PriorBand: HyperBand + Human Expert Knowledge

Anonymous Author(s)
Affiliation
Address
email

Abstract

Hyperparameters of Deep Learning (DL) pipelines are crucial for their performance. While a large number of methods for hyperparameter optimization (HPO) have been developed, they are misaligned with the desiderata of a modern DL researcher. Since often only a few trials are possible in the development of new DL methods, manual experimentation is still the most prevalent approach to set hyperparameters, relying on the researcher's intuition and cheap preliminary explorations. To resolve this shortcoming of HPO for DL, we propose `PriorBand`, an HPO algorithm tailored to DL, able to utilize both expert beliefs and cheap proxy tasks. Empirically, we demonstrate the efficiency of `PriorBand` across a range of DL models and tasks using as little as the cost of 10 training runs and show its robustness against poor expert beliefs and misleading proxy tasks.

1 Introduction

The performance of Deep Learning (DL) models are crucially dependent on dataset-specific settings of their hyperparameters, such as learning rate and weight decay. Therefore, hyperparameter optimization (HPO) is an integral step in the development of DL models. In contrast to the application of HPO to traditional ML models on fairly small datasets, current DL practitioners have the following desiderata for a strategy for setting hyperparameters: (i) Strong performance under low budgets: With ever-increasing resource requirements for training, a handful of model trainings needs to suffice to set the many hyperparameters of DL pipelines; (ii) Incorporating evaluations of cheap proxy tasks, while being robust to misleading ones; (iii) Integration of expert beliefs, while being robust to misspecified ones; (iv) Handling of mixed type search space, i.e., categoricals and numericals; (v) Simplicity: This includes conceptual simplicity (DL experts used to hand-tune their hyperparameters tend to dislike complex black-box HPO algorithms) and implementation simplicity (for ease of integration).

Methods for HPO have been explored extensively [Feurer and Hutter, 2019, Bischl et al., 2021]. Nonetheless, all existing HPO methods fail to meet some of the desiderata above and are thus not of practical relevance for DL developers. To resolve these shortcomings, we introduce `PriorBand`, a multi-fidelity optimization algorithm that integrates an expert's prior distribution over optimal hyperparameters in a simple yet efficient manner. The **main contributions** of our work are as follows:

1. We propose `PriorBand`, the first HPO algorithm that satisfies the above desiderata of DL experts (Section 2). In its design, we strive for the smallest possible update to `HyperBand` to effectively integrate human expert knowledge and accomplish low-budget efficiency.
2. We demonstrate the efficiency and robustness of `PriorBand` on a wide suite of DL tasks using practically feasible compute budgets (Section 3).

Submitted to the 6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans. Do not distribute.

Algorithm 1 The main algorithm for both `HyperBand` and `PriorBand`.

1: **Input:** Distribution over optimum π, halving parameter η, resource bounds $[r, R]$.
2: $s_{max} \leftarrow \lfloor \log_\eta(R/r) \rfloor$
3: **for** $s \in \{s_{max}, \ldots, 0\}$ **do**
4: $\quad n \leftarrow \left\lceil \frac{(s_{max}+1)}{s+1} \cdot \eta^s \right\rceil, \quad r \leftarrow R\eta^{-s}$
5: $\quad \boldsymbol{X}_{next} \leftarrow$ `get_hyperparameter_configurations`(π, η, n) ▷ Algorithm 2
6: $\quad \hat{\boldsymbol{x}} \leftarrow$ `do_successive_halving`$(\boldsymbol{X}_{next}, s, n, r, R, \eta)$
7: **end for**

In Section 4 we provide an overview of the related literature and in Section 5 we highlight possible limitations and thus future directions of work. We release the code[1] for reproducing our experiments.

2 An approach for low budget hyperpameter optimization

The problem we consider is the minimization of an expensive to evaluate objective function f, i.e., $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$, where the search space $\mathcal{X}$ may constitute any combination of continuous and discrete variables, including categorical variables (desideratum (iv)). To take desiderata (i)-(iii) into account, we modify this basic problem setting to include cheap proxy tasks and expert beliefs. The minimization of f represents the minimization of the loss of a DL model, the *configuration* $\boldsymbol{x}$ of hyperparameters represents a point in the search space $\mathcal{X}$, and the proxy tasks represent for example the number of epochs used to train the model. To satisfy desideratum (v), we strive for the smallest effective change to the simple and popular `HyperBand` algorithm [Li et al., 2018].

