RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Scalable Hyperparameter Optimization for LLM Fine-Tuning

Bayesian and Partition-based optimization

N. Davouse

Summary

01
Introduction

Large Language Models

Summary

- State-of-the-art of Natural Language Processing (NLP) problems
- Architecture : Transformers[18] block, mixed with classical layers (MLP, Conv)
- Huge size : Billions of parameters (1B to 405B for Llama 3)
- 2 phases of training : pre-training and **fine-tuning**

Self Attention

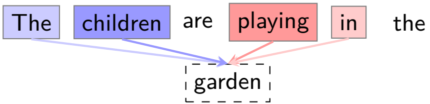


Figure: Self Attention mecanism illustration

Self attention is the key of LLM, used to compute the context of each token.

Fine-Tuning

Fine-tuning is used to correct behavior or add in-domain data to a model, with limited ressources.

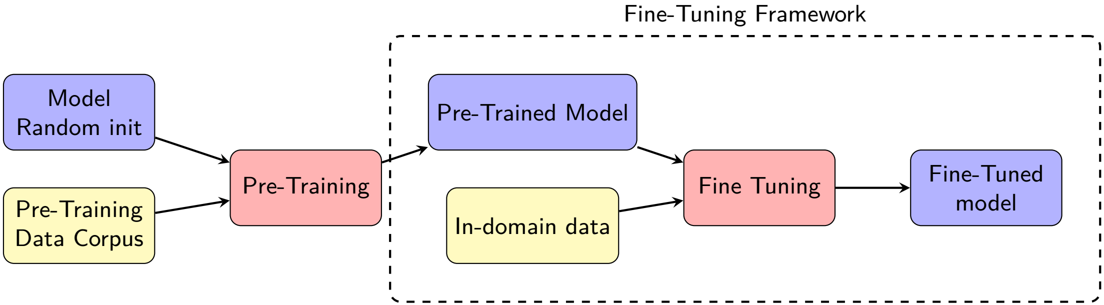


Figure: Pre-training and Fine-tuning generic workflow

Parameters Efficient Fine-Tuning (PEFT)

Set of methods aims to reduce the computation cost of fine-tuning. 2 approaches : *Additive* and **reparametrization**.

Reparametrization

Use lower-cost proxy as trainable weights, and merge at the end. Most famous method : LoRA [9]. These methods are hyperparameter-dependent.

Addivitive

Add part of the model, often linear layer, to train these. One con is to add inference to generation.

Low Rank Adaptation (LoRA)

Principle

Merging Fine-tuning layers with pre-trained ones can be written as $W = W_0 + \Delta W$, with W_0 the pre-trained weights and ΔW the fine-tuned ones.

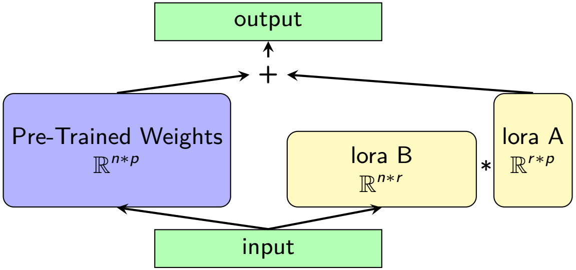


Figure: LoRA Decomposition

LoRA hyperparameters

- rank : the common dimension between A and B.
- alpha : apply a weighting between fine-tuning and pre-trained weights

02
Review of Related Works

Prompt Engineering

Prompt : process of interacting with an artificial intelligence (AI) system by providing specific instructions or queries to achieve a desired outcome.
Example with article [5], when a second LLM is used to modify the prompt.

Pros
Don't need to deal with architecture, weights : act like the LLM is a generating blackbox

Cons
Low impact, locate this work as the end-user, not so much usable

LLM applied to Optimization

Multiples articles show the use of LLM to develop or code optimization algorithms, in particular Evolutionnary Algorithm. One intersting impact is to popularize the development of optimization algorithm.

Pros

Extend the fields of Meta-Heuristics, with new kinds of operators.

Cons

Low impact, don't achieve remarkable performance.

