, we propose to fix the values of two hyperparameters at the same time: the warmup ratio and the hidden dropout. We denote the search space after fixing them as $\mathcal{S}_{-wr-hdo}$. If HPO overfits in both $\mathcal{S}_{full}$ and $\mathcal{S}_{-wr-hdo}$, the procedure will reduce the search space to S_{min} which contains the learning rate and batch size only.

5.2 Execution Results of the Procedure

In this section, we apply the troubleshooting procedure on the initial HPO results from Table 3 and observe the execution paths. In Table 10 and Table 11 of Appendix A.4, we list the full execution results of the procedure for random search and random search + ASHA. Table 10&11 have included only the tasks where the HPO does not succeed in the initial study. In Table 10&11, we show the validation and test accuracy for the three repetitions of HPO runs as well as their average score.

An Example of Executing the Procedure. In Figure 4, we show an example of applying the procedure on random search for Electra on RTE. In round 0, the validation and test accuracies of all three repetitions are lower than grid search. That implies RS needs more time budget, therefore we increase the budget (marked as ↑res) for RS from 1GST to 4GST. After the increase, overfitting is detected in the 1st repetition of round 1 (validation accuracy = 84.5, test accuracy = 74.6). We thus reduce the search space (marked as ↓ space) from $\mathcal{S}_{full}$ to $\mathcal{S}_{-wr}$. In round 2, the 1st repetition still shows (weak) overfitting: RS has the same

	round 0		round 1		round 2		round 3	
	val	test	val	test	val	test	val	test
grid	**84.1**	**76.8**	↑ res		↓ space		↓ space	
rep1	81.9	76.1	84.5	74.6	84.1	76.1	84.8	75.3
rep2	81.6	75.1	83.8	74.5	83.0	74.0	84.1	75.7
rep3	83.0	75.7	83.4	74.7	82.3	73.1	83.8	75.2
Avg	82.2	75.6	83.9	74.6	83.1	74.4	84.2	75.4

Table 4: An example of executing the experimental procedure applied to random search for Electra on RTE. The grid search accuracy is denoted using the **blue bold font**. An HPO run is highlighted in dark grey if it overfits and medium grey if it overfits weakly.

validation accuracy as grid search (84.1), a smaller test accuracy (76.1), and a smaller validation loss (RS's validation loss = 0.8233, grid search's validation loss = 0.9517). We thus continue reducing the search space to $\mathcal{S}_{min}$, and overfitting is detected again in the 1st repetition of round 3 (validation accuracy = 84.8, test accuracy = 75.3). After round 3, the search space cannot be further reduced, so we classify this case as 'HPO overfits task'.

We analyze the execution results in Table 10 and 11 jointly as follows.

Effects of Reducing the Search Space. From the two tables we can observe that reducing the search space can be effective for controlling overfitting. In WNLI (Electra), both algorithms outperform grid search after reducing the search space once. In WNLI (RoBERTa), ASHA outperforms grid search after reducing the search space twice. We can observe a similar trend in MRPC (Electra), SST (Electra), RTE (RoBERTa), and CoLA (RoBERTa). However, for these cases, overfitting still exists even after we reduce the search space twice, i.e., using the minimal search space.

Effects of Increasing the Time Budget. By observing cases of increased budget in Table 10 and 11, we can see that this strategy is generally effective for improving the validation accuracy. After increasing the time budget, in STS-B (Electra) all HPO methods outperform grid search's validation and test accuracy; in SST (Electra-RS) and CoLA (RoBERTa) HPO outperforms grid search in only the validation accuracy. In RTE (Electra) and QNLI (Electra), however, this increase is not enough for bridging the gap with grid search, thus HPO remains behind. For RTE (Electra), SST (Electra), QNLI (Electra), and CoLA (RoBERTa), overfitting happens after increasing the time budget from 1GST to 4GST. After reducing the search space, we still observe overfitting in most cases.

