

Hyperparameter Optimization of LLM Fine-Tuning




Bayesian and Partition-based approaches

N. Davouse, E-G. Talbi,
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

01

Introduction



Large Language Models

Summary

- ▶ State-of-the-art of Natural Language Processing (NLP) problems
- ▶ Architecture : Transformers¹ 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**

¹Vaswani et al., « Attention is All you Need ».

Self Attention

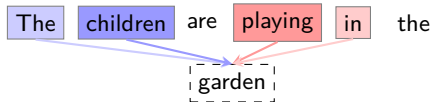


Figure: Self Attention mechanism illustration

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

Fine-Tuning

Following a first phase of pre-training, Fine-tuning is used to correct behavior or add in-domain data to a model, with limited resources.

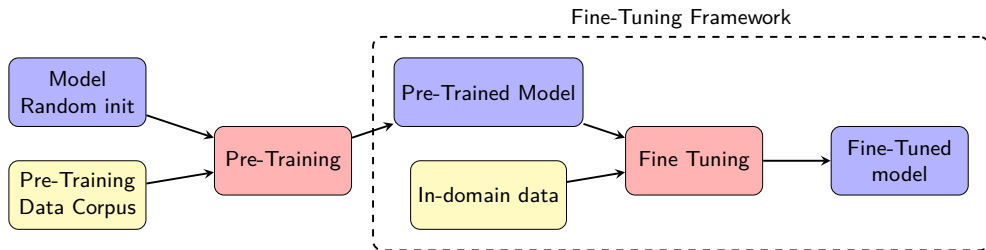


Figure: Pre-training and Fine-tuning generic workflow



Parameters Efficient Fine-Tuning (PEFT)

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

Reparametrization

Use lower-cost proxy as trainable weights, and merge at the end. e.g. : LoRA and derived methods

Quantization

To reduce further the cost of computing during the training, quantization can also be used. This can be combined with either of precedent approaches.

Additive

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. With LoRA, $W = W_0 + \frac{\alpha}{r} B.A$

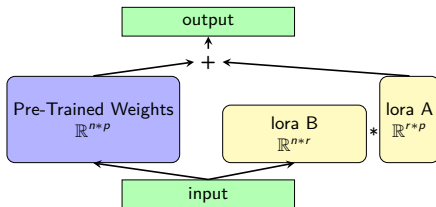


Figure: LoRA Decomposition

LoRA hyperparameters

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



Hyperparameter Optimization (HPO)

Objectives

- ▶ Better performance than manual tuning
- ▶ Ease popularization of the Fine Tuning

