AI VIET NAM – RESEARCH TEAM

Towards Learning Universal Hyperparameter Optimizers with Transformers

December 24, 2022

1. **Purpose/outputs:**

   - **OPTFORMER** [1] is a general hyperparameter optimization (HPO) framework based on Transformers [9]. Using a Transformer as a universal interface for modelling experimental data and learn HPO algorithms given a sufficient amount of data, Transformer can potentially learn a more complex prior distribution than standard Bayesian Optimization.
   - Dealing with large datasets consisting of experimental trials from reality can be challenging, due to large variations in HPO problems and their associated text metadata. Moreover, most meta and transfer-learning HPO methods consider a restrictive setting where all tasks must share the same set of hyperparameters. The authors adopt a supervised learning approach, by learning to predict parameters and hyperparameter response functions from offline tuning data (see Figure 1).

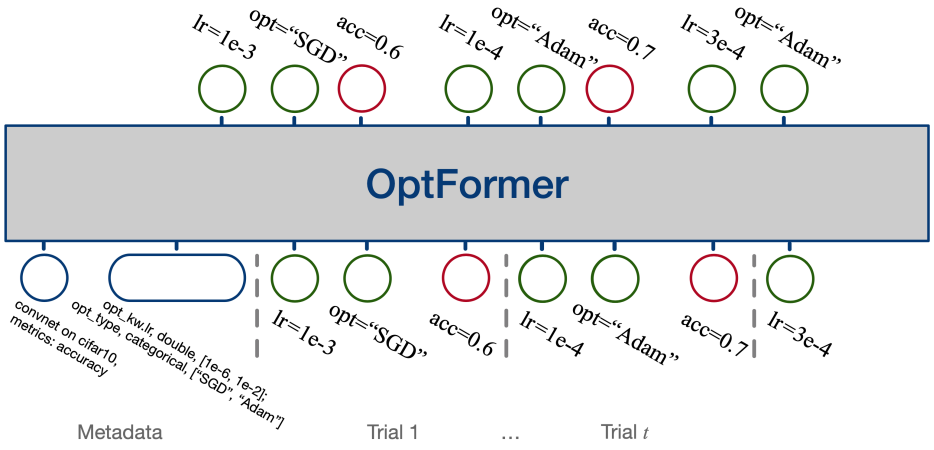


Figure 1: Illustration of the OPTFORMER model over a hyperparameter optimization trajectory. It is trained to predict both hyperparameter suggestions (in green) and response function values (in red).

2. **Contributions**

   - OptFormer makes use of data produced by Google Vizier [2], an internal default platform for hyperparameter optimization and overcome drawbacks of previous methods for meta-learning over such data, which includes heavy dependence on numerical constraints and rare usage of textual information.
   - OptFormer is one of the first Transformer-based frameworks for hyperparameter tuning, learned from large-scale optimization data using flexible text-based representations.

3. **Methodology:**

- **Meta-learning for hyperparameter optimization**
  HPO aims to find a set of hyperparameters x from search space $\mathcal{X}$ to maximize a model performance metric, $y = f(x)$. The goal of the meta-learning [3] approach for HPO is to learn the shared knowledge among the objective functions f from a dataset of multiple tuning experiments represented as studies and to obtain an optimal HPO algorithm for new hyperparameter tuning tasks from a similar distribution to those in the dataset.

- **Representing studies as Tokens**
  OPTFORMER takes textual information of study data, including learning rate, optimizer type and accuracy then using concepts from natural language to represents all of the study data as a sequence of tokens. The OptFormer generates new hyperparameters, predicts the task accuracy, and finally receives the true accuracy, which will be used to generate the next round's hyperparameters. The authors train a 250M parameters T5 [6] on approximately 200M past training runs from Google's internal hyperparameter optimization service.

- **Inputs tokenization**
  All scalar hyperparameters are uniformly quantized to 1000 integer values. Concretely,

$$\bar{x} = \text{int}[x_{norm}.Q], \text{where} x_{norm} = (x - x_{min})/(x_{max} - x_{min}) \text{ and } Q = 1000$$

- **Intimating policies**
  OPTFORMER is trained to predict the next trial in the sequence, which was originally chosen by another HPO algorithm. In other words, the reinforcement learning problem of sequentially choosing the parameters has now become the supervised imitation learning and greedy selection. The OPTFORMER will imitate the algorithm's behavior given text-based prompt in the metadata, as can be seen in Figure 2.