Comparisons between RS and ASHA. By comparing the results between random search and ASHA in Table 10 and 11, we find that before increasing the budget, RS rarely outperforms ASHA in the validation accuracy; however, after the budget of both RS and ASHA increases to 4GST, the best validation accuracy of RS has consistently outperformed ASHA, i.e., in all of RTE (Electra), STS-B (Electra), SST (Electra), and QNLI (Electra). That is, the increase in the time budget has led to more significant (validation) increase in RS than ASHA. This result may be caused by two reasons. First, at 1GST, ASHA already samples a larger number of trials (Appendix A.2), which may be sufficient to cover its search space; on the other hand, RS cannot sample enough trials, thus increasing the time budget is more helpful. Second, ASHA may make mistake by pruning a good trial that shows a bad performance at the beginning.

5.3 Summary of the Main Findings

In Table 5, we list the final execution results for each task in Electra and RoBERTa. Our main findings can be summarized as follows. After increasing the time budget and reducing the search space, HPO outperforms grid search in the following cases: (1) in 3 cases (i.e., CoLA (Electra), STS-B (Electra) and MNLI (Electra)), HPO outperforms grid search by using the full search space, where STS-B needs more budget; (2) in 4 cases (i.e., WNLI (Electra), WNLI (RoBERTa), MRPC (RoBERTa) and STS-B (RoBERTA)), HPO succeeds after reducing the search space; (3) in the other 7 cases, HPO cannot outperform grid search even after increasing the time budget and reducing the search space. This result shows that when searching in a continuous space surrounding the recommended grid configurations, it can be difficult for existing automated HPO methods (e.g., Random Search, ASHA, Bayesian optimization) to outperform grid search (with manually tuned grid configurations recommended by the language model) within a short amount of time; even if we can identify a configuration with good validation score, most likely the test score is still worse than

task	Execution Results
WNLI	All HPO succeed w/ 1GST, $\mathcal{S}_{-wr}$
RTE	RS overfits
	ASHA and BO+ASHA TBD
MRPC	All HPO overfit
CoLA	All HPO succeed w/ 1GST, $\mathcal{S}_{full}$
STS-B	All HPO succeed w/ 4GST, $\mathcal{S}_{full}$
SST	All HPO overfit
QNLI	All HPO TBD
MNLI	All HPO succeed w/ 1GST, $\mathcal{S}_{full}$

task	Execution Results
WNLI	ASHA succeeds* w/ 1GST, $\mathcal{S}_{-wr-hdo}$
	RS and BO+ASHA overfit
RTE	All HPO overfit
MRPC	ASHA succeeds* w/ 1GST, $\mathcal{S}_{-wr-hdo}$
	RS and BO+ASHA overfit
CoLA	All HPO overfit
STS-B	RS succeeds w/ 1GST, $\mathcal{S}_{-wr-hdo}$
	ASHA and BO+ASHA succeed w/ 1GST, $\mathcal{S}_{min}$

Table 5: Final results of executing the troubleshooting procedure on Electra (top) RoBERTa (bottom). * means the risk of overfitting still exists based on the result of BO+ASHA.

grid search.

The Total Running Time for the Procedure. The execution for all experiments in Table 10 and 11 took 6.8×4V100 GPU days. This is in contrast to the cost if we enumerate all 5 factors in Section 3, which is 16×4V100 GPU days.

A Caveat on Results in Table 5. For all study results in this paper (i.e., Table 3, Table 10 and Table 11), we have repeated each HPO run three times. Therefore if a case succeed in Table 5, it is because no overfitting is detected in the 3 repetitions, if we ran more repetitions, the risk of overfitting can increase. In addition, all results are evaluated under transformers version=3.4.0 and Ray version=1.2.0. If these versions change, results in Table 5 may change.

An Analysis on the Relation between Overfitting and Train/Validation/Test split. As overfitting indicates a negative correlation between the validation and test accuracy, one hypothesis is that overfitting is caused by the different distribution of the validation and test set. We thus compare HPO runs using the original GLUE spilt and a new split which uniformly partition the train/validation/test data. The results can be found in Appendix A.5.