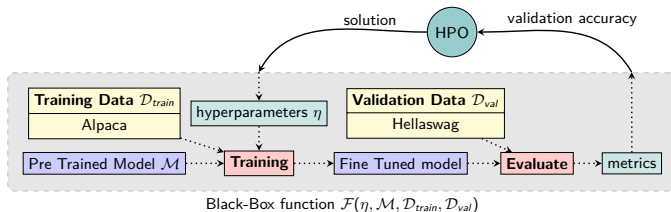


Figure: HPO workflow

Related Works

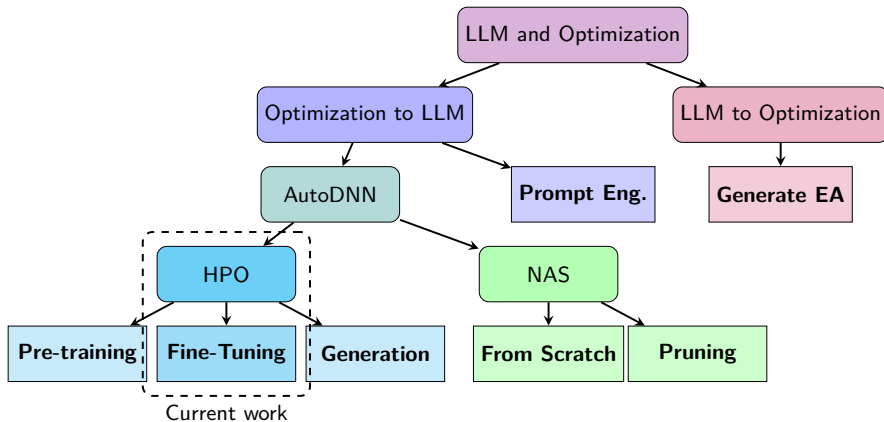


Figure: Summary of links between LLM and Optimization

02

Problem Definition





Problem Definition

Problem Formulation

The HPO problem can be defined as

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

This function can be characterized as an **expensive, mixed-variable, noisy, blackbox** function.

3 phases of an optimization problem

- ▶ **Search Space \mathcal{H}** : all variables and how to handle them
- ▶ **Search Strategy** $\arg \min$: how to search for the minimum of the function
- ▶ **Performance Evaluation Strategy $\mathcal{F}(\cdot)$** : how to evaluate a given solution

Search Space

Hyperparameters

Hyperparameters	Optimization range		Type	Conversion
	Lower Bound	Upper Bound		
Learning Rate	-10	-1	log.	$f(x) = 10^x$
LoRA Rank	1	64	int.	$f(x) = \text{round}(x)$
LoRA scale (α)	1	64	int.	$f(x) = \text{round}(x)$
LoRA Dropout	0	0.5	cont.	$f(x) = x$
Weight Decay	-3	-1	log.	$f(x) = 10^x$

Table: Summary of Hyperparameter Search Space

- ▶ Conversion and naming convention is taken from LitGPT framework.
- ▶ Variable conversion for handling mixed-variables with continuous optimization algorithms
- ▶ No *A-priori* knowledge on hyperparameters importance



Search Strategy

Algorithms for LLM HPO are *Global Optimization* algorithms. Can be classified as :

- ▶ **Exploratory**(GS, Random Search, LHS) : sample the search space
no exploitation, give a lower bound
- ▶ **Metaheuristics** (Genetic Algorithm, ILS, PSO) : bio-inspired heuristics
evaluation greedy, difficult to use for expensive function
- ▶ **Surrogate-Model based Optimization** (Bayesian Optimization with Gaussian Process, TS) :
Use a surrogate to enhance exploitation - innate sequential nature, strong exploitation
- ▶ **Partition-Based Optimization**(FDA, SOO, DiRect) : partition the search space
massively parallel, 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

2 ways to evaluate LLM Fine-Tuning :

- ▶ **Loss (validation/test)** : dataset and model dependant, difficult to compare to other models.
- ▶ **Accuracy on Benchmark dataset (GLUE, MMLU)** : can be used to compare to other models throughout the training.

Complementary approaches

- ▶ **Multi-fidelity** : reduce the cost and the reliability of early evaluations. (ex : BOHB algorithm)

03

Design and Implementation





SMBO : Bayesian-Optimization based on Gaussian-Process (BO-GP)

Principle :

Iterate these two steps over budget :

1. Build a surrogate of the objective function
2. Optimize the surrogate to find the most promising point to evaluate

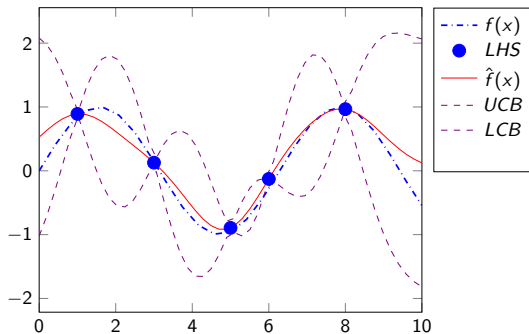


Figure: Illustration of BO-GP Algorithm



SMBO : Bayesian-Optimization based on Gaussian-Process (BO-GP)

Principle :

Iterate these two steps over budget :

