Tune My Adam, Please!

Theodoros Athanasiadis[1] **Steven Adriaensen**[1] **Samuel Müller**[1,2] **Frank Hutter**[1,3,4]

[1]University of Freiburg, Freiburg, Germany
[2]Meta, New York City, USA
[3]ELLIS Institute Tübingen, Tübingen, Germany
[4]Prior Labs, Freiburg, Germany

Abstract The Adam optimizer remains one of the most widely used optimizers in deep learning, and effectively tuning its hyperparameters is key to optimizing performance. However, tuning can be tedious and costly. Freeze-thaw Bayesian Optimization (BO) is a recent promising approach for low-budget hyperparameter tuning, but is limited by generic surrogates without prior knowledge of how hyperparameters affect learning. We propose `Adam-PFN`, a new surrogate model for Freeze-thaw BO of Adam's hyperparameters, pre-trained on learning curves from TaskSet, together with a new learning curve augmentation method, CDF-augment, which artificially increases the number of available training examples. Our approach improves both learning curve extrapolation and accelerates hyperparameter optimization on TaskSet evaluation tasks, with strong performance on out-of-distribution (OOD) tasks.

1 Introduction

Hyperparameter Optimization (HPO) can be costly and time-consuming, especially in the age of LLMs. Although Bayesian Optimization (BO) is a sample-efficient alternative, it still requires training a model in full to get a new evaluation point for the BO loop.

Freeze-thaw BO (FT-BO) (Swersky et al., 2014) is a more efficient alternative that allocates resources to the most prominent configurations one step (e.g., one epoch) at a time. Additionally, rather than disregarding a configuration, FT-BO keeps it frozen in memory and can decide whether to thaw that configuration or start a new one.

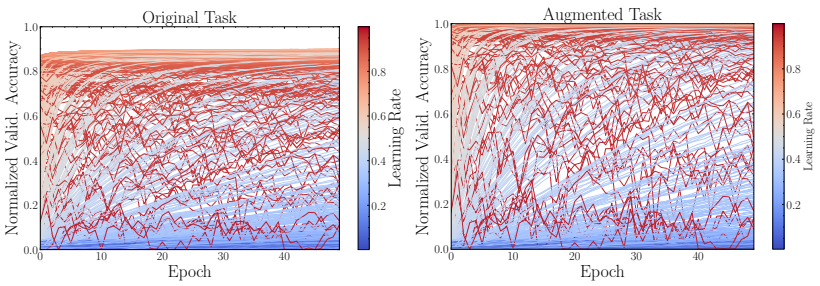
Recently, Rakotoarison et al., 2024 introduced `ifBO`, a new, state-of-the-art framework for FT-BO. It consists of `FT-PFN`, a PFN (Müller et al., 2022) based surrogate that performs Bayesian learning curve extrapolation, trained on synthetic hyperparameter (HP) configurations and learning curve combinations. However, `ifBO` attempts to fit every HPO case with one surrogate model.

Our work introduces a drop-in replacement surrogate model for `ifBO`, `Adam-PFN`, specialized to tune the HPs of the Adam optimizer (Kingma and Ba, 2015). We also introduce a learning curve augmentation method that allows us to train on real data from TaskSet (Metz et al., 2020), a collection of learning curves optimized using Adam, instead of sampling them from a synthetic prior.

2 Adam-PFN

2.1 Real Data: TaskSet

TaskSet offers a collection of 1162 learning curves on a variety of tasks, including language modeling on words, subwords, and characters with RNNs, text classification with RNNs, CNNs trained on image data, and MLPs, all optimized with Adam. Each task contains the learning curves of the same 1000 randomly sampled HP configurations, including the learning rate, β_1, β_2, ϵ, two parameters that control L_1 and L_2 regularization, and two parameters that control the learning rate decay schedule. All hyperparameter values are uniformly sampled on a log scale. Table 2 in Appendix A gives a more detailed overview of the hyperparameters' lower and upper bounds.



(a) (b)

Figure 1: (a) A selection of learning curves from TaskSet, the color of each curve is based on the learning rate value. (b) The same set of learning curves augmented with CDF-augment. CDF-augment non-linearly transformed "task hardness" by making the task easier.

2.2 Augmentation Method: CDF-augment

To generate an even more diverse training set, we introduce CDF-augment, a local learning curve augmentation method that uses the cumulative distribution function (CDF) of the Beta distribution.

To augment a learning curve, we first sample Beta's mode μ uniformly in $[0, 1]$ and concentration κ uniformly in $[2, 5]$, and calculate the CDF. We then forward-propagate the learning curve y through the CDF as follows

$$y' = F_{Beta}(y; \mu, \kappa) \tag{1}$$

Using this augmentation, we increase the learning curve diversity by non-linearly transforming task-hardness. Due to the nature of the CDF, the rank (ordering) of learning curves is preserved. Figure 1 shows the result of applying CDF-augment on a set of learning curves.