Multi-fidelity hyperparameter optimization with expert priors To utilize cheap proxy tasks on the one hand, we introduce an additional fidelity variable $z \in [z_{\text{min}}, z_{\text{max}}]$ to our objective function f, where the highest fidelity z_{max} is the desired fidelity. On the other hand, to incorporate expert beliefs, we allow experts to define a distribution over the location of the optimum $\pi(\boldsymbol{x}) = \mathbb{P}\left(f(\boldsymbol{x}) \text{ is optimal}\right)$, i.e., the prior probability that $\boldsymbol{x}$ is optimal according to the user. Therefore, we introduce the problem of multi-fidelity HPO with expert priors as

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}, z_{\text{max}}) \quad \text{given } \pi, \tag{1}$$

where $f(\boldsymbol{x}, \cdot)$ can be queried at the full range of $[z_{\text{min}}, z_{\text{max}}]$. Thus, we look to solve Equation 1 while adhering to restrictive budget constraints, using cheaper, lower-fidelity approximations where $f(\boldsymbol{x}, z) \approx f(\boldsymbol{x}, z_{\text{max}})$, and the expert prior π can be a valuable source of information to optimize f.

The `PriorBand` algorithm `PriorBand` is a minimal modification of the prominent `HyperBand` algorithm [Li et al., 2017], which is given in Algorithm 1. It only changes one function (shown in blue at line 5) from `HyperBand`, namely `get_hyperparameter_configurations`, as follows. While `HyperBand` implements this function by random sampling, we propose to replace this with a mix of random samples, samples from the user prior, and samples close to the current best known configuration (the so-called *incumbent*). In Algorithm 2, `PriorBand` starts each new `SuccessiveHalving` [Jamieson and Talwalkar, 2016] bracket by sampling η configurations from the vicinity of the current incumbent configuration $\hat{\boldsymbol{x}}$; This vector of configurations is represented by $\boldsymbol{X}_{next}$ in Algorithm 1. Subsequently, a bracket with n_i initial configurations adds the remaining n_i/η configurations by sampling from the expert prior π. The remaining required initial configurations are randomly sampled. We detail the sampling of neighbors procedure, called `sample_from_incumbent` in Algorithm 2 in Appendix F.

3 Experiments

We answer three research questions — For details on our experimental setting see Appendix D.

[1] `https://anonymous.4open.science/r/mf-prior-exp-anon-submission-5093`

Algorithm 2 `PriorBand` ensemble sampling

Input: Distribution over optimum π, halving parameter η, number of samples n
$n_\pi \leftarrow \lfloor n/\eta \rfloor, \quad n_U \leftarrow n - \eta - n_\pi$ ▷ Compute counts of remaining samples
if $\hat{\boldsymbol{x}}$ exists **then**
  $\boldsymbol{X}_s^{\hat{\boldsymbol{x}}} \leftarrow$ `sample_from_incumbent`$(\eta, \hat{\boldsymbol{x}})$ ▷ Draw η neighbors of $\hat{\boldsymbol{x}}$
end if
$\boldsymbol{X}_s^{\pi} \leftarrow$ `sample_from_prior`(n_π, π)
$\boldsymbol{X}_s^{U} \leftarrow$ `sample_from_uniform`(n_U)
return $\left(\boldsymbol{X}_s^{\hat{\boldsymbol{x}}} \cup \boldsymbol{X}_s^{\pi} \cup \boldsymbol{X}_s^{U}\right)$

RQ1: Is it enough to replace random samples by samples from the prior? To illustrate the challenges that may arise when combining expert priors with multi-fidelity optimization, we experiment with variations of a synthetic function (Hartmann 3D) to study four possible cases: The combination of good (high $\pi(\boldsymbol{x}^*)$) or bad (low $\pi(\boldsymbol{x}^*)$) expert priors, and good or bad inter-fidelity correlations.