1. Build a surrogate of the objective function
2. Optimize the surrogate to find the most promising point to evaluate

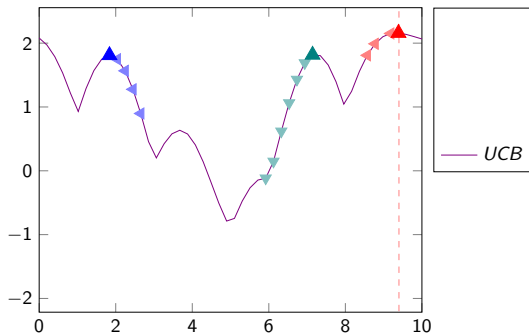


Figure: Illustration of BO-GP Algorithm

PBO : Simultaneous Optimistic Optimization(SOO)

Principle⁶ :

- ▶ K-inary partition of the space
- ▶ Evaluate the center of each partition
- ▶ Expand a maximum of one node by iteration / by depth

⁶Munos, Rémi. « Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness ». 2011

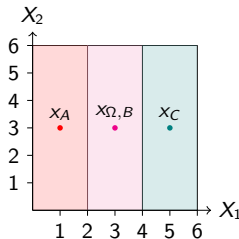


Figure: SOO Partition

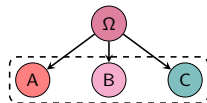


Figure: SOO Tree

PBO : Simultaneous Optimistic Optimization(SOO)

Principle⁶ :

- ▶ K-inary partition of the space
- ▶ Evaluate the center of each partition
- ▶ Expand a maximum of one node by iteration / by depth

⁶Munos, Rémi. « Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness ». 2011

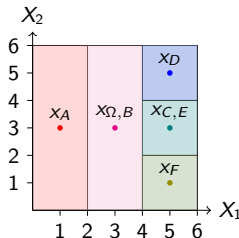


Figure: SOO Partition

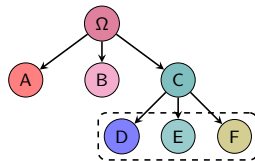


Figure: SOO Tree

PBO : Simultaneous Optimistic Optimization(SOO)

Principle⁶ :

- ▶ K-inary partition of the space
- ▶ Evaluate the center of each partition
- ▶ Expand a maximum of one node by iteration / by depth

⁶Munos, Rémi. « Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness ». 2011

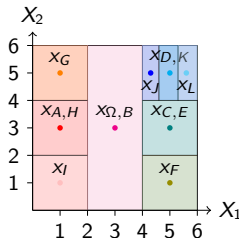


Figure: SOO Partition

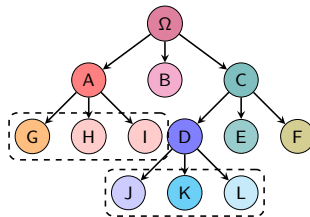


Figure: SOO Tree



Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Principle⁷ :

- ▶ SOO partitionning
- ▶ Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- ▶ If $UCB(x) > f^+$: // x has potential to beat f^+
 - $g(x) = f(x)$ // score x using $f(x)$
- ▶ Else :
 - $g(x) = LCB(x)$ // score x using $LCB(x)$

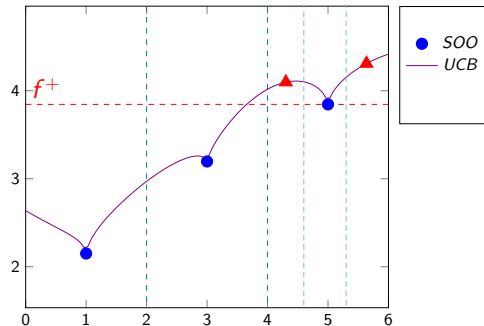


Figure: Illustration of BaMSOO Algorithm

⁷Wang et al. « Bayesian Multi-Scale Optimistic Optimization ». 2014



Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Principle⁷ :