Auto DNN

Sub-domains

- **HyperParameter Optimization(HPO) : Automaticaly define best hyper-parameters, from training to inference**
- Neural Architecture Search(NAS) : Define the best architecture, from scratch or from pruning an existing one

Metrics

- Performance metrics : Accuracy, Latency
- Ressource metrics : inference time, memory usage, energy consumption

Summary

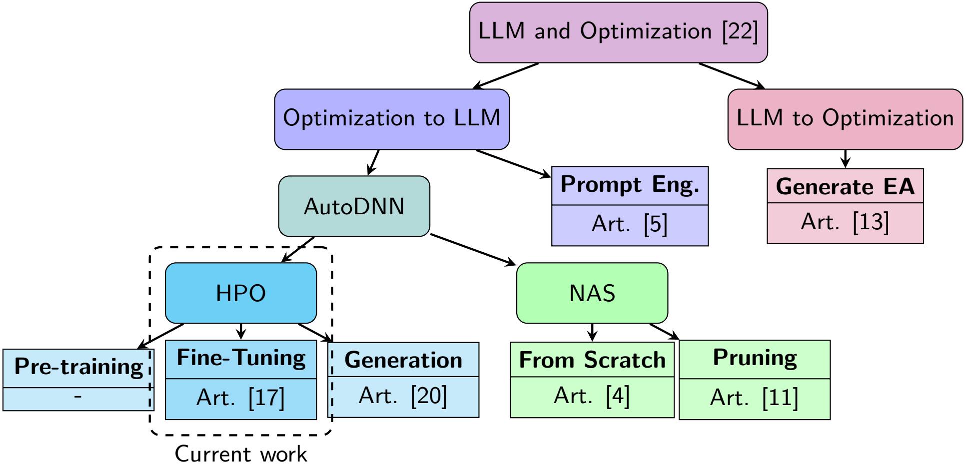


Figure: Summary of links between LLM and Optimization

03
Problem Definition

Problem Definition

This problem can be characterized the optimization of an **expensive, mixed-variable, noisy, blackbox** function.

Problem Formulation

The HPO problem can be defined as

$$\eta^* \in \arg\min_{\eta \in \mathcal{H}} \mathcal{F}(\eta) \tag{1}$$

Search Space $\mathcal{H}$

$\mathcal{H}$ is the set of all values the solution tuple η can take. This stage includes method to handle the mixed-variables aspect of the problems.

Search Strategy

With η_i all the tested solutions, the search strategy is the method used to define the next solution η_{next} to evaluate.

Perfomance Evaluation Strategy

$\mathcal{F}$ represent the objective function, how the function is implemented. Also includes method like multi-fidelity that affect the fidelity of the evaluation.

Search Space

The search space is composed of variables of different type.

Hyper-parameters

Hyper-parameter	Optimization range		Conversion
	Lower Bound	Upper Bound	
Learning Rate	-10	-1	$f(x) = 10^x$
LoRA Rank	2	512	$f(x) = \text{round}(x)$
LoRA scale (α)	1	64	$f(x) = \text{round}(x)$
LoRA Dropout	0	0.5	$f(x) = x$
Weight Decay	-5	-1	$f(x) = 10^x$

Table: Summary of Hyperparameter Search Space

Conversion and naming convention is taken from LitGPT framework.

Search Strategy

The search strategy of an optimization problem can be seen as a balance between the exploration, i.e. going to unexplored regions, and exploitation, i.e. going close to promising areas. Here are the fields of optimization to tackle HPO problems. Standard optimization fields :

- sampling/exploratory : Grid Search, Random Search, Latin Hypercube Sampling
  no exploitation, give a lower bound
- Bayesian Optimization : use surrogate to approximate the objective function, and optimize it.
  weak parallel ability, strong exploitation
- Partition-Based Optimization : FDA, SOO, DiRect
  innate parallel ability, slow convergence

Performance Evaluation Strategy

Evaluation context

In this part, there are many options, like the number of epochs (if not an hyperparameters), the precision of the model, the datasets of training or evaluation.

Objective function

For this problem, there are 2 ways to evaluate a solution :

- Loss (validation or testing) : the loss is computed through the training, and we can keep a small part of the datasets unused to use it the evaluate the model. Cons : dataset dependant, difficult to put in global context
- **Benchmark dataset (GLUE[19], MMLU[7])** : the accuracy on a literature benchmark dataset can be used to evaluate the training. It's interesting, since it's a good measure of generalization, since the model has not read this type of questions. Warning : the benchmark used during the optimization can't be used as a final testing.

Multi-fidelity approaches can be used to reduce the cost of evaluation in earlier steps. Algorithms like Bayesian Optimization and HyperBand (BOHB[2]) achieve cost-efficient optimization by reducing the part of the datasets in early stages.

