Les Cahiers du GERAD ISSN: 0711–2440

Hyperparameter optimization for Large Language Model instruction-tuning

C. Tribes, S. Benarroch-Lelong, P. Lu, I. Kobyzev

G–2023–62

December 2023
Revised: February 2024

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2023
– Bibliothèque et Archives Canada, 2023

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2023
– Library and Archives Canada, 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Hyperparameter optimization for Large Language Model instruction-tuning

Christophe Tribes [a]

Sacha Benarroch-Lelong [a]

Peng Lu [b, c]

Ivan Kobyzev [b]

[a] *Département de mathématiques et de génie industriel, Polytechnique Montréal & GERAD, Montréal (Qc), Canada*

[b] *Huawei Noah's Ark Lab, Montréal (Qc), Canada*

[c] *RALI, Université de Montréal, Montréal (Qc), Canada*

sacha.benarroch@polymtl.ca
christophe.tribes@polymtl.ca

December 2023
Revised: February 2024
Les Cahiers du GERAD
G–2023–62

Abstract : The fine-tuning of Large Language Models (LLMs) has enabled them to recently achieve milestones in natural language processing applications. The emergence of ever larger LLMs has paved the way for more efficient fine-tuning methods. Among these, the Low-Rank Adaptation (LoRA) method keeps most of the weights of the pre-trained LLM frozen while introducing a low-rank decomposition of the weight matrix, enabling the tuning of only a very small proportion of the network. The performance on downstream tasks of models fine-tuned with LoRA heavily relies on a set of hyperparameters including the rank of the decomposition. In this work, we investigate the choice of these hyperparameters through two main blackbox optimization (BBO) techniques. We examine the whole pipeline of performing fine-tuning and validation on a pre-trained LLM as a blackbox and efficiently explore the space of hyperparameters with the NOMAD algorithm, achieving a boost in performance and human alignment of the tuned model.

Acknowledgements: This work is supported by the NSERC Alliance grant 544900-19 in collaboration with Huawei-Canada and by the NSERC Alliance-Mitacs Accelerate grant ALLRP 571311-21 ("Optimization of future energy systems") in collaboration with Hydro-Québec.

The authors want to thank Sébastien Le Digabel and Vahid Partovi Nia for their support and constructive comments.

1 Introduction

Large-scale Language Models (LLMs) have shown exceptional ability in language understanding and generation [8, 28, 29, 39]. State-of-the-art models like ChatGPT [25] and GPT-4 [26] have garnered a great deal of interest from the academic and industrial communities. One of the main challenges of LLMs is how to control their behavior and make them follow specific instructions given by users [27]. Additional fine-tuning of LLMs on a dataset of instructions is called Instruction-Tuning; this technique has become ubiquitous due to its efficiency [38]. However, tuning large models demands a large amount of computer power. To overcome this, a common practice is to use Parameter Efficient Fine Tuning (PEFT) methods, which modify a limited selection of parameters in a pre-trained LLM while leaving the rest unchanged [24]. Such methods are quite sensitive to the choice of hyperparameters [18, 34]. In this work we investigate how hyperparameter optimization can better the instruct-tuning results.

Hyperparameters selection by a human in order to tune a model is a tedious task but it can significantly improve model performance. Bergstra et al. 2011 suggest that hyperparameters optimization (HPO) forms the outer loop of a learning process. Applying an algorithmic approach to automate the process in finding better hyperparameters should also bring some efficiency. A grid search algorithm is a systematic but inefficient approach that tries a finite number of hyperparameters combinations. A blackbox optimization (BBO) algorithm should be a better choice for solving HPO efficiently within a fixed computational budget.

In this work we investigated how two BBO solvers implementing different types of algorithms, namely MADS (a direct search algorithm implemented in NOMAD) and TPE (a Bayesian model-based optimization algorithm implemented in NNI) behave when used to solve HPO for the Instruction-tuning of a specific LLM. We found different patterns in hyperparameter selection for these two optimizers, and assessed their effects on downstream tasks. Overall, we confirmed the necessity of careful HP selection in Instruction-tuning for performance boosting, both in downstream tasks and human preference.