- ▶ SOO partitionning
- ▶ Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- ▶ If $UCB(x) > f^+$: // x has potential to beat f^+
 - $g(x) = f(x)$ // score x using $f(x)$
- ▶ Else :
 - $g(x) = LCB(x)$ // score x using $LCB(x)$

⁷Wang et al. « Bayesian Multi-Scale Optimistic Optimization ». 2014

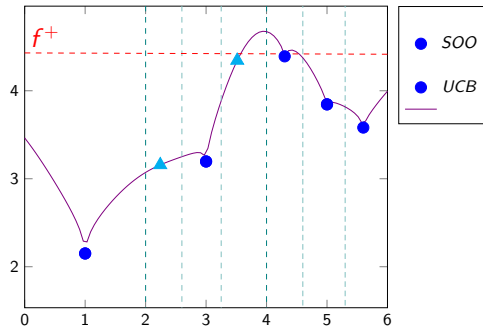


Figure: Illustration of BaMSOO Algorithm



Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Principle⁷ :

- ▶ SOO partitionning
- ▶ Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- ▶ If $UCB(x) > f^+$: // x has potential to beat f^+
 - $g(x) = f(x)$ // score x using $f(x)$
- ▶ Else :
 - $g(x) = LCB(x)$ // score x using $LCB(x)$

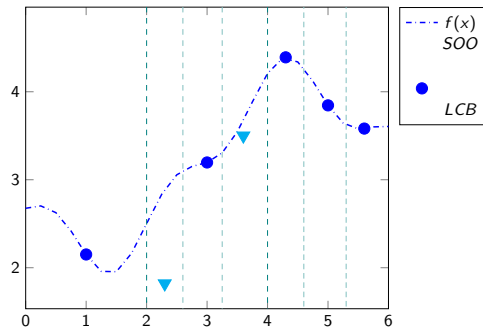


Figure: Illustration of BaMSOO Algorithm

⁷Wang et al. « Bayesian Multi-Scale Optimistic Optimization ». 2014



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-3.2-1B²
- ▶ dataset : Alpaca³
- ▶ 1 epochs of training
- ▶ Fully Sharded Data Parallelism (FSDP) as distributed strategy

²Touvron et al. « LLaMA: Open and Efficient Foundation Language Models », 2023

³Hashimoto et al. « Stanford Alpaca: An Instruction-following LLaMA model ». 2024

Evaluating

Based on lm_eval library


- ▶ validation dataset : Hellaswag⁴
- ▶ testing dataset : MMLU⁵

⁴Zellers et al. « HellaSwag: Can a Machine Really Finish Your Sentence? » 2019.

⁵Hendrycks et al. « Measuring Massive Multitask Language Understanding ». 2021

04

Computation Experiments





Experimental Setup

Experimental testbed

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>).

Hardware and budget allocated

One evaluation on chuc cluster, using 4*A100 40G of VRAM GPU, is taking around 40 minutes. Each algorithms have a budget of 50 evaluations, including the 10 sampling evaluation of BO.

Sampling experiment : Latin Hypercube Sampling

Objective : Explore the space and define a lower bound for next experiments

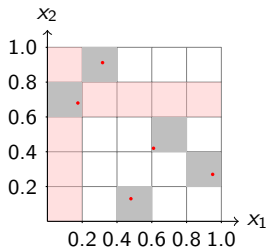


Figure: LHS illustration with $g = 5$ samples

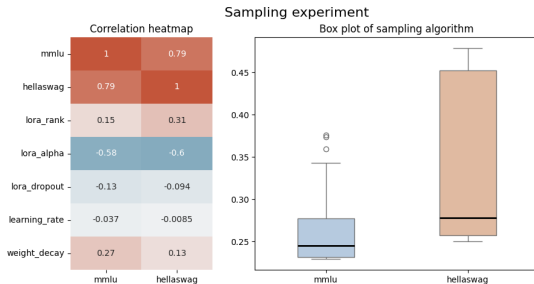


Figure: Results of LHS Experiment⁵

⁵At left : correlation between metrics and hyperparameters // At right : metrics distribution