6 Related Work

6.1 Automated Hyperparameter Optimization

Hyperparameter optimization methods for generic machine learning models have been studied for a decade (Feurer and Hutter, 2019; Bergstra et al., 2011; Bergstra and Bengio, 2012; Swersky et al., 2013). Prior to that, grid search was the most common tuning strategy (Pedregosa et al., 2011). It discretizes the search space of the concerned hyperparameters and tries all the values in the grid. It can naturally take advantage of parallelism. However, The cost of grid search increases exponentially with hyperparameter dimensions. A simple yet surprisingly effective alternative is to use random combinations of hyperparameter values, especially when the objective function has a low effective dimension, as shown in (Bergstra and Bengio, 2012). Bayesian optimization (BO) (Bergstra et al., 2011; Snoek et al., 2012) fits a probabilistic model to approximate the relationship between hyperparameter settings and their measured performance, uses this probabilistic model to make decisions about where next in the space to acquire the function value, while integrating out uncertainty. Since the training of deep neural networks is very expensive, new HPO methods have been proposed to reduce the cost required. Early stopping methods (Karnin et al., 2013; Li et al., 2017, 2020) stop training with unpromising configurations at low fidelity (e.g., number of epochs) by comparing with other configurations trained at the same fidelity. Empirical study of these methods is mostly focused on the vision or reinforcement learning tasks, there has been few work focusing on NLP models. ASHA was evaluated on an LSTM model proposed in 2014 (Zaremba et al., 2014). In (Wang et al., 2015), the authors empirically studied the impact of a multi-stage algorithm for hyperparameter tuning. In (Zhang and Duh, 2020), a look-up table was created for hyperparameter optimization of neural machine translation systems. In BlendSearch (Wang et al., 2021), an economical blended search strategy was proposed to handle heterogeneous evaluation cost in general and demon-

strates its effectiveness in fine-tuning a transformer model Turing-NLRv2.[7] Some existing work has addressed overfitting in HPO (Lévesque, 2018) or neural architecture search (Zela et al., 2020). For HPO, cross validation can help alleviate the overfitting when tuning SVM (Lévesque, 2018), which is rarely applied in deep learning due to high computational cost. For neural architecture search (Zela et al., 2020), the solution also cannot be applied to our case due to the difference between the two problems.

6.2 Fine-tuning Pre-trained Language Models

As fine-tuning pre-trained language models has become a common practice, existing works have studied how to improve the performance of the fine-tuning stage. Among them, many has focused on improving the robustness of fine-tuning. For example, ULMFit (Howard and Ruder, 2018) shows that an effective strategy for reducing the catastrophic forgetting in fine-tuning is to use the slanted triangular learning rate scheduler (i.e., using a non-zero number of warmup steps). Other strategies for controlling overfitting in fine-tuning include freezing a part of the layers to reduce the number of parameters, and gradually unfreezing the layers (Peters et al., 2019), adding regularization term to the objective function of fine-tuning (Jiang et al., 2020), multi-task learning (Phang et al., 2018). Applying these techniques may reduce overfitting in our experiments; however, our goal is to compare grid search and HPO, if these techniques are helpful, they are helpful to both. To simplify the comparison, we thus focus on fine-tuning the original model. Meanwhile, the performance of fine-tuning can be significantly different with different choices of the random seeds (Dodge et al., 2020). To remove the variance from random seed, we have fixed all the random seeds to 42, although HPO can be used to search for a better random seed. (Zhang et al., 2021) identifies the instability of fine-tuning BERT model in few-sample cases of GLUE (i.e., RTE, MRPC, STS-B, and CoLA). Similar to our work, they also found that overfitting increases when searching for more trials. However, they have not compared grid search with HPO. There are also many discussions on how to control overfitting by tuning hyperparameters (in manual tuning), e.g., learning rate (Smith and Le, 2017), batch size (Smith et al., 2017), dropout rates (Srivastava et al., 2014; Loshchilov and Hutter, 2019, 2018), which may help with designing a search space for HPO that overfits less.

7 Conclusions, Discussions and Future Work

Our study suggests that for the problem of fine-tuning pre-trained language models, it is difficult for automated HPO methods to outperform manually tuned grid configurations with a limited time budget. However, it is possible to design a systematic procedure to troubleshoot the performance of HPO and improve the performance. We find that setting the search space appropriately per model and per task is crucial. Having that setting automated for different models and tasks is beneficial to achieve the goal of automated HPO for fine-tuning. For example, one may consider automatically mining the pattern from Table 8&9 to identify the hyperparameters that likely cause overfitting. Further, for the tasks remaining to be unsuitable for HPO, other means to reduce overfitting is required. One possibility is to use a different metric to optimize during HPO as a less overfitting proxy of the target metric on test data.