We first study the simplest extensions of random search (RS) and `HyperBand` (HB) with expert priors one can think of: Replacing the uniform sampling of configurations in these approaches by sampling from the prior. We improve this further by drawing the first sample from the mode of the expert prior and refer to the resulting algorithms as RS+prior and HB+prior. Figure 1 shows that this simple way of integrating the expert prior is not very helpful. While the prior boosts performance when it is well located, it hurts substantially when it is poorly so, for both RS and HB. This result confirms that simply substituting random sampling with prior-based sampling is inadequate in handling multi-fidelity optimization.

RQ2: Does `PriorBand` alleviate the failure mode of prior sampling? Building on the findings of RQ1, we designed `PriorBand` to be robust to misspecified expert priors. We compare our `PriorBand` against the prior-based variants of random search and `HyperBand`. In Figure 2 (top row) we display `PriorBand`'s ability to handle the extreme cases demonstrated in RQ1 well, while not being much slower in the presence of a good prior.

In Figure 2 (middle and bottom row) we run a similar comparison but across real-world DL benchmarks (Appendix D.1) that include joint architecture and hyperparameter spaces [Bansal et al., 2022, Zimmer et al., 2021] and transformer spaces [Wang et al., 2021]. All configurations are given the default configuration (on the lowest fidelity where applicable) as the first sample. In almost all cases, `PriorBand` outperforms `HyperBand` with prior sampling. Naturally, this is more apparent in cases where the user's prior input is misleading (bottom row of Figure 2).

RQ3: Does `PriorBand` work in practice? Having seen the improvement over `HyperBand` in RQ2, we now pit `PriorBand` against the commonly used classes of hyperparameter optimizers, demonstrating its efficacy if used in practice on real DL problems. Here, we repeat the experiments of RQ2 but compare against Bayesian Optimization and πBO [Hvarfner et al., 2022] as strong black-box optimizers, and against `HyperBand` and BOHB [Falkner et al., 2018a] as multi-fidelity optimizers. Random search with priors and πBO are baselines that can take a distribution over optimal

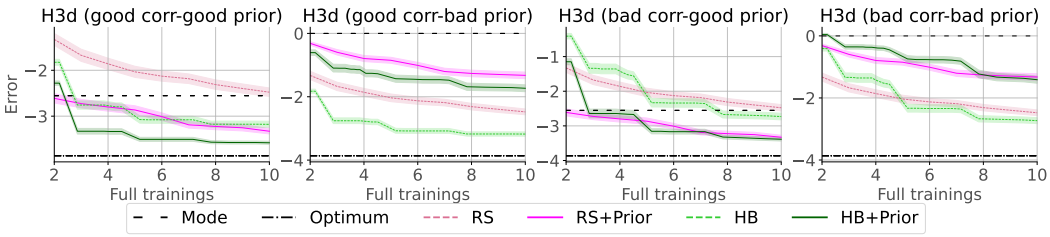


Figure 1: Illustrating the 4 extreme cases of multi-fidelity optimization with expert priors on the synthetic Hartmann 3D function. We show the mean over 50 seeds with the standard error band.