2 Instruction-tuning Large Language Model

Instruction-tuning has emerged recently as an important training paradigm [27, 30, 35, 36] to better adapt pre-trained models for human needs and enhance their ability to comprehend and respond to a diverse range of human requests. Instruction-tuning is an additional training step for LLMs when the models are fine-tuned on a dataset of instruction and output pairs [12, 20, 22, 32]. It aims to bridge the gap between the next-word prediction objective of a language model and the users' objective of having LLMs follow their instructions across various tasks and domains.

2.1 Parameter-Efficient Fine-Tuning (PEFT)

The success of Instruction-tuning heavily relies on a powerful model with at least several billion parameters. Tuning of such models is usually difficult due to high computational costs in both time and memory. To circumvent this bottleneck, researchers developed Parameter-Efficient Fine-Tuning (PEFT) methods [24]: instead of training all parameters, one freezes the majority of parameters in pre-trained models and only updates an incremental number of parameters.

There are various PEFT techniques generally falling into two groups: Prompt Tuning [21] when a few trainable tokens are added to the prompt; and different kinds of Adaptors [14, 16] when extra trainable layers are inserted between layers of the pre-trained model. In this work, we utilize the Low-Rank Adaptation (LoRA) [17] method which adds trainable low-rank matrices to every model weights during training and merges the added parameters to the original pre-trained matrices for inference. The performance of LoRA-tuned models is very sensitive to the rank selection [34], hence the rank needs to be carefully picked for each dataset: too large rank could result in more overfitting on small datasets, yet a small rank may fail to capture the diversity of complicated instructions. Another

important hyperparameter for LoRA is a scaling factor (LoRA-α), which determines the scaling of low rank blocks that are added to the frozen parameters. We perform hyperparameters optimization (HPO) to select the optimal combination of these and some other LoRA hyperparameters to improve the performance of the tuned model.

3 Hyperparameters optimization

In this work, the aim of hyperparameters optimization (HPO) is to obtain a fine-tuned model with the best performance measure. NOMAD and NNI-TPE are considered for solving this HPO problem.

3.1 The MADS algorithm and NOMAD

NOMAD[1] [5] is a software package for solving blackbox optimization (BBO) problems [3] in which there is no analytical expressions for objective and constraint functions. The optimization problems have the following general form

$$\min_{x \in \mathcal{X} \subseteq \mathbb{R}^n} \{f(x) \ : \ c(x) \leq 0\}, \tag{1}$$

where $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $c : \mathcal{X} \subseteq \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^m$ are the given functions. The function properties are not known and their evaluations are typically obtained after a computer program execution, with provided inputs and observed outputs. In addition, a blackbox function evaluation may take a significant amount of time and may fail to return valid outputs. A HPO problem can be framed as a BBO problem where the objective function is linked to a performance measure of a model and the hyperparameters are the variables x.

NOMAD implements the mesh adaptive direct search (MADS) algorithm [1]. MADS is supported by a rigorous hierarchical convergence analysis based on various degrees of smoothness of the functions defining the problem. The MADS algorithm iterates *search* and *poll* steps to generate trial points on a *mesh* discretizing the space of variables. The search step generates trial points disseminated more globally in the space of variables. The poll step generates trial points around the current best solutions following rigid rules to ensure convergence to points satisfying some necessary optimality conditions. The mesh size may be adapted at each iteration. In addition, the mesh properties support by construction real variables, binary variables and granular variables [4]. The mesh adaptation combined with the poll and search steps allows to explore more globally early during the optimization and more locally when the mesh is refined. This is one advantage of the MADS algorithm .

The MADS algorithm can handle general inequality constraints using the progressive barrier [2] approach to exploit the measure of constraint violation. NOMAD includes BBO algorithms other than MADS. In particular, DMULTI-MADS [7] solves multiobjective optimization problems seeking detailed Pareto fronts. Hence, NOMAD is suited to solve HPO problems with or without inequality constraints or that can have multiple objectives.

3.2 Neural Network Intelligence (NNI) toolkit

Microsoft Neural Network Intelligence[2] (NNI) is an open-source toolkit to automate machine learning techniques such as hyperparameters optimization, model pruning, quantization, neural architecture search (NAS) and feature engineering. Among the tuning algorithms available in NNI we have selected the Tree-structured Parzen Estimator [6] (TPE) which is a Bayesian model-based optimization method. Bayesian optimization methods are appropriate to balance exploration and exploitation of the variable space with a limited evaluation budget.