Previous work has shown that random seed is crucial in the performance of fine-tuning (Dodge et al., 2020). Fine-tuning also benefits from ensembling or selecting a few of the best performing seeds (Liu et al., 2019). It would be interesting to study HPO's performance by adding the random seed to the search space for future work.

In our study, the simple random search method stands strong against more advanced BO and early stopping methods. It suggests room for researching new HPO methods specialized for fine-tuning. A method that can robustly outperform random search with a small resource budget will be useful.

It is worth mentioning that although we find HPO sometimes underperforms grid search, the grid search configurations we study are the default ones recommended by the pre-trained language models for fine tuning, therefore they may be already extensively tuned. We may not conclude that HPO is not helpful when manual tuning has not been done. How to leverage HPO methods in that scenario is an open question.

[7] msturing.org

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, volume 33, pages 4271–4282. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Amog Kamsetty. 2020. Hyperparameter Optimization for HuggingFace Transformers: A guide. https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for-transformers-a-guide-c4e32c6c989b.

Zohar Karnin, Tomer Koren, and Oren Somekh. 2013. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR.

Julien-Charles Lévesque. 2018. Bayesian hyperparameter optimization: overfitting, ensembles and conditional spaces.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A system for massively parallel hyperparameter tuning. In *Machine Learning and Systems*.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2017. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*.

Samuel L. Smith and Quoc V. Le. 2017. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. 26.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chi Wang, Qingyun Wu, Silu Huang, and Amin Saied. 2021. Economic hyperparameter optimization with blended search strategy. In *International Conference on Learning Representations*.

Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. 2015. Efficient hyperparameter optimization for nlp applications. In *Conference on Empirical Methods in Natural Language Processing*, pages 2112–2117.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. 2020. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Xuan Zhang and Kevin Duh. 2020. Reproducible and efficient benchmarks for hyperparameter optimization of neural machine translation systems. *Transactions of the Association for Computational Linguistics*, 8:393–408.

A Appendix

A.1 HPO Checkpoint Settings

In this paper, we report the validation and test accuracy of the *best* checkpoint (in terms of validation accuracy) of the best trial instead of the *last* checkpoint of the best trial. While the default setting in Ray Tune uses the last checkpoint, when fine-tuning pretrained language model without HPO, the best checkpoint is more widely used than the last checkpoint. To further study the difference between the two settings, we compare their validation and test accuracy of grid search using Electra on three tasks: WNLI, RTE and MRPC. The result shows that the validation and test accuracy of the *best* checkpoint of the best trial are both higher than those of the *last* checkpoint of the best trial. As a result, we propose and advocate to report the *best* checkpoint of all the trials for HPO fine-tuning pretrained language models. The checkpoint frequencies in our experiment are set to 10 per epoch for larger tasks (SST, QNLI, and MNLI) and 5 per epoch for smaller tasks (WNLI, RTE, MRPC, CoLA and STS-B), with lower frequency in smaller tasks to reduce the performance drop caused by frequent I/Os within a short time.

A.2 Number of Trials Searched by HPO

In Table 6, we show the number of trials searched by each HPO algorithms in the initial comparative study (Table 3).

HPO	RS	ASHA	BO+ASHA
WNLI	4	12	12
RTE	6	27	38
MRPC	5	36	36
CoLA	9	31	30
STS-B	4	31	33
SST	5	33	30
QNLI	4	26	24
MNLI	7	31	27

Table 6: Average numbers of trials searched by each HPO algorithm in the initial experiment on Electra.

A.3 Choosing the Hyperparameter to Fix

The hyperparameters of the best trials in overfiting runs are shown in Table 8 and Table 9. We use colors to denote the comparison with the hyperparameter value in grid search: dark grey if the value is higher than grid search; light grey if the value is lower than grid search.

A.4 Execution Results of Procedure

In Table 10 and Table 11, we show the execution results of applying the experimental procedure to Electra and RoBERTa respectively.

A.5 An Analysis on Overfitting and Train/Validation/Test split

In this paper, we have observed that HPO tends to overfit when the number of trials/time budget increases. In other words, the higher the validation score, the lower the test score. One hypothesis for the reason behind this phenomenon is that the validation set has a different distribution than the test set. Since GLUE is a collection of NLP datasets from different sources, it is unclear whether the validation and test set in all GLUE tasks share the same distribution.