Results : Bayesian Optimization

Score over time

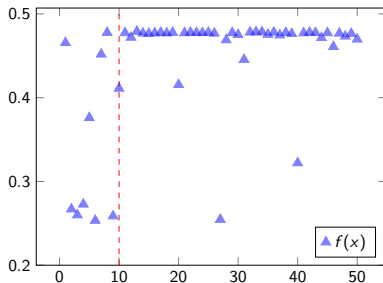


Figure: Results on validation dataset for BO-GP Algorithm

Results

Hellaswag(D_{val}) Best score : 47,91%

Behavior

- Fast convergence after sampling phase
- Few shots emphasizing exploration with lower score



Results : SOO

Score over time

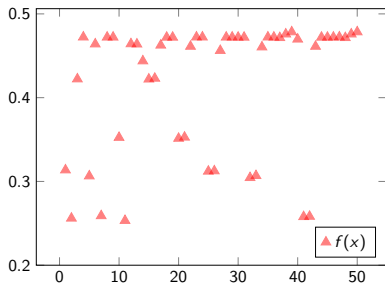


Figure: Results on validation dataset for SOO Algorithm

Results

Hellaswag(D_{val}) Best score : 47.84%

Behavior

- ▶ A lot of low score evaluation, due to one hyperparameter
- ▶ maximum depth of 8 (only 2 points with depth = 8)



Results : BaMSOO

Score over time

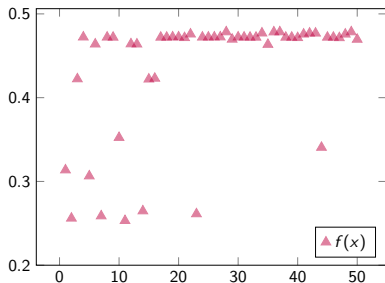


Figure: Results on validation dataset for BaMSOO Algorithm

Results

Hellaswag(D_{val}) Best score : 47.84%
Do not achieve to overperform SOO best score

Behavior

- ▶ Prevent SOO unpromising evaluation (16 approximated evaluations)
- ▶ maximum depth of 8 (8 points with depth = 8)



Comparison and analysis

Analysis

- ▶ Upper Bound on Hellaswag is irrelevant
- ▶ Only BO-GP beat LHS
- ▶ with more high-performing solution, BaMSOO overperform SOO on MMLU

Results

	Hellaswag	MMLU
Lower Bnd ¹	47.90	37.61
Upper Bnd ²	41.5	49.3
BO-GP	47.91	38.11
SOO	47.84	37.42
BaMSOO	47.84	37.50

Table: Bound and best score for datasets (val and test)

1 : LHS results; 2 : Meta Fine tuning

Score over time on testing dataset

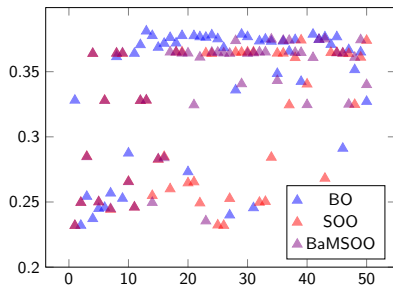


Figure: Results on testing dataset for the three algorithms

05

Conclusions and Perspectives



Conclusions

- ▶ With limited budget, BO-GP demonstrated exceptional efficiency, quickly converging to high-performing solutions.
- ▶ BaMSOO succeeded to hasten SOO convergence by preventing unpromising evaluations.
- ▶ Lays the groundwork for using PBO algorithms, enhance with BO-GP to deal with LLM Fine-Tuning

Extend experiments on the application

- ▶ Extend search space (add hyperparameters, bigger range...)
- ▶ Use more models/datasets
- ▶ Make a distributed implementation

Generalization of BaMSOO Hybridization

- ▶ explore parallelism of BO-GP based algorithms
- ▶ Global Framework of *Parallel Bayesian-enhanced Fractals Optimization*

Thank You.