04
Methodology

Global HPO workflow

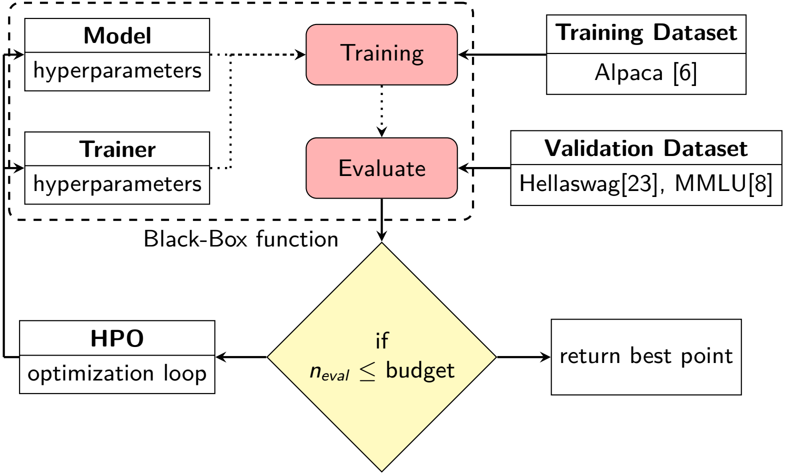


Figure: HPO workflow

Evaluate the solution

Use LitGPT framework with it's CLI to perform an evaluation of a solution. All models and datasets are taken from HuggingFace Hub.

Training

- Model : Llama-3.2-3B
- dataset : Alpaca
- 1 epochs of training
- Fully Sharded Data Parallelism (FSDP) as distributed strategy

Evaluating

Based on lm_eval library

- validation dataset : Hellaswag
- testing dataset : MMLU

Optimization algorithms

Partition Based Algorithm : Simultaneous Optimistic Optimization (SOO)
Perform a K-inary partition of the space, evaluating every center of partition during the expansion of a node.

Algorithm 3.3: SOO

Input: Ω, f, K, $n_{\max}$

// initiate

$x_{0,0} \leftarrow \text{center}(\Omega)$
$f_{0,0} \leftarrow f(x_{0,0})$
$\mathcal{T}_1 \leftarrow \{x_{0,0}, f_{0,0}, \Omega\}$
$n \leftarrow 1$
while $n < n_{max}$ **do**
  $\nu_{\max} \leftarrow -\infty$
  for $h \leftarrow 0$ **to** $depth(\mathcal{T}_n)$ **do**
    $j \leftarrow \arg\max_{j \in \{j|(h,j) \in L_n\}} f(x_{h,j})$ // select function
    if $f(x_{h,j}) > \nu_{max}$ **then**
      $\Omega_{h+1,j+1}, \ldots, \Omega_{h+1,j+K} \leftarrow \text{section}(\Omega_{h,j}, K)$
      for $i \leftarrow 1$ **to** K **do**
        $n \leftarrow n + 1$
        $x_{h+1,j+i} \leftarrow \text{center}(\Omega_n)$
        $f_{h+1,j+i} \leftarrow f(x_{h+1,j+i})$ // Scoring function
        $\mathcal{T}_n \leftarrow \{(x_{h+1,j+i}, f_{h+1,j+i}, \Omega_{n+1})\}$ // add_leaf function
        $\nu_{\max} \leftarrow f_{h,j}$
      end
    end
  end
end
return best of $x_{h,j}, f(x_{h,j})$

Figure: SOO Algorithm

Optimization algorithms II

Surrogate Model Based Optimization : Bayesian Optimization with Gaussian Process (BO-GP)
Use Gaussian Process as a surrogate for the objective function, and optimize it to found the most promising point to evaluate

Algorithm 3.2: BO

Input: Ω, f, K_D, $\mathcal{O}$, f_{acq}, n_{init}, n_{opt}

// initiate function

for $i \leftarrow 1$ **to** n_{init} **do**
  $\lambda' \leftarrow \text{LHS}(\Omega, \mathcal{D})$ // Sample one point
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda', f(\lambda'))\}$ // Add solution and evaluation to set of data
end
for $i \leftarrow 1$ **to** n_{opt} **do**
  $\mu_D, K_D \leftarrow \text{Update}(K_D, \mathcal{D})$
  $K_D \leftarrow \text{Fit}(\text{GP}(K_D), \mathcal{D})$
  $\lambda' \leftarrow \text{Optimize}(f_{\text{acq}}(K_D), \mathcal{O})$ // Generate new point
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda', f(\lambda'))\}$ // scoring function
end
return best of $\{(\lambda^*, f(\lambda^*)) \in \mathcal{D}\}$