Origin		Resplit	
validation	test	validation	test
93.3	93.3	91.9	91.8
93.2	93.2	91.7	91.6
93.2	93.1	91.6	91.1
93.1	93.4	91.6	91.5

Table 7: Comparison of the orders of validation and test scores for the original split of GLUE and resplit.

To observe whether HPO still overfits under a uniformly random split, we have performed the following experiment: we merge the training and validation folds of QNLI in GLUE, randomly shuffle the merged data, and resplit it into train/validation/test with the proportion 8:1:1. We run random search, rank all trials based on the validation accuracy, and examine the Pearson correlation coefficient between the top-4 trials's validation and test accuracies (the trials are ranked by the validation accuracy), which are listed in Table 7. For the original GLUE dataset, we also save the best checkpoints of the top 4 trials and submit them to the GLUE website to get the test accuracies. The Pearson coefficient of the original dataset is $(r = -0.1414, p = 0.858)$ while for resplit it is $(r = 0.6602, p = 0.339)$. Thus one potential explanation of the observed overfitting in this work is due to different distribution between validation and test data.

HPO run	val acc	test acc	lr	wr	bs	hidd. do	att. do
MRPC, grid	92.3/89.2	91.1/87.9	1.0e-4	0.100	32	0.100	0.100
MRPC, RS, rep 1	92.7/90.0	90.4/87.1	3.9e-5	0.014	16	0.050	0.063
MRPC, RS, rep 2	93.4/90.9	90.6/87.6	4.3e-5	0.005	16	0.044	0.024
MRPC, ASHA, rep 1	92.8/90.0	90.8/87.6	6.5e-5	0.075	16	0.038	0.090
MRPC, ASHA, rep 2	93.4/90.9	90.5/87.4	3.1e-5	0.030	16	0.067	0.097
MRPC, ASHA, rep 3	92.9/90.0	90.4/86.9	1.3e-4	0.066	32	0.097	0.015
MRPC, Opt+ASHA, rep 1	93.0/90.4	90.7/87.5	6.4e-5	0.084	16	0.196	0.002
MRPC, Opt+ASHA, rep 2	93.3/90.7	90.4/86.9	8.0e-5	0.010	32	0.031	0.108
SST, grid	95.1	95.7	3.0e-5	0.100	32	0.100	0.100
SST, RS, rep 1	95.4	95.6	3.1e-5	0.011	32	0.006	0.044
STS-B, grid	91.5/91.4	89.7/89.2	1.0e-4	0.100	32	0.100	0.100
STS-B, Opt+ASHA, rep 1	91.6/91.4	89.6/89.1	4.7e-5	0.015	32	0.028	0.082

Table 8: Comparison between the hyperparameter values of the best trial of grid search and the best trials (in validation accuracy) of all the 9 overfitting HPO runs (out of 72) in the initial comparative study using Electra (Table 3). dark grey indicates the value is higher than grid search; light grey indicates the value is lower than grid search

HPO run	val acc	test acc	lr	wr	bs	hidd. do	att. do	wd
WNLI,grid	56.3	65.1	-	0.060	-	0.100	0.100	0.100
WNLI,RS,rep 3	60.6	64.4	1.8e-5	0.111	16	0.128	0.122	0.078
CoLA,grid	65.1	61.7	3.0e-5	0.060	16	0.100	0.100	0.100
CoLA,ASHA, rep 1	65.5	59.5	2.7e-5	0.020	32	0.090	0.197	0.180
CoLA,Opt+ASHA,rep 1	65.4	59.4	2.3e-5	0.067	32	0.063	0.117	0.293
RTE,grid	79.8	73.9	3.0e-5	0.060	16	0.100	0.100	0.100
RTE,RS,rep 1	80.5	73.6	2.8e-5	0.085	16	0.025	0.173	0.142
RTE,ASHA,rep 3	80.5	73.2	2.4e-5	0.022	16	0.053	0.137	0.016
RTE,Opt+ASHA,rep 2	81.9	73.5	2.7e-5	0.024	32	0.083	0.190	0.094
MRPC,grid	93.1/90.4	90.5/87.1	2.0e-5	0.060	16	0.100	0.100	0.100
MRPC,RS,rep 2	93.2/90.7	89.6/86.1	2.4e-5	0.094	64	0.019	0.138	0.299
MRPC,RS,rep 3	93.2/90.4	90.3/86.7	1.4e-5	0.003	16	0.011	0.062	0.176
MRPC,ASHA,rep 3	93.3/90.7	90.3/86.8	2.7e-5	0.008	16	0.140	0.130	0.255
MRPC,Opt+ASHA,rep 3	93.5/91.2	89.6/86.2	2.7e-5	0.036	16	0.094	0.153	0.291
STS-B,grid	91.2/90.8	89.3/88.4	2.0e-5	0.060	16	0.100	0.100	0.100
STS-B,ASHA,rep 1	91.3/91.0	89.0/88.2	2.0e-5	0.042	16	0.004	0.061	0.247
STS-B,ASHA,rep 2	91.4/91.1	89.0/88.2	2.1e-4	0.061	16	0.056	0.008	0.226
STS-B,Opt+ASHA,rep 1	91.3/90.9	89.1/88.2	2.7e-5	0.052	16	0.096	0.070	0.224