2.3 Surrogate Model Training

Our surrogate model is a PFN, trained on augmented data from real learning curves of the Adam optimizer, similar to the data it will encounter in practice. PFNs are based on the transformer architecture (Vaswani et al., 2017), and leverage in-context learning to perform Bayesian learning curve extrapolation in a single forward pass.

We follow the same training pipeline for our surrogate model as `FT-PFN`, the only difference is that we do not sample the learning curves with and their HP configurations from a synthetic prior, rather, we train on real curves from TaskSet, which we augment using CDF-augment. We include more implementation details in Appendix B.

Starting with a set of learning curves randomly sampled from 878 tasks and 1000 HP configurations per task, we augment and then split each curve into context and query points. We train our model to extrapolate the performance of a curve at a query point, given the context and HP configuration. More details on the training tasks can be found in Appendix C.

3 Experiments

3.1 Evaluation Procedure

Our evaluation set consists of 12 NLP tasks from TaskSet, which were not used during training and are also used in prior work (Kadra et al., 2023), (Wistuba et al., 2022), (Rakotoarison et al., 2024). A list of the evaluation tasks is included in Appendix D.

To evaluate the quality and runtime complexity of predictions, we compare our approach against `DyHPO` (Wistuba et al., 2022), `DPL` (Kadra et al., 2023), `FT-PFN` (Rakotoarison et al., 2024), and a Uniform Predictor that outputs a uniform distribution over the $[0, 1]$ range for Log-likelihood and a constant value of 0.5 for MSE.

Algorithm	Context 400			Context 1000			Context 1600		
	LL	MSE	Time	LL	MSE	Time	LL	MSE	Time
Adam-PFN (CDF)	**5.326**	**0.00054**	0.697	**5.422**	**0.00046**	2.151	**5.441**	**0.00043**	4.014
Adam-PFN (Mixup)	4.916	0.00080	0.677	4.986	0.00067	2.131	5.022	0.00061	3.972
Adam-PFN (No aug.)	4.947	0.00062	0.671	5.042	0.00054	2.090	5.066	0.00052	3.993
FT-PFN	3.440	0.00184	0.693	3.473	0.00171	2.103	3.443	0.00164	4.017
DPL	-29.752	0.00191	17.790	-22.192	0.00175	38.855	-16.154	0.00164	63.748
DyHPO	-0.494	0.00336	25.662	-0.397	0.00317	113.217	-0.383	0.00317	264.284
Uniform	-6.908	0.22881	N/A	-6.908	0.22852	0.000	-6.908	0.22896	N/A

Table 1: Comparison of `Adam-PFN(CDF)`, `Adam-PFN(Mixup)`, and `Adam-PFN (No Aug.)` against the baselines for different context sizes. Values are the median across the evaluation tasks.

We also evaluated the HPO performance of our surrogate as part of `ifBO` by replacing `FT-PFN` while keeping the proposed acquisition function. For HPO performance, we additionally compare against `HyperBand` (Li, Jamieson, DeSalvo, et al., 2018), `ASHA` (Li, Jamieson, Rostamizadeh, et al., 2020), `Freeze-thaw with GPs` (Swersky et al., 2014), and `Random Search` (Bergstra and Bengio, 2012). We include additional details on the baselines and discuss how they relate to our work in Appendix E.

We also included two variants of Adam-PFN, "No Aug" and "Mixup". `Adam-PFN (No aug.)` was trained on the initial TaskSet tasks, without any learning curve augmentation, while `Adam-PFN (Mixup)` was trained using the recently introduced Mixup learning curve augmentation method (Lee et al., 2024). Further details on Mixup are included in Appendix F.

3.2 Learning Curve Extrapolation Results

Table 1 presents the learning curve extrapolation results. `Adam-PFN (CDF)` outperforms all other baselines, both in terms of Log-likelihood (LL) and MSE. As expected, the inference time of all PFN approaches is significantly lower than that of DPL and DyHPO, since they are trained offline.

3.3 HPO Results

Figure 2 shows the aggregated normalized regret and the average rank of each algorithm. `Adam-PFN (CDF)` outperforms all baselines both in terms of normalized regret and average rank. The other two alternatives, `Adam-PFN (Mixup)` and `Adam-PFN (No. aug)`, perform similarly well. All variants are also sample-efficient, achieving the same normalized regret at approximately epoch 150, as `FT-PFN` does at approximately epoch 750.

3.4 Evaluation In The Wild: Out-Of-Distribution Tasks

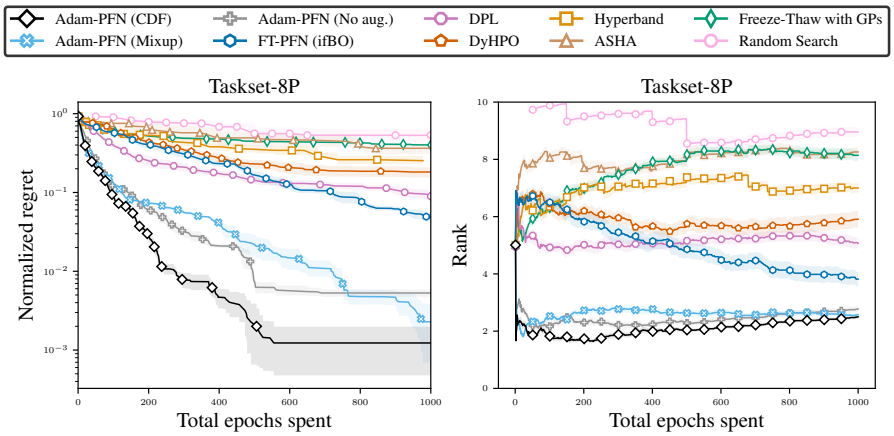
We also evaluated the HPO performance of all `Adam-PFN` variants on some real-world tasks from the publicly available PyTorch Examples[1] repository. We compare our approach with `FT-PFN`, HyperBand, ASHA, and Random Search.