Optimization algorithms III

Hybridation : Bayesian Multi-Scale Optimistic Optimization(BaMSOO)
Replace the scoring of SOO with a BO-GP based approximation to determine if it's relevant to evaluate the point.

$$\mathcal{UCB}(x|\mathcal{D}_t) = \mu(x|\mathcal{D}_t) + B_N * \sigma(x|\mathcal{D}_t)$$
$$\text{with } B_N = \sqrt{2\log(\pi^2 N^2/6\eta)}, \eta \in (0,1) \tag{2}$$

Algorithm 3.4: BamSOO scoring

if $\mathcal{UCB}(x_{h+1,j+i}, \mu, \sigma) \geq f^+$ **then**
  $g_{h+1,j+i} \leftarrow f(x_{h+1,j+i})$
  $t \leftarrow t+1$
end
else
  $g_{h+1,j+i} \leftarrow \mathcal{LCB}(x_{h+1,j+i}, \mu, \sigma)$
end
if $g_{h+1,j+i} > f^+$ **then**
  $f^+ \leftarrow g_{h+1,j+i}$
end
$n \leftarrow n+1$
$\mathcal{T}_n \leftarrow \{(x_{h+1,j+i}, f_{h+1,j+i}, \Omega_{h+1,j+i})\}$
return best of $x_{h,j}, g(x_{h,j})$

05
Experiments

Experimental Setup

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).
One evaluation on chuc cluster, using 4*A100 40G of VRAM GPU, is taking around 1 hour. Each algorithms have a budget of 50 evaluations, including the 10 sampling evaluation of BO.

Hellaswag bounds

- Upper bound : best accuracy on Hellaswag : 95.3%. Done with GPT4 model, with 10-shot evaluation
- Lower bound : Sampling without exploitation : 55,7%. Using one-shot LHS, with 10 picks to evaluate.

Sampling experiment : Latin Hypercube Sampling

Objective : explore the search space and make a reference for other algorithms.

Analysis

- Top scores :
  Hellaswag : 47.9%
  MMLU : 37.6%
- High range for Hellaswag, allowing to discriminate efficiently between solutions.

Running time : arround 36 hours

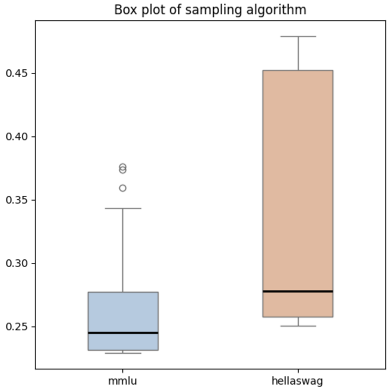


Figure: Distribution of score for sampling experiment

Sampling experiment : Latin Hypercube Sampling II

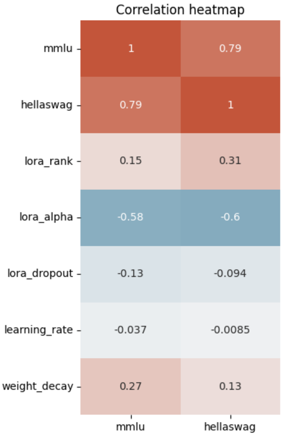


Figure: Correlation between variables and metrics

Correlation between metrics

With 79% of correlation, Hellaswag and MMLU accuracy are relevant as validation/testing metrics.

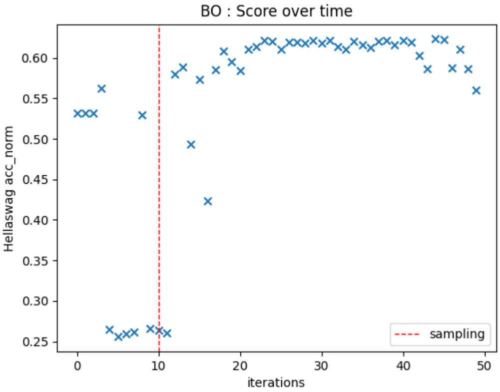
Correlation between variables and metrics

High factor variables : LoRA alpha the Lora rank / weight decay.