TPE performs a series of optimization on a model of the objective function f that is cheaper to evaluate (inner loop). A Gaussian Process (GP) is used to build the model and the inner loop aims

[1] Available at https://www.gerad.ca/nomad and https://github.com/bbopt/nomad.
[2] Software available at https://github.com/microsoft/nni

to maximize the expect improvement (EI) of f. As new trial points are evaluated new models are fit based on the overall observation history. This process of sequential model-based optimizations [19] (SMBO) can be repeated until the evaluation budget is used. TPE is best suited for single objective HPO without inequality constraints.

4 Experimental setup

4.1 Instruction-tuning settings

Backbone Model LLaMA is a family of open-sourced large language models including models ranging from 7B to 65B parameters [33]. As our experiments aim at investigating the behavior of BBO algorithms, we conduct them with the 7 billions parameter version of LLaMA 2.[3] The fine-tuning of LLaMA 2 is done via the LoRA method. The method has some specific hyperparameters that we explore with BBO (see Section BBO Settings below for details).

Datasets To perform our fine-tuning procedure, we use a mix of two same-structured instruction-following datasets (see Table 2 in Appendix). First is the 52k-entry dataset used in the Stanford Alpaca Project [32], that features a large diversity of instructions. Second is Databricks' Dolly dataset [12] containing 15k entries. We build a 54k-sized training set and a 13k-sized validation set, both containing 70% of data from the Alpaca dataset and 30% from Dolly, ensuring an identical distribution.

Training Details The fine-tuning procedure minimizes the training loss by adapting LoRA trainable parameters. Once fine-tuned the validation loss of the model is computed. The HuggingFace Tranformers API [37] is used for handling the model, its training and validation on datasets. The default AdamW optimizer [23] is selected for training with a batch size fixed to 4. This pipeline is run on four NVIDIA-A100 GPUs with 80 GB memory.

4.2 BBO Settings

In addition to LoRA rank, LoRA scaling α and dropout rate, we also seek to optimize the learning rate that impacts the reduction of training loss (see Table 3 in Appendix).

For the problem at hand, we can consider different types of performance measure. The model training procedure indirectly seeks a fine-tuned model with low validation loss. But, the validation datasets are relatively small and a model may not generalize well. Hence, other performance measures on various instruction-following benchmarks are necessary to assess models' downstream capability but it would be very time-consuming if done during optimization.

Moreover, considering multiple measures could require to use a multiobjective BBO formulation which demands a larger evaluation budget to obtain a refined Pareto front. In this work, to control the HPO computation time we chose a fixed and relatively small evaluation budget. Also, we decided to test if the validation loss computed at the last epoch can be used as the BBO single objective function. For validation on downstream tasks, we need to perform post-optimization assessments on several candidates.

5 Experimental results

5.1 First optimization round

A first optimization using NOMAD was conducted to validate several a priori choices. We started with a budget of 50 evaluations with 3 epochs. The validation loss is computed at each epoch. The duration

[3] https://huggingface.co/meta-llama/Llama-2-7b-hf

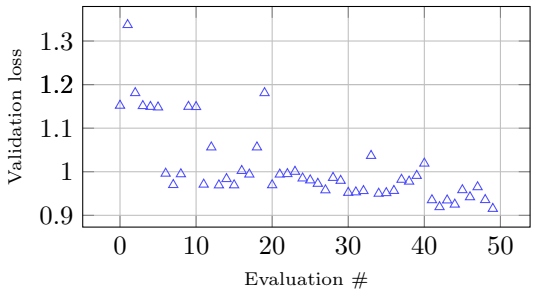


Figure 1: Objective value history. First NOMAD optimization with 50 evaluations and a 3 epochs fine-tuning.

of a single evaluation is around 2 hours and 15 minutes. It took less than 5 days to complete this optimization.