The results are presented in Figure 3. Focusing on subfigure (b), which shows the average ranks, we observe that `Adam-PFN (CDF)` is the best-performing alternative early on, on average. As the budget increases, `FT-PFN` catches up and eventually surpasses our model. An approach that warm-starts the HPO process with `Adam-PFN` and then switches to `FT-PFN` might be optimal for these tasks, but we leave that as future work. We include more details and results in Appendix G.

4 Limitations & Future Work

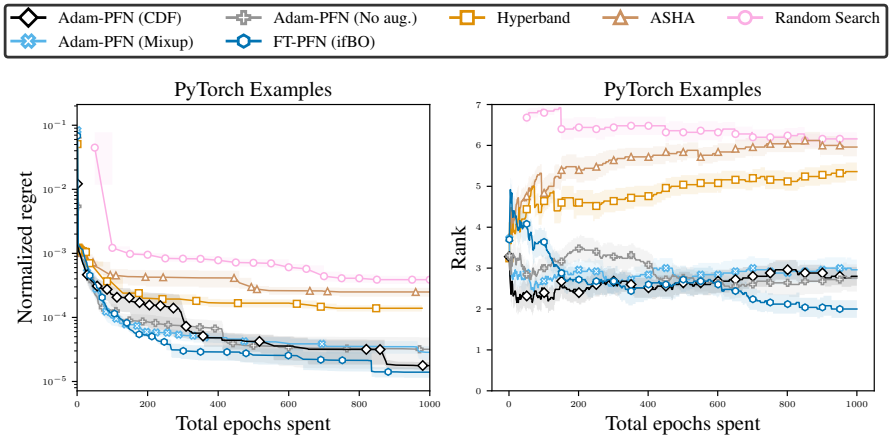
Our preliminary results highlight the importance of specialization and the use of prior knowledge when it comes to HPO. Despite that, we acknowledge that our surrogate model is trained and tested

[1] https://github.com/pytorch/examples



(a) Normalized regret (b) Average rank

Figure 2: HPO results of normalized regret and average rank of `Adam-PFN (CDF)` against the baselines. The results are the mean across the 12 evaluation tasks for 5 random seeds.



(a) Normalized regret (b) Average rank

Figure 3: HPO results of normalized regret and average rank of `Adam-PFN (CDF)` against the baselines on OOD tasks. The results are the mean across tasks for 5 random seeds.

on a fixed search space with a pre-defined number of HPs. Tuning different sets of HPs needs to be further explored. One possible approach would be to set unused HPs to default values during training and testing.

Unlike Mixup, we only consider augmentations in learning curve space. While we explored HP augmentation, we surprisingly found that augmenting in the parameter space hurts performance, whereas Mixup's results improve when we remove HP augmentation (refer to Appendix H). We believe that our results could be further improved with the introduction of a new HP augmentation method.

Finally, especially based on the results on OOD tasks, we believe it is worth exploring how our prior could be mixed with that of `FT-PFN`. One possibility would be to train a model from scratch with a prior that is a mixture of the two. Another would be to fine-tune `FT-PFN` on learning curves from our prior.

Acknowledgements. Frank Hutter is a Hector Endowed Fellow at the ELLIS Institute Tubingen. All authors acknowledge funding by the state of Baden-Wurttemberg through bwHPC, the German Research Foundation (DFG) through grant number 417962828, and the European Union (via ERC Consolidator Grant Deep Learning 2.0, grant no. 101045765). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

Bergstra, J. and Y. Bengio (2012). "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.10, pp. 281–305.

Hochreiter, S. and J. Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780.

Jamieson, K. and A. Talwalkar (2016). "Non-stochastic Best Arm Identification and Hyperparameter Optimization". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Vol. 51. Proceedings of Machine Learning Research, pp. 240–248.

Kadra, A. et al. (2023). "Scaling Laws for Hyperparameter Optimization". In: *37th Conference on Neural Information Processing Systems*.

Kingma, D. P. and J. Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015*.

Kipf, T. N. and M. Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*.

Lee, D. B. et al. (2024). *Cost-Sensitive Multi-Fidelity Bayesian Optimization with Transfer of Learning Curve Extrapolation*.

Li, L., K. Jamieson, A. Rostamizadeh, et al. (2020). "A System for Massively Parallel Hyperparameter Tuning". In: *Proceedings of Machine Learning and Systems*. Vol. 2, pp. 230–246.