Table 9: Comparison between the hyperparameter values of the best trial of grid search and the best trials (in validation accuracy) of all the 11 overfitting HPO runs (out of 45) in the initial comparative study using RoBERTa (Table 3). dark grey indicates the value is higher than grid search; light grey indicates the value is lower than grid search

	Random Search								ASHA							
	round 0		round 1		round 2		round 3		round 0		round 1		round 2		round 3	
	val	test	val	test	val	test	val	test	val	test	val	test	val	test	val	test
WNLI	**56.3**	**65.1**	↓ space								↓ space					
	57.7	62.3	57.7	65.8					57.7	63.0	59.2	65.8				
	56.3	65.8	57.7	65.1					57.7	59.6	57.7	65.1				
	56.3	65.1	57.7	65.1					56.3	65.1	57.7	65.8				
	56.8	64.4	57.7	65.3					57.2	62.6	58.2	65.6				
RTE	**84.1**	**76.8**	↑ res		↓ space		↓ space				↑ res					
	81.9	76.1	84.5	74.6	84.1	76.1	84.8	75.3	81.9	76.2	83.4	75.3				
	81.6	75.1	83.8	74.5	83.0	74.0	84.1	75.7	75.5	72.1	81.9	73.9				
	83.0	75.7	83.4	74.7	82.3	73.1	83.8	75.2	83.4	74.1	83.8	74.4				
	82.2	75.6	83.9	74.6	83.1	74.4	84.2	75.4	80.3	74.1	83.0	74.5				
MRPC	**89.2**	**87.9**	↓ space		↓ space				↓ space		↓ space					
	90.9	87.6	90.7	86.3	90.4	86.5			90.9	87.4	90.0	87.2	90.2	87.6		
	90.0	87.1	90.2	87.2	90.7	86.5			90.0	86.9	90.4	87.8	90.9	88.3		
	90.2	87.8	90.7	86.9	90.7	87.8			90.0	87.6	89.5	86.0	90.7	87.6		
	90.4	87.5	90.5	86.8	90.6	87.4			90.3	87.3	90.4	87.0	90.6	87.8		
STS-B	**91.4**	**89.2**	↑ res								↑ res					
	90.8	89.1	91.5	89.4					91.3	89.2	91.5	89.8				
	89.6	85.9	91.4	89.6					91.5	89.7	91.4	89.2				
	90.1	87.7	91.5	89.9					91.0	88.3	91.4	89.2				
	90.2	87.6	91.4	89.6					91.3	89.1	91.4	89.4				
SST	**95.1**	**95.7**	↓ space		↑ res		↓ space				↑ res		↓ space		↓ space	
	95.4	95.6	93.2	93.8	96.0	94.7	95.6	95.2	95.4	95.8	95.5	95.3	95.5	95.2	95.2	94.9
	94.3	95.1	94.7	95.0	95.3	95.7	95.1	95.7	94.4	94.1	95.1	94.7	94.8	94.3	94.2	93.6
	94.5	94.6	95.8	95.7	95.5	95.8	95.0	94.5	95.0	94.9	95.4	95.4	94.5	93.5	94.8	94.5
	94.7	95.1	94.6	94.8	95.6	95.4	95.2	95.1	94.9	94.9	95.3	95.1	94.9	94.3	94.7	94.3
QNLI	**93.5**	**93.5**	↑ res								↑ res					
	93.0	92.9	93.2	93.4					92.5	92.4	93.4	93.2				
	93.1	93.6	93.3	93.3					93.4	93.0	93.2	93.1				
	92.9	92.5	93.3	93.1					93.4	93.4	93.2	93.0				
	93.0	93.0	93.3	93.3					93.1	92.9	93.3	93.1				