As expected, hyperparameter selection affects the fine-tuning training process. The smallest validation losses are obtained for evaluation points featuring the highest reduction in training losses. The best evaluation happens to be the last one. In addition, from the optimization history (see Figure 1) we can expect further reduction of the validation loss given an increased evaluation budget. From the intermediate fine-tuning training steps (not shown here) we realize that on most evaluation points there is no significant change in validation loss between epoch 2 and epoch 3. Also, we observe that most of the best evaluation points have a LoRA rank value of 128, which is the upper bound for this variable.

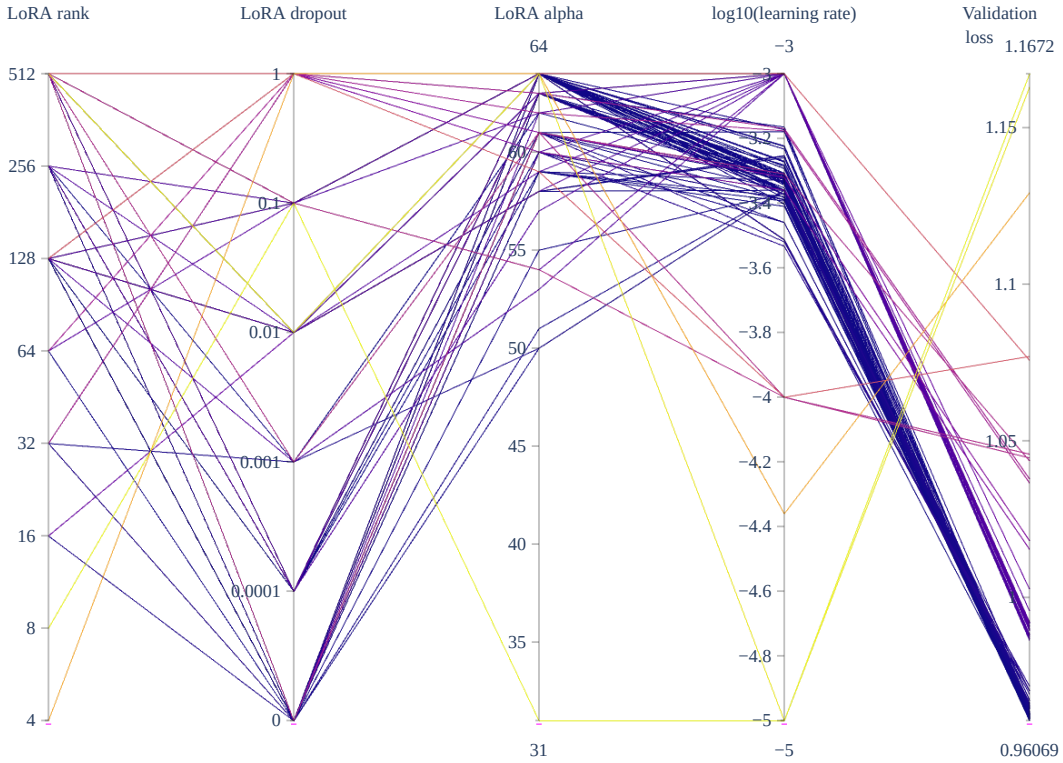
5.2 Second optimization round

For the next step, NOMAD and NNI-TPE were used for HPO on 100 evaluations with 2 epochs. We also decided to increase the LoRA rank upper bound to 512 in order to explore how it impacts model fine-tuning; in particular the capability to capture the diversity of instructions with the possible overfitting drawback.

