Project 25: Measuring Optimizer-Agnostic Hyperparameter Tuning Difficulty

NEW YORK UNIVERSITY

Mentors: Nicholas Lourie, Kyunghyun Cho, He He Group 18: Zhanyi Ding, Jinyang Liu, Tina Wan, Karen Fan

Motivations

How can we compare two ML models when considering hyperparameter tuning algorithms?

When we are proposing new ML models, we usually compare the performances in terms of benchmarking dataset metrics and tuning curves. However, one could argue that the model could have better/worse performances under a different hyperparameter tuning algorithm. One natural thought would be to standardize two models under one simple tuning algorithm such as random search.

What are the conditions that would enable such standardization?

If ML models have an intrinsic notion of hyperparameter tuning difficulty, i.e., the tuning difficulty is a property of ML models instead of the tuning algorithm, then we could safely use a simple tuning algorithm such as random search as baseline to compare each model.

Related Work

- Show Your Work: Improved Reporting of Experimental Results
  Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019
  Highlighted the inadequacy of relying solely on test-set performance scores for accurate model performance assessments. Proposed a closed form formula for expected validation performance of the best-found model as a function of computation budget.
- Tunability: Importance of Hyperparameters of Machine Learning Algorithms
  Philipp Probst, Bernd Bischl, and Anne-Laure Boulesteix. 2018.
  Formalize the problem of tuning from a statistical point of view, defined data-based defaults and suggest general measures quantifying the tunability of hyperparameters of algorithms.
- Importance of Tuning Hyperparameters of Machine Learning Algorithms
  Hilde J. P. Weerts, Andreas C. Mueller, and Joaquin Vanschoren. 2020.
  Presented a methodology to determine the importance of tuning a hyperparameter based on a non-inferiority test and tuning risk: the performance loss that is incurred when a hyperparameter is not tuned, but set to a default value.
- Optimizer Benchmarking Needs to Account for Hyperparameter Tuning
  Prabhu Teja Sivaprasad, Florian Mai, Thijs Vogels, Martin Jaggi, and François Fleuret. 2020.
  Addressed the crucial aspect of computational cost in assessing optimizer performance, and argued for fair evaluations by incorporating hyperparameter tuning expenses. Suggested Adam as a practical solution, particularly in low-budget scenarios, acknowledging the pragmatic challenges of optimizing under resource constraints.

While these studies collectively underscore the importance of hyperparameter tuning and advocate for a holistic evaluation considering computational costs, none delve into the inherent difficulties associated with model tuning. This apparent gap presents an opportunity for our project to contribute novel insights into the intricacies of the model tuning process.

Experiment Design

Models and Dataset

1. **ResNet**, Convolutional Neural Network with residual connections. Specifically, we used ResNet9, which consists of 5 convolutional blocks, 3 residual connections and 1 output MLP layer. It is trained on CIFAR-100, which is a image classification benchmarking dataset with 100 classes. Model is evaluated by classification accuracy.
2. **NanoGPT**, a character-level GPT with a context size of up to 256 characters, 384 feature channels, and it is a 6-layer Transformer with 6 heads in each layer. It is trained on **wikitext-103 dataset**, a collection of over 100 million tokens extracted from the set of verified Good and Featured articles on Wikipedia. This model is evaluated by perplexity.

Tuning Algorithms

1. **Random Search**, a stochastic approach for hyperparameter optimization. It operates by randomly selecting combinations of hyperparameters from a predefined range.
2. **Bayesian Optimization**, An advanced method using Gaussian Processes to model the mapping of hyperparameters to performance metrics. It is implemented using Sequential Model-Based Algorithm Configuration (SMAC), which starts with 8 iterations of Random Search for initial exploration, followed by 24 iterations of Bayesian Optimization, refining the search in promising regions of the hyperparameter space.