Table 10: The execution results of applying the procedure on Electra. Each task's grid search accuracy is denoted using the **blue bold font**. An HPO run is highlighted in dark grey if it overfits and medium grey if it overfits weakly. The average of 3 repetitions is highlighted in light grey if it outperforms grid search's validation and test accuracy. For STS-B we only report the Spearman correlation, for MRPC we only report the accuracy.

	Random Search								ASHA							
	round 0		round 1		round 2		round 3		round 0		round 1		round 2		round 3	
	val	test	val	test	val	test	val	test	val	test	val	test	val	test	val	test
WNLI	**56.3**	**65.1**	↓ space		↓ space						↓ space		↓ space			
	60.6	64.4	62.0	64.4	57.7	62.3			59.2	65.1	59.2	65.1	57.7	65.8		
	56.3	65.1	56.3	65.1	56.3	65.1			56.3	65.1	56.3	65.1	56.3	65.1		
	56.3	65.1	56.3	65.1	56.3	65.1			56.3	65.1	56.3	65.1	56.3	65.1		
	57.8	64.9	58.2	64.9	56.8	64.2			57.3	65.1	57.3	65.1	56.8	65.3		
RTE	**79.8**	**73.9**	↓ space		↓ space						↓ space		↓ space			
	81.2	73.9	80.1	72.8	81.6	72.2			80.5	73.2	80.5	73.3	79.8	72.5		
	80.5	73.6	81.2	72.9	75.5	72.1			80.2	74.9	82.0	72.9	79.1	73.4		
	79.4	73.1	79.8	73.6	79.8	72.6			80.5	74.1	80.5	73.5	79.8	73.7		
	80.4	73.5	80.4	73.1	78.9	72.3			80.8	74.1	80.5	73.3	79.5	73.2		
MRPC	**90.4**	**87.1**	↓ space		↓ space						↓ space					
	90.7	86.1	90.7	86.9	91.2	86.7			90.7	86.8	91.4	87.7				
	90.4	86.7	90.4	88.0	90.2	87.6			90.4	87.4	90.4	87.2				
	90.9	87.2	91.2	87.2	90.4	87.0			91.4	87.6	90.4	87.6				
	90.7	86.7	90.8	87.4	90.6	87.1			90.8	87.3	90.8	87.5				
CoLA	**65.1**	**61.7**	↑ res		↓ space		↓ space				↓ space		↓ space			
	64.3	60.1	66.0	59.3	65.8	59.2	65.3	60.2	65.5	59.5	65.0	60.9	65.9	58.2		
	64.6	60.5	65.0	60.5	65.0	61.7	65.4	62.5	63.6	58.8	62.9	58.4	63.9	58.9		
	63.5	56.8	64.4	60.3	65.2	60.7	64.6	58.5	64.6	60.0	64.9	62.0	64.4	59.0		
	64.1	59.1	65.1	60.0	65.3	60.5	65.1	60.4	64.5	59.4	64.3	60.4	64.7	58.7		
STS-B	**90.8**	**88.4**	↓ space								↓ space		↓ space			
	90.8	88.3	91.0	88.9					91.1	88.2	90.9	88.3	90.8	88.6		
	90.8	88.9	90.8	88.6					91.0	88.2	90.8	88.5	91.0	88.5		
	91.2	88.7	90.9	88.9					90.7	88.5	90.9	88.4	90.9	88.7		
	90.9	88.6	90.9	88.8					90.9	88.3	90.8	88.4	90.9	88.6		

Table 11: The execution results of applying the procedure on RoBERTa. Each task's grid search accuracy is denoted using the **blue bold font**. An HPO run is highlighted in dark grey if it overfits and medium grey if it overfits weakly. The average of 3 repetitions is highlighted in light grey if it outperforms grid search's validation and test accuracy. For STS-B we only report the Spearman correlation, for MRPC we only report the accuracy.