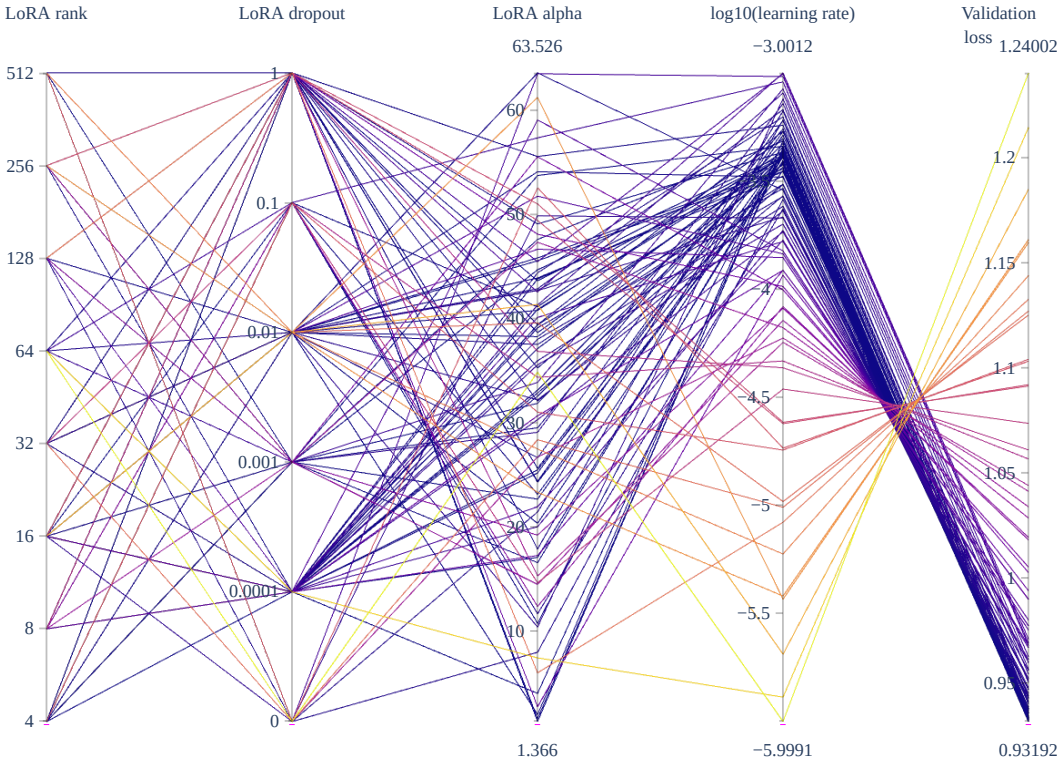
The evaluation points obtained during the first round were given in a cache file to jump-start NOMAD in the second round. These points were used during MADS search steps to construct quadratic models of the objective function and propose new promising trial points.

NOMAD results Figure 2a shows the hyperparameters combinations assessed by NOMAD during this experiment and the validation loss yielded by the corresponding fine-tuned models. It makes clear that a learning rate around $10^{-3.5}$ and a scaling parameter α around 60 yield the best results. Among the 10 best evaluations points, 5 have LoRA rank $r = 512$ (including the best one), 4 have $r = 256$ and 1 has $r = 128$. The trend observed in first round linking large rank and lower validation loss is again observed. NOMAD has obtained efficient hyperparameters combinations in high-rank regions and has put the emphasis on exploitation through refining the other hyperparameters. By activating optional exploration methods in the search step, NOMAD may have produced more trial points in low-rank regions. Moreover, feeding the algorithm with a cache file from the first round may have introduced a bias in the search step in favor of these high-rank regions.

NNI-TPE results Compared with the hyperparameters values tested by NOMAD, NNI-TPE (see Figure 2b) shows more variety confirming its explorative capability. It also obtains evaluation points with lower validation losses compared to the best of NOMAD. Among the 10 best evaluations points of NNI-TPE, only 2 have a LoRA rank higher than 32, the best one having rank 8. This result shows that increasing LoRA rank is not the only way to obtain a lower validation loss, rather that low rank can perform well provided that other hyperparameters are chosen adequately.



(a) NOMAD **second round**



(b) NNI-TPE

Figure 2: Parallel plots showing hyperparameters values and validation losses. Darker lines indicate lower validation losses.

5.3 Evaluation of candidate best models

Validation of the best candidate models was performed on downstream instruction-following tasks. Instruct-Eval[4] [10] source codes and datasets are used to automate evaluation and obtain scores on a series of instruction-following tasks. The benchmarks considered in this work are MMLU [15], BBH [31], DROP [13] and HumanEval [9] and are of quite different natures.

Table 4a in Appendix shows the scores of the 10 best and 10 worst models (in terms of validation loss) explored by NOMAD during the second optimization round. In summary, the model ranked first does not give the best scores. Nevertheless, the 10 best models have very close validation losses. The 10 best models outscore the 10 worst models (including the one with default fine-tuning hyperparameters) and the baseline (without fine-tuning) for MMLU and HumanEval. For the BBH and DROP benchmarks the trend is not as clear.