Li, L., K. Jamieson, G. DeSalvo, et al. (2018). "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.185, pp. 1–52.

Loshchilov, I. and F. Hutter (2017). "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *International Conference on Learning Representations*.

Metz, L. et al. (2020). *Using a thousand optimization tasks to learn hyperparameter search strategies*.

Müller, S. et al. (2022). "Transformers Can Do Bayesian Inference". In: *International Conference on Learning Representations*.

Rakotoarison, H. et al. (2024). "In-Context Freeze-Thaw Bayesian Optimization for Hyperparameter Optimization". In: *41st International Conference on Machine Learning*.

Stoll, D. et al. (2024). *Neural Pipeline Search (NePS)*. URL: `https://github.com/automl/neps`.

Swersky, K., J. Snoek, and R. P. Adams (2014). *Freeze-Thaw Bayesian Optimization*.

Vaswani, A. et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30.

Wistuba, M., A. Kadra, and J. Grabocka (2022). "Supervising the Multi-Fidelity Race of Hyperparameter Configurations". In: *36th Conference on Neural Information Processing Systems*.

Zhang, H. et al. (2018). "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*.

A TaskSet Hyperparameter Bounds

Table 2 provides the sampling bounds for TaskSet hyperparameters. We normalize all hyperparameters to lie on the [0, 1] range using those bounds before we feed them to the model for training and inference.

HP	Log	Lower Bound	Upper Bound
Learning Rate	✓	$1e^{-8}$	10
β_1[2]	✓	$1e^{-4}$	1
β_2[2]	✓	$1e^{-6}$	1
Linear Decay	✓	$1e^{-7}$	0.0001
Expon. Decay	✓	$1e^{-6}$	0.001
ϵ	✓	$1e^{-10}$	1000
ℓ_1	✓	$1e^{-8}$	1
ℓ_2	✓	$1e^{-8}$	1

Table 2: Sampling strategies for TaskSet Hyperparameters.

B Adam-PFN Implementation Details

Following Rakotoarison et al., 2024, we used a transformer (Vaswani et al., 2017) with 6 layers and 4 attention heads, an embedding size of 512, and a hidden size of 1024. During training, we used a batch size of 25 and the Adam (Kingma and Ba, 2015) optimizer with an initial learning rate of 0.0001. We scheduled the learning rate using cosine annealing (Loshchilov and Hutter, 2017) with a linear warm-up schedule for the first 200 epochs. We train on a total of 2.0M learning curves. A list of the sampled tasks we used during training is presented in Appendix C.

C List of TaskSet Training Tasks

TaskSet provides both hand-designed tasks designed by experts, and sampled tasks, where the task parameters such as dataset, network architecture, activation functions, etc., are randomly sampled. There are far more sampled tasks in TaskSet, since they are easier to create. In our work, we exclusively train on sampled tasks. This was a design decision to highlight the fact that the learning curves we use during training do not have to come from carefully curated tasks. Instead, randomly sampling a task's parameters and optimizing it for a number of epochs suffices.

To facilitate reproducibility, we provide a detailed list of each task we used during training.

- `word_rnn_language_model_family` seeds: 0, 1, 3, 4, 5, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 29, 30, 31, 33, 34, 37, 39, 41, 42, 43, 45, 46, 47, 48, 51, 52, 53, 54, 55, 57, 58, 59, 61, 63, 64, 65, 66, 67, 69, 72, 73, 74, 76, 78, 81, 82, 84, 85, 86, 87, 88, 89, 91, 93, 94, 97, 98, 99
- `char_rnn_language_model_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99
- `conv_fc_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 30, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

[2] β_1 and β_2 are parametrized as $1 - x$ and then sampled.

- `conv_pooling_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99
- `losg_tasks_family` seeds: 0, 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 50, 51, 52, 53, 54, 56, 57, 58, 59, 61, 62, 63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99
- `maf_family` seeds: 3, 5, 7, 14, 17, 20, 23, 27, 40, 41, 44, 46, 47, 48, 60, 61, 64, 66, 68, 69, 70, 76, 78, 83, 84, 89, 94
- `mlp_ae_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99
- `mlp_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99
- `mlp_vae_family` seeds: 1, 3, 4, 7, 9, 10, 12, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 30, 32, 33, 34, 35, 36, 37, 41, 42, 45, 50, 52, 54, 56, 57, 59, 60, 61, 62, 66, 67, 69, 71, 76, 77, 78, 79, 80, 81, 82, 83, 84, 86, 87, 90, 92, 94, 97
- `nvp_family` seeds: 5, 7, 14, 19, 27, 28, 32, 35, 36, 42, 47, 49, 55, 65, 70, 73, 74, 77, 88, 89, 92, 93, 95
- `quadratic_family` seeds: 0, 1, 5, 7, 8, 9, 10, 11, 12, 13, 14, 18, 19, 21, 23, 24, 25, 28, 31, 32, 34, 35, 36, 38, 42, 43, 44, 48, 53, 62, 63, 64, 66, 68, 69, 70, 72, 73, 75, 78, 81, 82, 83, 85, 89, 91, 93, 97
- `rnn_text_classification_family` seeds: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 93, 94, 95, 96, 97, 98, 99