TO DO : verify with other experiment the relevance of using dropout and learning rate.

BO

Score evolution

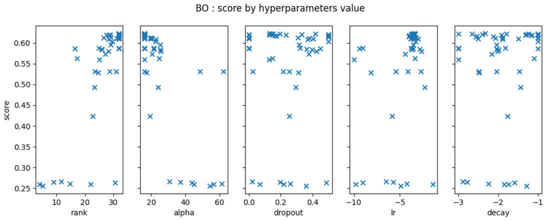


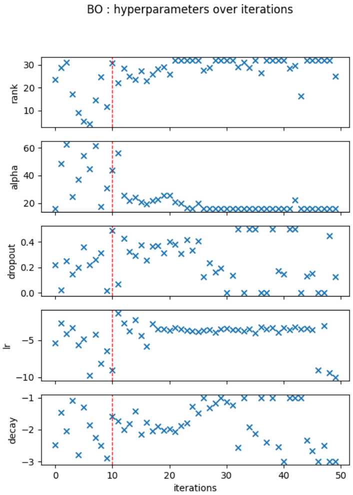
Results

Best score : 62.3%

BO II

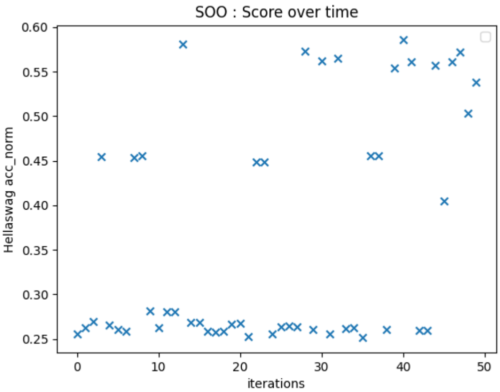
Score by variables and Varibles over iterations





SOO

Score evolution

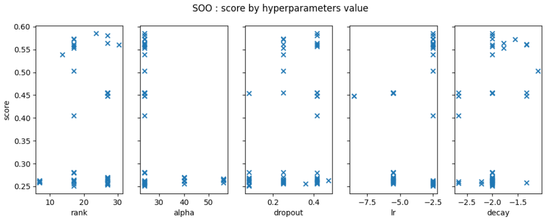


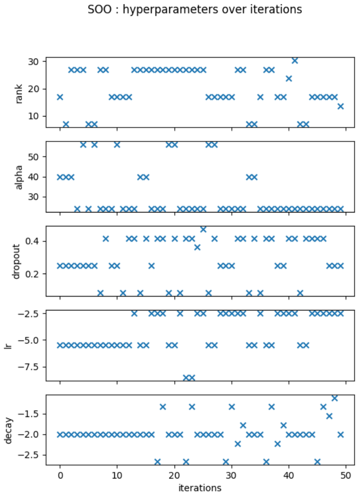
Results

Best score : 58.4%
Slow convergence

SOO II

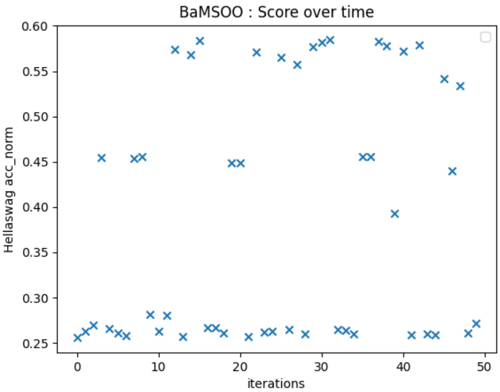
Score by variables and Varibles over iterations





BaMSOO

Score evolution



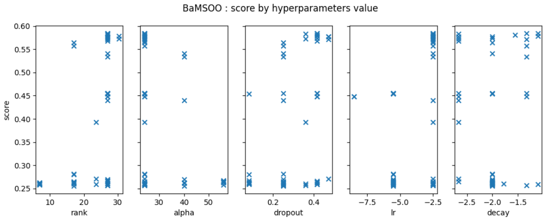
Results

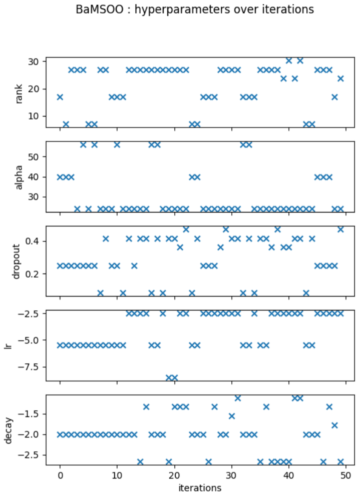
Best score : 58.5%

Not so much approximations, need to increase η in equation 2 to speed the convergence