Data from the 10 best models (validation loss) explored by NOMAD (Table 4a) and NNI-TPE (Table 4b) optimizations is summarized in Table 1. The 10 best MMLU, DROP and HumanEval scores are lower in average for NNI-TPE than what is obtained by NOMAD even though NNI-TPE obtains the lowest validation loss. When judging by Instruct-Eval performance measures we can conclude that HPO using validation loss as objective function results in better models. However, lower validation losses do not necessarily translate into higher benchmark scores. With the current HPO problem formulation several candidates should be considered before selecting the best model for a downstream task.

Table 1: Statistics of the 10 best models on downstream instruction-following tasks.

Method		min	max	avg.	st. d.
NOMAD	MMLU	45.88	46.7	46.24	0.29
	BBH	32.07	32.99	32.50	0.25
	DROP	29.67	30.95	30.28	0.45
	HumanEval	14.63	18.9	16.94	1.52
NNI-TPE	MMLU	45.49	46.56	46.08	0.31
	BBH	32.27	34.43	32.93	0.42
	DROP	29.23	30.77	30.03	0.61
	HumanEval	14.02	16.46	15.24	0.91
Default HPs	MMLU	43.56			
	BBH	32.13			
	DROP	29.02			
	HumanEval	15.24			

Human Preference We also conducted Human evaluation to check whether the generated results are aligned with human preferences. We sampled 30 questions randomly from the Vicuna [11] human preference dataset[5] and asked Human evaluators to compare the answers generated by two models: the one tuned with NOMAD as described above and the one with the default hyperparameters for LoRA. For each question, all evaluators are asked to judge which answer is better without knowing the source of answer. Figure 3 shows that our HP-tuned model has a clear human preference compared to the default one by an overall preference score of 5%.

[4] https://github.com/declare-lab/instruct-eval
[5] https://github.com/lm-sys/vicuna-blog-eval

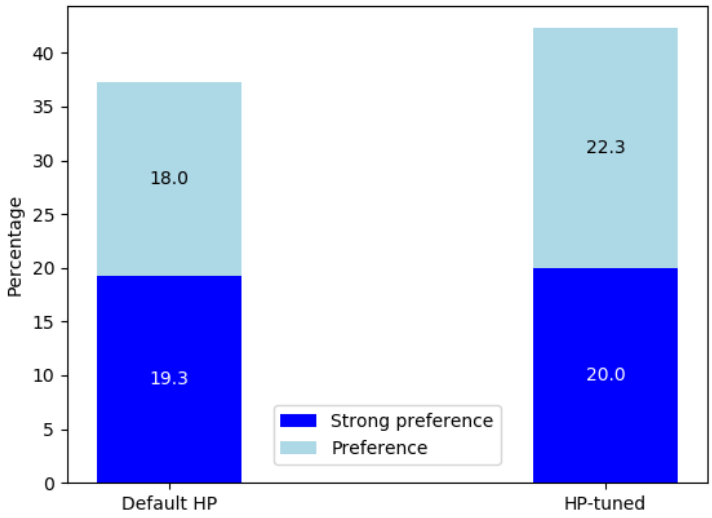


Figure 3: Human evaluation on the Vicuna human preference dataset.

6 Conclusion

Hyperparameters optimization using blackbox optimization algorithms improves the performance of fine-tuned LLMs on downstream tasks and human evaluation. In particular, the best models are better than the model with default fine-tuning parameters. Also, for the three out of the four downstream tasks, the best candidate models are obtained by NOMAD. NNI-TPE found candidate models with performance relatively close to those obtained by NOMAD but with clearly lower LoRA ranks suggesting that different sets of hyperparameters may be optimal. More experiments should be conducted to either identify a single proper set of hyperparameters for LLM fine-tuning or to conclude that hyperparameters optimization should form the outer loop for every LLM fine-tuning whenever possible.