D List of TaskSet Evaluation Tasks

We evaluated our pipeline on the following 12 text classification tasks, which are also used in prior work (Rakotoarison et al., 2024), (Wistuba et al., 2022), (Kadra et al., 2023):

- FixedTextRNNClassification_imdb_patch128_LSTM128_avg_bs64
- FixedTextRNNClassification_imdb_patch128_LSTM128_bs64
- FixedTextRNNClassification_imdb_patch128_LSTM128_embed128_bs64
- FixedTextRNNClassification_imdb_patch32_GRU128_bs128
- FixedTextRNNClassification_imdb_patch32_GRU64_avg_bs128
- FixedTextRNNClassification_imdb_patch32_IRNN64_relu_avg_bs128
- FixedTextRNNClassification_imdb_patch32_IRNN64_relu_last_bs128
- FixedTextRNNClassification_imdb_patch32_LSTM128_E128_bs128
- FixedTextRNNClassification_imdb_patch32_LSTM128_bs128
- FixedTextRNNClassification_imdb_patch32_VRNN128_tanh_bs128
- FixedTextRNNClassification_imdb_patch32_VRNN64_relu_avg_bs128
- FixedTextRNNClassification_imdb_patch32_VRNN64_tanh_avg_bs128

E Baselines

DyHPO (Wistuba et al., 2022) includes a learned deep kernel GP as the surrogate model, combined with a generalization of the Expected Improvement (EI) acquisition function for a multi-fidelity setup. The candidate configuration suggested by the acquisition function is evaluated only for a number of fidelity steps. During evaluation we used the NePS (Stoll et al., 2024) framework implementation provided by Rakotoarison et al., 2024, which follows the implementation details and code by Wistuba et al., 2022 from their publicly available repository at `https://github.com/machinelearningnuremberg/DyHPO`.

DPL (Kadra et al., 2023) is a framework that performs Bayesian Optimization with learning curve extrapolation under the assumption that learning curves follow power law functions. Using an ensemble of neural networks as a surrogate model, and Expected Improvement (EI) at the maximum budget as the acquisition function, the candidate configurations are again trained for a number of epochs before the next BO iteration. During evaluation, we used the NePS framework implementation provided by Rakotoarison et al., 2024, which follows the implementation details and code by Kadra et al., 2023 from their publicly available repository at `https://github.com/machinelearningnuremberg/DPL`.

FT-PFN is the surrogate model introduced by Rakotoarison et al., 2024, it is a PFN (Müller et al., 2022) that performs Bayesian learning curve extrapolation, trained on synthetic datasets created by 4 weighted basis functions with additive Gaussian noise. During evaluation, we used the implementation suggested by Rakotoarison et al., 2024 and their publicly available surrogate model at `https://github.com/automl/ifBO`.

Uniform Predictor is a simple baseline that outputs a uniform distribution over the $[0, 1]$ range for the log-likelihood and a constant value of 0.5 for the MSE point estimate.

Freeze-Thaw with GPs (Swersky et al., 2014) Freeze-Thaw using GPs as the surrogate model. During evaluation, we follow the same implementation used by Rakotoarison et al., 2024.

Hyperband (Li, Jamieson, DeSalvo, et al., 2018) is a Multi-fidelity algorithm which uses different Successive Halving (Jamieson and Talwalkar, 2016) brackets with different definitions of lower fidelity per bracket. During evaluation, we used the default NePS implementation with $\eta = 3$, which leads to 3 Succesive Halving brackets. The minimum budget was set to 1 and the maximum to 50. The cutoff fidelities for the first SH bracket were $[1, 5, 16, 50]$.

ASHA (Li, Jamieson, Rostamizadeh, et al., 2020) the asynchronous version of Successive Halving. During evaluation, we used the default NePS implementation, which uses $\eta = 3$. The minimum budget was set to 1, and the maximum to 50.

Random Search (Bergstra and Bengio, 2012) an uninformed global optimization algorithm that samples hyperparameter configurations uniformly at random. During evaluation, we used the default NePS implementation. We allowed a total number of 1000 optimization steps, corresponding to 20 full random search evaluations in TaskSet tasks.

F CMBO & Mixup

CMBO (Lee et al., 2024) is a recently introduced FT-BO framework to which our work is closely related. `CMBO` is a Cost-Sensitive Multi-fidelity BO framework that introduces a new utility function, aiming to balance a user-specific tradeoff between the expected performance improvement gain during an HPO run, and the computational resources cost needed to achieve that gain.

Similar to Rakotoarison et al., 2024, CMBO uses a PFN surrogate model, but unlike `ifBO`, the surrogate is trained on tasks from TaskSet, using Mixup, a learning curve augmentation method based on Mixup for images (Zhang et al., 2018).