BaMSOO II

Score by variables and Varibles over iterations





Conclusion

review

On a sequential comparaison, BO-GP algorithms is the most efficient between theses 3 algorithms, even considering the exploitation made by BamSOO algorithms. But this kind of performance needs to efficiently scale to be able to be usable with very expensive function, especially if the evaluation can't be distributed.
With it's acceleration using GP, BaMSOO keep most of the SOO abilities, in particular it's parallelism inate abilities, but achieve to be efficient with a smaller number of evaluation.
To be able to effectively compare theses approaches, it's necessary to look at higher dimensionnal problem.

Perspective

- Expand search space : add dimensions (Adam momentum, precision, matrices to apply LoRA)
- use more training datasets
- make a distributed implementation

Bibliography

[1] Mike Conover et al. *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 (visited on 06/30/2023).

[2] Stefan Falkner, Aaron Klein, and Frank Hutter. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale". In: *CoRR* abs/1807.01774 (2018). arXiv: 1807.01774.

[3] Thomas Firmin and El-Ghazali Talbi. "A fractal-based decomposition framework for continuous optimization". working paper or preprint. July 2022.

[4] Jiahui Gao et al. "AutoBERT-Zero: Evolving BERT Backbone from Scratch". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.10 (June 2022). Number: 10, pp. 10663–10671.

[5] Qingyan Guo et al. *Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers*. arXiv:2309.08532 [cs]. Feb. 2024.

Bibliography II

[6] Rohan Taori and Ishaan Gulrajani and Tianyi Zhang and Yann Dubois and Xuechen Li and Carlos Guestrin and Percy Liang and Tatsunori B. Hashimoto. *Stanford Alpaca: An Instruction-following LLaMA model*. publisher : GitHub. Dec. 2024.

[7] Dan Hendrycks et al. "Measuring Massive Multitask Language Understanding". In: *arXiv preprint arXiv:2009.03300* (2020).

[8] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. arXiv:2009.03300 [cs]. Jan. 2021.

[9] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[10] Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. "Lipschitzian Optimization Without the Lipschitz Constant". In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181.

[11] Aaron Klein et al. "Structural Pruning of Large Language Models via Neural Architecture Search". en. In: (Oct. 2023).

Bibliography III

[12] Guillaume Lample, Hugo Touvron, Lucas Beeching, et al. *LLaMA 3: Open and Adaptable Foundation Models*. https://github.com/meta-llama/llama3. 2024.

[13] Siyi Liu, Chen Gao, and Yong Li. *Large Language Model Agent for Hyper-Parameter Optimization*. arXiv:2402.01881 [cs]. Feb. 2024.

[14] Rémi Munos. "Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness". In: *Advances in Neural Information Processing Systems 24 (NeurIPS)*. 2011, pp. 783–791.

[15] OpenAI. *GPT-4 Technical Report*. https://openai.com/research/gpt-4. 2023.

[16] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[17] Christophe Tribes et al. *Hyperparameter Optimization for Large Language Model Instruction-Tuning*. arXiv:2312.00949. Jan. 2024.

[18] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

Bibliography IV

[19] Alex Wang et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2018, pp. 353–355.

[20] Chi Wang, Xueqing Liu, and Ahmed Hassan Awadallah. "Cost-Effective Hyperparameter Optimization for Large Language Model Generation Inference". en. In: *Proceedings of the Second International Conference on Automated Machine Learning*. ISSN: 2640-3498. PMLR, Dec. 2023, pp. 21/1–17.

[21] Ziyu Wang et al. "Bayesian multi-scale optimistic optimization". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics* 33 (2014), pp. 1005–1013.

[22] Xingyu Wu et al. *Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap*. arXiv:2401.10034. May 2024.

[23] Rowan Zellers et al. *HellaSwag: Can a Machine Really Finish Your Sentence?* arXiv:1905.07830 [cs]. May 2019.

Thank You.