Experiment Results

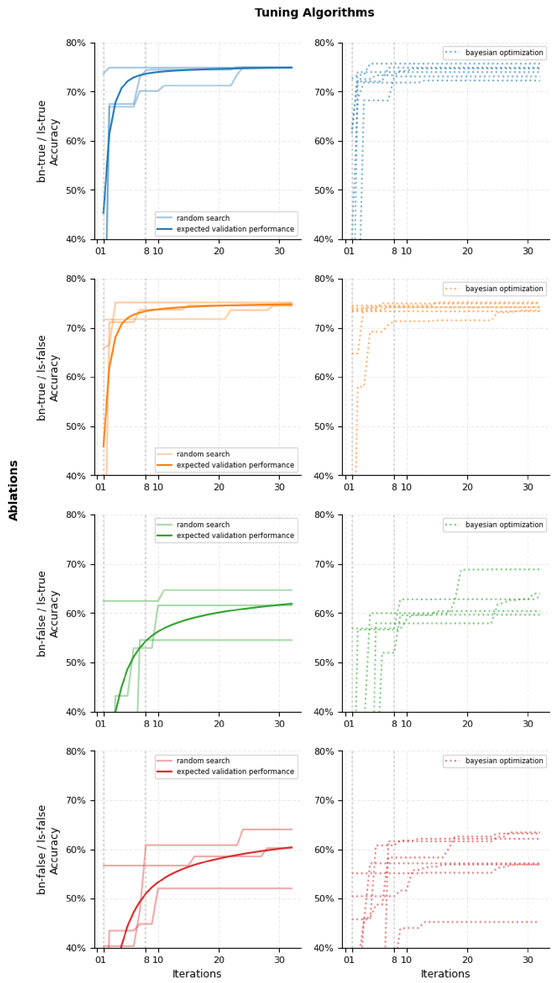

Figure 1. Best Validation Accuracy v.s. # of Iterations for ResNet9

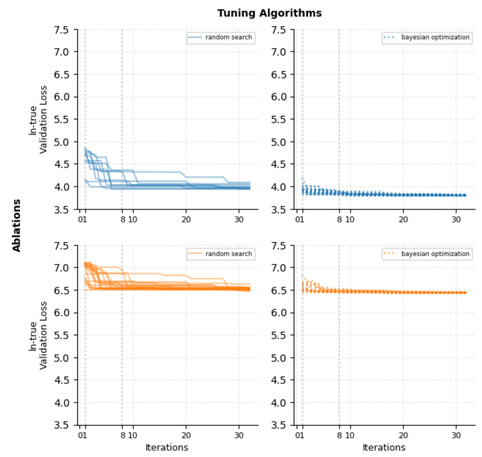

Figure 2. Best Validation Loss v.s. # of Iterations for NanoGPT

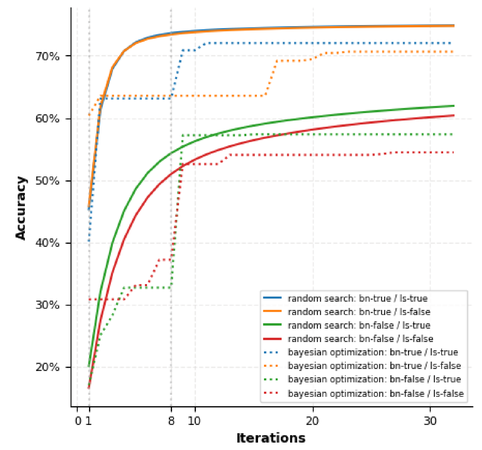

Figure 3. Averaged Accuracy v.s. # of Iterations for ResNet9

Experiment Design Continued - Ablations

Label smoothing is a regularization technique used in classification tasks to introduce noise for labels. It involves modifying the target labels of the training data by distributing some probability mass from the true label to other incorrect labels, which accounts for potential mistakes in dataset.

Batch/Layer normalization is a technique used to improve neural network training and performances. By normalizing the inputs to each layer, batch/layer normalization helps stabilize and speed up the training process.

These two parts are considered important components of the model, and removing any one or combination of them would impact the structure and performance of the model, thus changing the tuning difficulty. Our ablation on ResNet is a 4-fold combination of with and without label smoothing and batch norm. While for NanoGPT, it is 2-fold with or without layer norm.

Experiment Results Continued

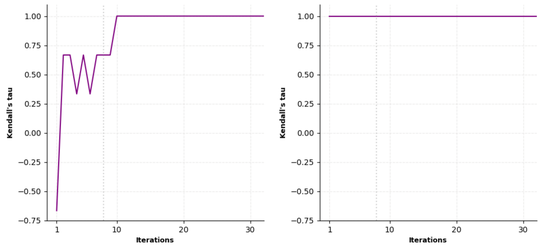

Figure 4. Kendall's tau for ResNet9

Figure 5. Kendall's tau for NanoGPT

In figure 1, horizontally, models are under the same ablation setting, and vertically the models are using the same hyperparameter tuning algorithm. There are two key elements reflected in the tuning curve: how many iterations required to reach optimal performances, and at what level would the optimal accuracy be at. The translucent lines represent direct outcomes from experiments. Acknowledging the limited experiments in the context of random search, we employed the mean tuning curve [1], depicted in solid, to estimate the expected accuracy over tuning iterations. As we could see from the plot, the tuning curves tend to stay in similar shape if they are under the same ablation setting: i.e. whether they preserve batch norm and label smoothing, regardless of the tuning algorithms used. Such behavior indicates that the tuning difficulty of the models, are independent of the hyperparameter tuning algorithms used, but significantly influenced by the model architecture.

Similar behavior could be observed from plot2, which comes from the NanoGPT experiment. Since the evaluation here is validation loss, the tuning curve is therefore decreasing. We can still observe that for the same ablation setting, two tuning algorithms would have similar shape of tuning curve, yet for different ablation settings the tuning curve vary a lot in terms of the peak performances as well as the iterations needed to get there.

We conducted an additional assessment using the Kendall ranking correlation coefficient, a metric that gauges the strength and direction of the association between two ranked variables, with values ranging from -1 to 1, signifying a reversed rank to the same rank. In this context, we ranked model ablations for ResNet9/NanoGPT based on the best validation accuracy/loss after each iteration round for each tuning algorithm. Comparisons were made between the two rankings across various numbers of iterations. Intriguingly, we observed consistent rankings for NanoGPT ablations across all tuning algorithms. Conversely, rankings for ResNet9 ablations initially exhibited a reversal but swiftly stabilized after 10 iterations. This observation underscores that distinct tuning algorithms do not exhibit a preference for specific model configurations, and the variance in the impact of tuning algorithms remains consistent across various model ablations.

[1] Nicholas Lourie, Kyunghyun Cho, and He He.
Show your work with confidence: Confidence bands for tuning curves, 2023.

Conclusion and Future Work

As our empirical experiment shows, machine learning models have an intrinsic notion of hyperparameter tuning difficulty, and models don't have preferences for hyperparameter tuning algorithms. Therefore, when proposing new models, researchers could compare the performance of two different models under a simple tuning algorithm such as random search, which standardizes the process, saves lots of computational resources and eliminates the bias of different tuning algorithms.

However, our work is highly empirical and only tested out two models with ablations and two hyperparameter tuning algorithms. The future work could be two-fold:

1. It would be helpful to test out more models in different tasks and more tuning algorithms to confirm the hypothesis that the intrinsic notion of hyperparameter tuning difficulty exists in every model.
2. Developing a theoretical framework could help understand why certain models present more tuning challenges, guiding future model development and optimization.