CMBO differs from ifBO in several important ways. Unlike ifBO, the surrogate model in CMBO is meta-learned using augmented learning curves from real tasks. A second key difference lies in the stopping criterion. CMBO uses a user-defined utility function to determine whether to continue the optimization, whereas ifBO runs until the full optimization budget is exhausted. Lastly, CMBO is not yet publicly available.

Our approach also differs from CMBO in several ways. Our surrogate is trained on augmented curves from 878 tasks, compared to only 21 used in CMBO. We evaluated our model on both hand-designed NLP tasks and real-world tasks. Additionally, we investigate an alternative to Mixup that does not include the HP augmentation component (see also Appendix H). Lastly, CMBO is a new FT-BO framework, while our work builds upon the pre-existing open source ifBO and offers a drop-in replacement for its surrogate model.

For completeness, this appendix section describes Mixup, though more details can be found in the original paper by Lee et al., 2024[3].

Mixup consists of two discrete augmentation steps. The first step involves augmenting the same hyperparameter configurations across tasks, and the second step involves augmenting across hyperparameter configurations within the same task.

Assume M tasks and LC_m a vector of learning curve - hyperparameter configuration combinations for task $m \in M$,

$$LC_m = [LC_{m,1}, LC_{m,2}, LC_{m,3}, ..., LC_{m,n}] \tag{2}$$

During task mixup, Lee et al., 2024 sample a scalar value uniformly at random $\lambda_1 \sim Unif(0, 1)$ and perform learning curve augmentation as follows:

$$LC'_m = \lambda_1 LC_m + (1 - \lambda_1) LC_{m'} \text{ for all } m, m' \in M \tag{3}$$

This step produces a new learning curve for each hyperparameter configuration as a linear interpolation between the learning curves of tasks m and m'.

The second step involves mixing hyperparameter configurations and learning curves within the same task. Assume we have N hyperparameter configuration - learning curve pairs denoted as (x, y) per task. During hyperparameter mixup, Lee et al., 2024 sample a scalar value uniformly at random $\lambda_2 \sim Unif(0, 1)$ and perform the augmentation as follows:

$$(x'', y'') = \lambda_2 (x, y) + (1 - \lambda_2)(x', y') \tag{4}$$

Figure 4 shows the result of applying Mixup on the same set of learning curves as Figure 1. Due to the linear interpolation nature of Mixup, the resulting learning curves are smoother than the original, with some information, especially of the oscillating high learning curves, being phased out.

G Out-of-Distribution Results Per Task

We considered the following tasks:

- **Language Translation**: This task trains a transformer to perform language translation from German to English.
- **Word Language Model**: This task trains a neural network for language modeling by learning to predict the next word in a sequence. We evaluated two variants of this task, one using LSTMs (Hochreiter and Schmidhuber, 1997) and one using a transformer.
- **Sequence Predictor**: This task uses an LSTM to predict future values of sine wave signals.

[3]All the equations in this section are from Lee et al., 2024

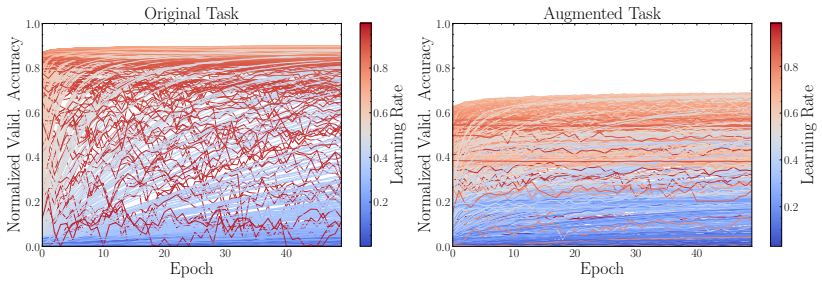


Figure 4: (a) A selection of learning curves from TaskSet, the color of each curve is based on the learning rate value of the configuration. (b) The same set of learning curves after we apply Mixup learning curve augmentation.

- **GCN**: This task uses a Graph Convolutional Network (Kipf and Welling, 2017) to perform node classification on the Cora dataset.