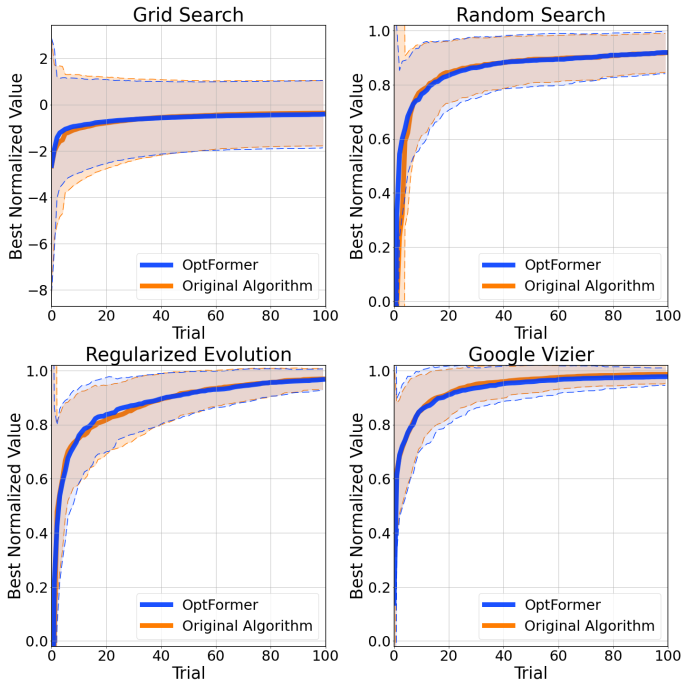
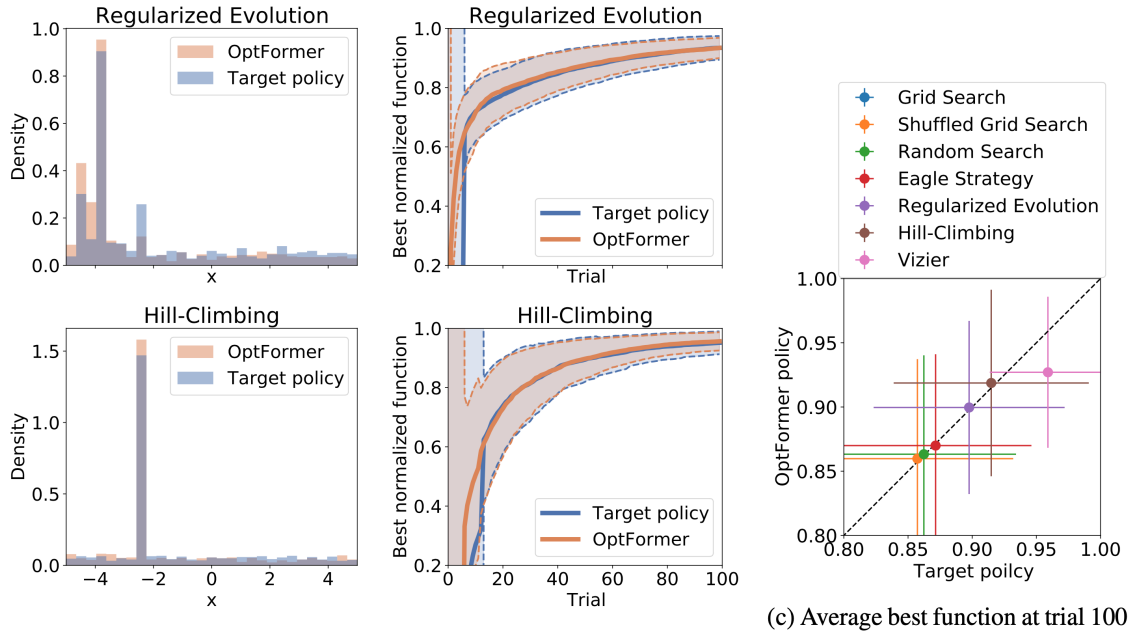


Figure 2: Over an unseen test function, the OptFormer produces nearly identical optimization curves as the original algorithm. Mean and standard deviation error bars are shown.

4. **Results:**

- As can be seen in Figure 3, OptFormer imitates most algorithms very accurately in both the mean and variance except for the most complicated algorithm.



(a) Policy distribution (b) Optimization trajectory (c) Average best function at trial 100 with standard deviation.

Figure 3: Comparing the performance of different algorithms outputted by the OPTFORMER conditioned on the corresponding algorithm's name.

- The model is capable of predicting objective values very accurately, in many cases surpassing Gaussian Processes, which are commonly used in algorithms such as Bayesian Optimization.