The experiments show that validation losses are not perfectly aligned with downstream tasks scores. As future work we aim to develop an efficient and robust methodology to pickup a single best model. This can be achieved by guiding the blackbox optimization to consider more criteria into the HPO problem. Not all BBO algorithms offers enough flexibility to consider inequality constraints and multiple objectives. NOMAD is a good option to handle such problems.

Appendix

A Datasets

Table 2: Instruction datasets used in this work. We report also the number of data samples and the average length of prompts (Avg. L), the average length of completion (Avg. C).

Dataset	Type	# samples	Avg. L	Avg. C
Alpaca	LLM	52,002	27.8	64.6
Dolly	Human	15,011	118.1	91.3

B BBO settings

Table 3 gives the mapping between the variables handled by NOMAD and the hyperparameters for evaluation. The type of variables, the bounds and initial values for the hyperparameters are also provided in this table. Contrary to NOMAD, TPE does not require a special mapping between its variables and the hyperparameters; initial values are not required either.

Default values reported in Table are taken from the HuggingFace PEFT documentation.

Table 3: Mapping NOMAD variables into hyperparameters (right column) and initial values.

Rank	Int. $r \in [1;8]$	$\rightarrow 2^{r+1}$
Dropout	Int. $d \in [1;6]$	$\rightarrow 0$ if $d = 1$ else 10^{6-d}
α	Int. $\alpha \in [1;64]$	$\rightarrow \alpha$
LR	Real $lr \in [-6;-3]$	$\rightarrow 10^{lr}$
Default values	$r = 2$ (Rank=8), $d = 5$ (Dropout=0.1) $\alpha = 32$, $lr = -5$ (LR=0.00001)	

C Second round detailed results

As our goal is a general-purposed model, we are also interested in Pareto optimality. A model is Pareto optimal if it is not dominated by any other model (among the ones evaluated). Picking-up a model is easier when the optimization returns a single Pareto optimal solution. Otherwise, Pareto optimal models have particular trade-offs between the different scores.

Table 4a shows the scores of the 10 best and 10 worst models (in terms of validation loss) explored by NOMAD during the second optimization round. We can note that no single model dominates all remaining models in Table 4a. Interestingly, the models ranked 6 and 8 are Pareto optimal, whereas

Table 4: Instruct-Eval scores on the models generated by the two optimizers during the second optimization round. Models are ranked by increasing validation loss. The 10 best and 10 worst models are displayed. Best score values are in bold. ⋆ indicates Pareto optimality for this subset. † marks the model with default LoRA hyperparameters.

Ranking (valid. loss)		MMLU	BBH	DROP	HumanEval
1		45.94	32.51	29.71	17.07
2	⋆	46.00	32.68	**30.95**	17.68
3		46.18	32.16	30.63	15.85
4	⋆	**46.70**	32.37	30.15	18.29
5		46.42	32.07	30.33	18.29
6	⋆	45.98	32.99	29.77	17.68
7	⋆	46.46	32.50	30.95	**18.90**
8	⋆	46.57	32.60	29.67	14.63
9		46.28	32.42	30.29	16.46
10		45.88	32.67	30.39	14.63
91		42.48	31.43	28.62	12.20
92		42.47	32.30	28.40	12.80
93		42.44	30.45	28.62	12.80
94	⋆	45.98	**33.40**	30.45	13.41
95	⋆	45.09	32.77	30.85	15.24
96		42.32	30.98	29.01	13.41
97		42.64	31.24	27.53	14.02
98		42.88	32.09	28.08	12.80
99		43.45	32.42	30.26	15.24
100	†	43.56	32.13	29.02	15.24
w/o fine-tuning		42.37	31.41	28.66	14.63