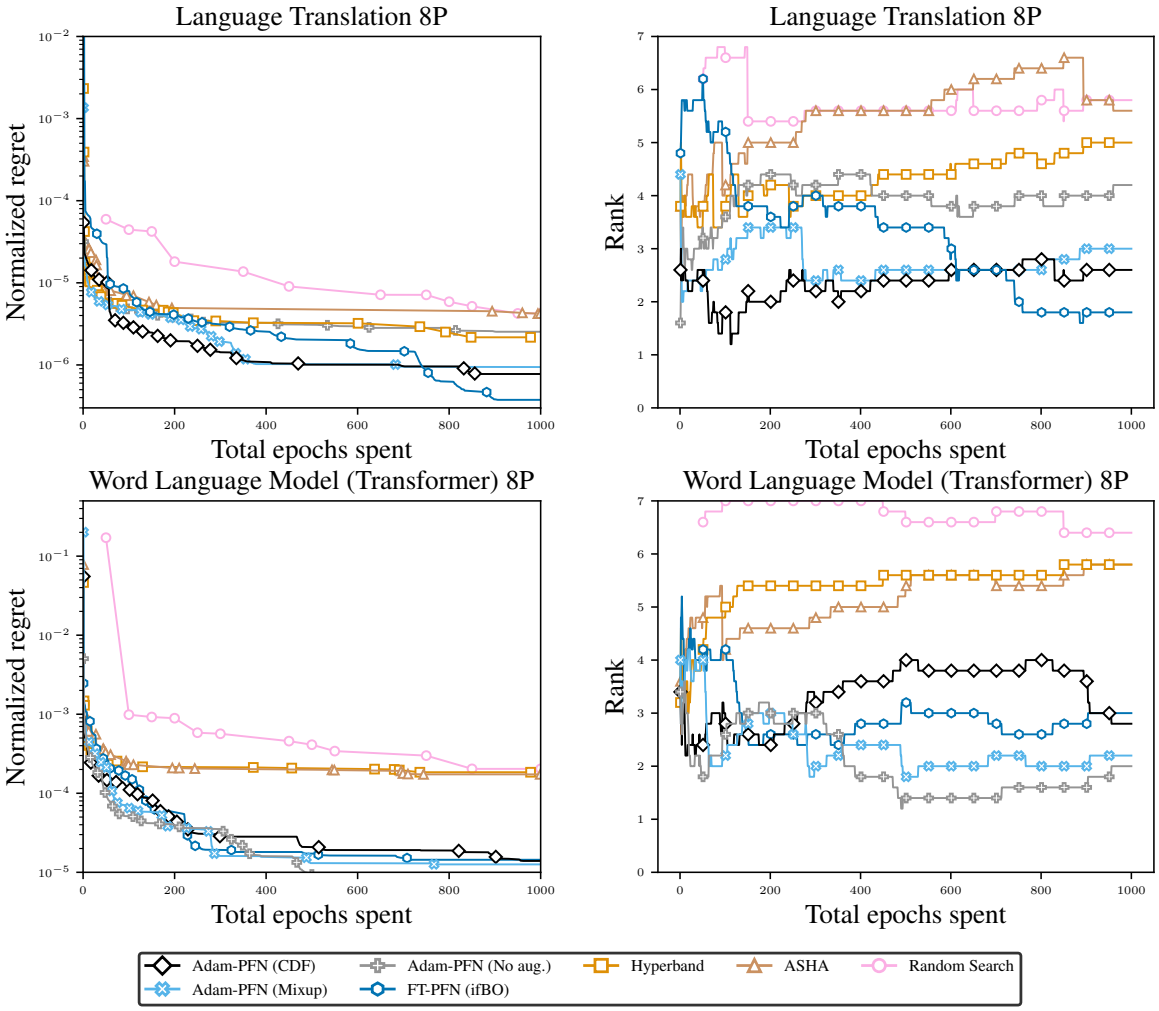


Figure 5: HPO results on real-world tasks.