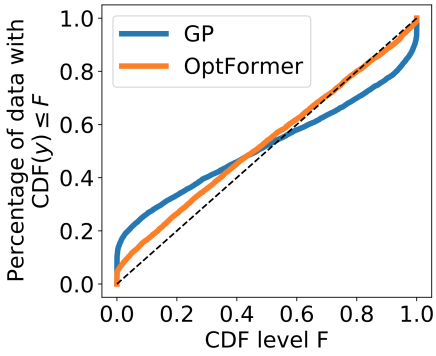


Figure 4: Cumulative histogram of predicted $CDF(y)$ on RealWorldData test set.

- The resulting OPTFORMER (EI) outperforms all baselines across the board on both benchmarks when the prior policy is augmented with the Expected Improvement acquisition function.

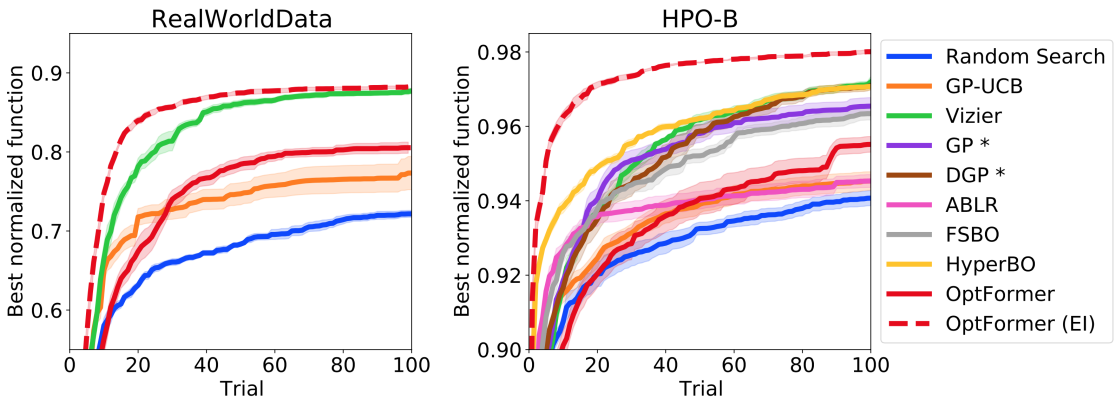


Figure 5: Best normalized function value averaged over 16 RealWorldData test functions (left) and over 86 HPO-B test functions (right) with 1-std confidence interval from 5 runs.

5. **Limitations**

- Parameters that do not always apply or are subject to dynamic constraints depending on other parameter values are not taken into consideration.
- Only sequential optimization with a batch size of one are considered.
- The authors considered a single objective function while multiple objectives can be easily included by outputting multiple function tokens in a trial.
- The maximum sequence length is limited by the memory size requirement of a Transformer.

6. **Notes**

- Organizations that have trained models will have an advantage when training new ones, suggesting wider adoption of model training services, rather than doing it manually.
- In the near future, model training could perhaps consolidate around a few shared services, and rapidly improve in both scale and scope, thanks to ever-larger HPO models.

References

[1] Y. Chen, X. Song, C. Lee, Z. Wang, Q. Zhang, D. Dohan, K. Kawakami, G. Kochanski, A. Doucet, M. Ranzato, S. Perel, and N. de Freitas. Towards learning universal hyperparameter optimizers with transformers, 2022.

[2] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. E. Karro, and D. Sculley, editors. *Google Vizier: A Service for Black-Box Optimization*, 2017.

[3] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey, 2020.

[4] Y. Li, Y. Shen, H. Jiang, W. Zhang, J. Li, J. Liu, C. Zhang, and B. Cui. Hyper-tune: Towards efficient hyper-parameter tuning at scale, 2022.

[5] D. Navon and A. M. Bronstein. Random search hyper-parameter tuning: Expected improvement estimation and the corresponding lower bound, 2022.

[6] A. Roberts, C. Raffel, and N. Shazeer. How much knowledge can you pack into the parameters of a language model?, 2020.

[7] S. Shekhar, A. Bansode, and A. Salim. A comparative study of hyper-parameter optimization tools, 2022.

[8] M. Tuli, M. S. Hosseini, and K. N. Plataniotis. Towards robust and automatic hyper-parameter tunning, 2021.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[10] H. Wang, A. Sakhadeo, A. White, J. Bell, V. Liu, X. Zhao, P. Liu, T. Kozuno, A. Fyshe, and M. White. No more pesky hyperparameters: Offline hyperparameter tuning for rl, 2022.

[11] L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, nov 2020.

[12] T. Yu and H. Zhu. Hyper-parameter optimization: A review of algorithms and applications, 2020.