(a) NOMAD

Ranking (valid. loss)		MMLU	BBH	DROP	HumanEval
1	⋆	**46.56**	32.41	30.26	14.63
2	⋆	46.23	**34.43**	30.15	14.02
3	⋆	46.28	32.86	29.28	**16.46**
4	⋆	46.40	32.27	29.77	15.85
5	⋆	45.94	32.83	30.58	14.02
6	⋆	45.84	33.49	30.25	**16.46**
7		46.13	32.3	29.72	14.63
8	⋆	46.06	32.9	30.34	15.85
9	⋆	45.91	32.78	30.77	15.24
10		45.49	33.03	29.23	15.24
91		43.16	32.02	28.91	14.63
92		43.10	32.38	29.79	15.85
93		43.27	31.54	29.26	14.02
94		43.46	31.55	28.80	14.02
95		43.42	31.65	28.94	14.63
96		42.81	32.05	29.23	14.02
97		43.01	31.45	32.41	14.02
98	⋆	42.86	32.00	**32.94**	14.63
99	⋆	46.20	32.04	32.78	15.85
100		46.05	31.24	28.89	14.02
w/o fine-tuning		42.37	31.41	28.66	14.63

(b) NNI-TPE

they do not achieve the best value for any score. In fact, a model outperforming in one kind of benchmark score can indicate its overspecialization.

Table 4b shows the scores of the 10 best and 10 worst models (validation loss) explored by NNI-TPE optimization. For BBH and DROP, when comparing NOMAD and NNI-TPE, similar scores are obtained. The 10 best MMLU scores and HumanEval scores are lower for NNI-TPE than what is obtained by NOMAD even though NNI-TPE obtains the lowest validation loss.

D Human evaluation setting

The evaluation is conducted with Google Forms with 30 instructions in that form. The ordering of the questions and the responses are totally randomized. We found 10 experienced volunteering annotators who are fluent in English and hold bachelor's degrees or above.

References

[1] C. Audet and J.E. Dennis, Jr. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. SIAM Journal on Optimization, 17(1):188–217, 2006. doi: 10.1137/040603371. URL https://dx.doi.org/10.1137/040603371.

[2] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. SIAM Journal on Optimization, 20(1):445–472, 2009. doi: 10.1137/070692662. URL https://dx.doi.org/10.1137/070692662.

[3] C. Audet and W. Hare. Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017. doi: 10.1007/978-3-319-68913-5. URL https://dx.doi.org/10.1007/978-3-319-68913-5.

[4] C. Audet, S. Le Digabel, and C. Tribes. The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables. SIAM Journal on Optimization, 29(2):1164–1189, 2019. doi: 10.1137/18M1175872. URL https://dx.doi.org/10.1137/18M1175872.

[5] C. Audet, S. Le Digabel, V. Rochon Montplaisir, and C. Tribes. Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm. ACM Transactions on Mathematical Software, 48(3):35:1–35:22, 2022. doi: 10.1145/3544489. URL https://dx.doi.org/10.1145/3544489.

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[7] J. Bigeon, S. Le Digabel, and L. Salomon. DMulti-MADS: Mesh adaptive direct multisearch for bound-constrained blackbox multiobjective optimization. Computational Optimization and Applications, 79(2):301–338, 2021. doi: 10.1007/s10589-021-00272-9. URL https://dx.doi.org/10.1007/s10589-021-00272-9.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[10] Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. Instructeval: Towards holistic evaluation of instruction-tuned large language models, 2023.

[11] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[12] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

[13] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Proc. of NAACL, 2019.

[14] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. ArXiv, abs/2110.04366, 2021. URL https://api.semanticscholar.org/CorpusID:238583580.

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR), 2021.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 2790–2799. PMLR, 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html.

[17] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. CoRR, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[19] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In International Conference on Learning and Intelligent Optimization, pages 507–523. Springer, 2011.

[20] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations–democratizing large language model alignment. arXiv preprint arXiv:2304.07327, 2023.

[21] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. CoRR, abs/2110.07602, 2021. URL https://arxiv.org/abs/2110.07602.

[22] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 22631–22648. PMLR, 2023. URL https://proceedings.mlr.press/v202/longpre23a.html.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[24] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[25] OpenAI. Chatgpt (Version Date version) [large language model], 2023. URL https://chat.openai.com.

[26] OpenAI. GPT-4 technical report. CoRR, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

[27] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In NeurIPS, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

[30] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, 2022.

[31] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.

[32] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[33] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[34] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. ArXiv, abs/2210.07558, 2022. URL https://api.semanticscholar.org/CorpusID:252907428.

[35] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. ArXiv preprint, 2022.

[36] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022, 2022.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[38] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey. ArXiv, abs/2308.10792, 2023.

[39] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068, 2022.