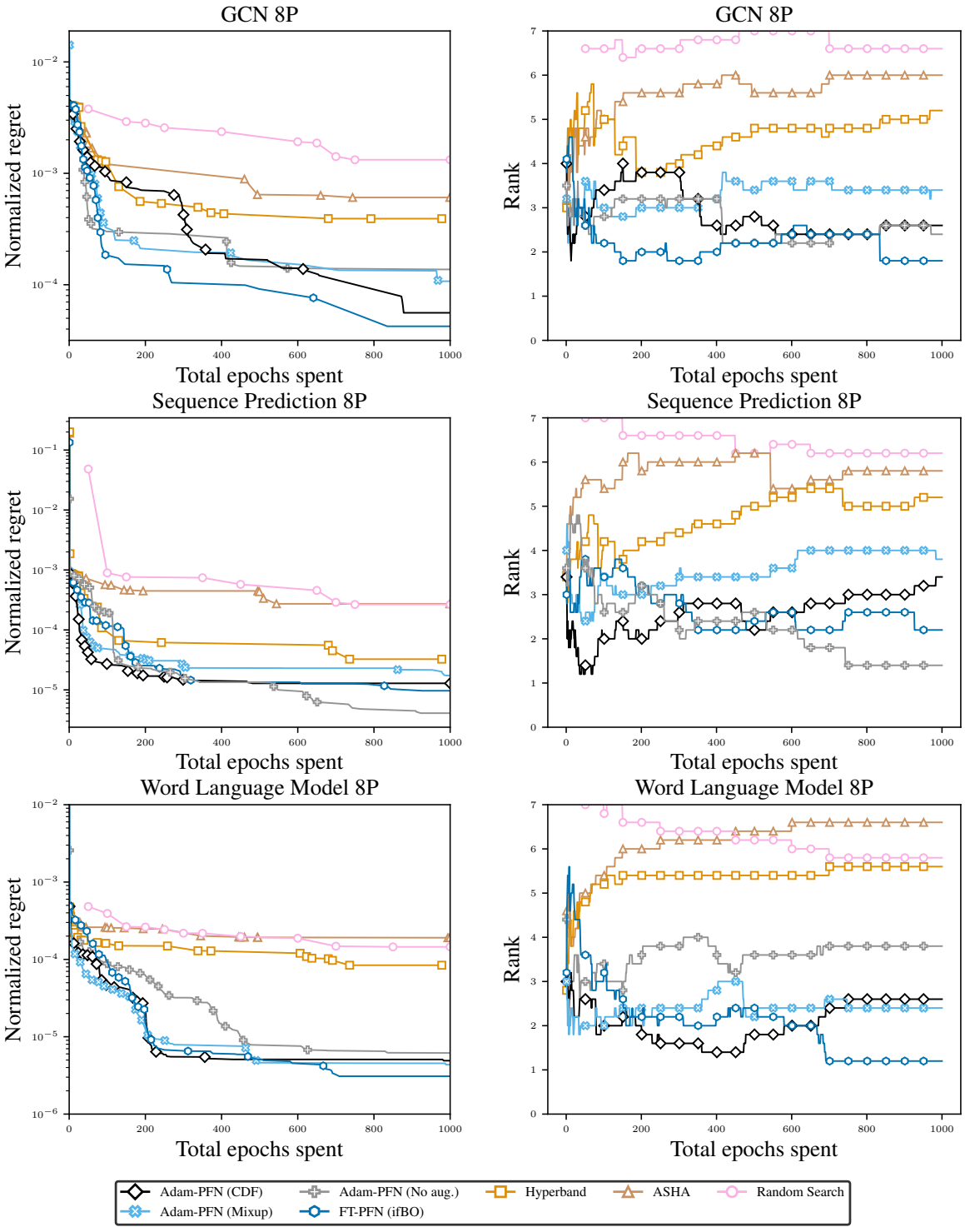


Figure 6: HPO results on real-world tasks.

H Augmenting the HPs

Mixup augments both learning curves and HP configurations, so we also extended CDF-augment to support HP augmentation. Specifically, we transformed HP configurations using the cumulative distribution function (CDF) of the Beta distribution, similar to Equation 1.

$$x' = F_{Beta}(x; \mu, \kappa) \tag{5}$$

To avoid extreme augmentation in HP space, we sample the concentration parameter κ uniformly from the [2, 3] range, instead of [2, 5] that we used for learning curve augmentation.

To further isolate the effect of HP augmentation in Mixup, we performed an additional ablation study, in which we disabled the HP component and applied only Equation 3. The learning curve extrapolation performance of these `Adam-PFN` variants is reported in Table 3 for different context sizes.

Overall, the results indicate that HP augmentation negatively impacts performance. For the TaskSet-8P evaluation tasks (see Table 3), the surrogate model performs worse when trained with HP augmentation for context sizes of 400 and 1000. The model performs equally well with the surrogate trained only with learning curve augmentation for a context size of 1600. HP augmentation also appears to hurt the performance of Mixup, since the results improved when we applied only task mixup.

Algorithm	**TaskSet-8P**					
	Context 400		**Context 1000**		**Context 1600**	
	LL	MSE	LL	MSE	LL	MSE
CDF (w/ HP)	5.309	0.00055	5.408	0.00047	**5.441**	**0.00043**
Mixup (w/ HP)	4.916	0.00080	4.986	0.00067	5.022	0.00061
CDF	**5.326**	**0.00054**	**5.422**	**0.00046**	**5.441**	**0.00043**
Mixup (w/o HP)	5.296	0.00062	5.374	0.00051	5.409	0.00046
No Aug.	4.947	0.00062	5.042	0.00054	5.066	0.00052

Table 3: Learning Curve extrapolation results for different augmentation methods. "w/o HP" indicates models trained without hyperparameter augmentation. "No Aug." is trained without any augmentation.

We were somewhat cautious about these results, as the models may have overfitted to the HP configurations present on TaskSet, and thus may not be able to generalize to unseen configurations. To force the models to generalize during evaluation, we performed an additional experiment where we randomly withheld 400 HP configurations during surrogate model training and evaluated the performance exclusively on this held-out set for the evaluation tasks. The results are shown in Table 4.

The results show that the models without HP configuration (`CDF` and `Mixup (w/o HP)`) were able to generalize to unseen HP configurations and also outperform their counterparts. Only learning curve augmentation seems to introduce enough variability in the data, and even though the transformer is trained on the same 600 HP configurations, it learns a sufficiently robust representation of the space, which allows generalization on previously unseen HPs.

Algorithm	TaskSet-8P (400 Leave-Out)					
	Context 400		Context 1000		Context 1600	
	LL	MSE	LL	MSE	LL	MSE
CDF (w/ HP)	5.258	0.00054	5.380	0.00045	5.418	0.00043
CDF	**5.329**	**0.00051**	**5.418**	**0.00042**	**5.458**	**0.00039**
Mixup (w/ HP)	4.850	0.00089	4.983	0.00078	5.035	0.00075
Mixup (w/o HP)	5.264	0.00053	5.357	0.00047	5.385	0.00044

Table 4: Learning Curve extrapolation results for Adam-PFN variants with and without hyperparameter augmentation on 400 leave-out configurations.