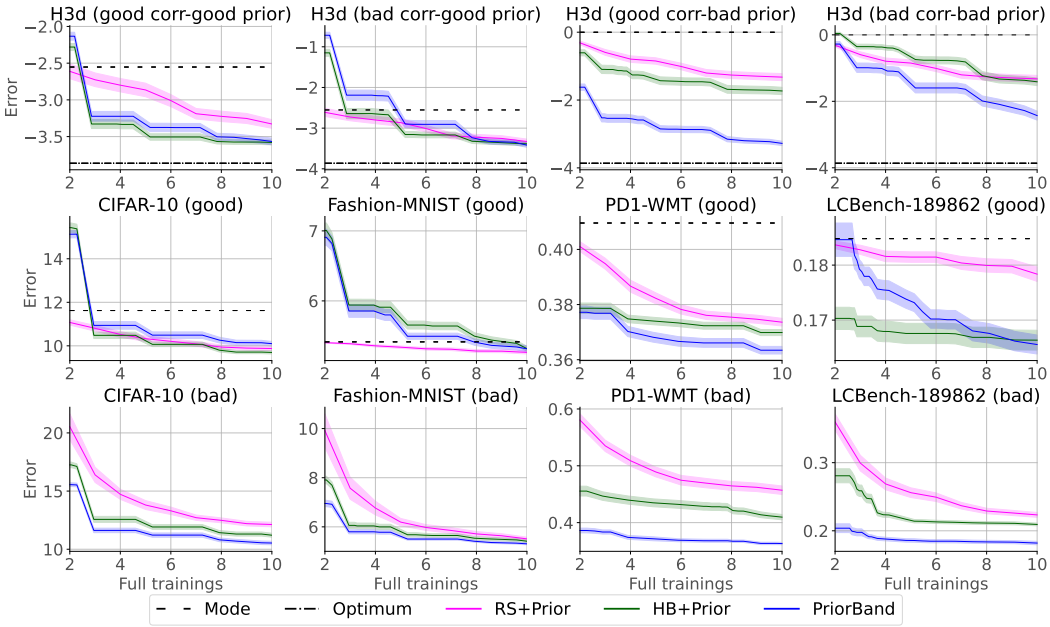


Figure 2: Comparative performance of `PriorBand`. For the real-world HPO tasks, we show plots for a good prior (row 2) and a bad prior (row 3). All algorithms evaluate the mode of the expert prior distribution first. We show the mean over 50 seeds with the standard error band.

configurations as the input and recover the mode as the first evaluation. The results in Appendix E, Figure 7 demonstrate the competitiveness of `PriorBand` against these baselines across benchmarks, for both settings of good and bad priors. Under a budget as small as 10 max-budget evaluations, model-based approaches may struggle to build an accurate enough model to make informed decisions. As such, the model-free `PriorBand` frequently outperforms model-based methods.

4 Related work

While using expert priors for hyperparameter optimization has been explored previously, few works considered priors over the optimum [Bergstra et al., 2011, Hvarfner et al., 2022, Souza et al., 2021], and they all target the single-fidelity setting. The expert priors we consider should not be confused with the priors natively supported by Bayesian optimization, which represent priors over function structure through the choice of the kernel [Snoek et al., 2012, Swersky et al., 2013, Oh et al., 2018].

In Deep Learning, training epochs and dataset subsets [Swersky et al., 2014, Klein et al., 2017, Nguyen et al., 2020] are frequently used as fidelity variables to create cheap proxy tasks, with input resolution, network width, and depth also occasionally used [Bansal et al., 2022]. Successive Halving [Jamieson and Talwalkar, 2016] and `HyperBand` [Li et al., 2018] are effective randomized policies for multi-fidelity HPO that use early stopping of configurations on a geometric spacing of the fidelity space and can also be extended to the model based setting [Falkner et al., 2018b]. Our `PriorBand` adaptation of `HyperBand` could potentially also be extended along these lines.

5 Limitations and future work

There are some limitations to `PriorBand`. The efficacy of our distance measure could be ablated and improved, and, in principle, `PriorBand` can be extended with a surrogate model and acquisition function to potentially build upon the headstart it obtains at low budgets. Finally, studying parallel versions of `PriorBand` will help improve efficiency further.

References

N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyberband for scalable, robust and efficient Hyperparameter Optimization. In Z. Zhou, editor, *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, pages 2147–2153, 2021.

A. Bansal, D. Stoll, M. Janowski, A. Zela, and F. Hutter. JAHS-bench-201: A foundation for research on joint architecture and hyperparameter search. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=_HLcjaVlqJ`.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Proceedings of the 24th International Conference on Advances in Neural Information Processing Systems (NeurIPS'11)*, pages 2546–2554. Curran Associates, 2011.

B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. Mantovani, J. N. van Rijn, and J. Vanschoren. Openml benchmarking suites and the openml100. *arXiv:1708.03731v1 [stat.ML]*, 2019.

B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A. Boulesteix, D. Deng, and M. Lindauer. Hyperparameter Optimization: Foundations, algorithms, best practices and open challenges. *arXiv:2107.05847 [stat.ML]*, 2021.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. Smola, C. Aggarwal, D. Shen, and R. Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pages 785–794. ACM Press, 2016.

X. Dong, M. Tan, A. Yu, D. Peng, B. Gabrys, and Q. Le. Autohas: Differentiable hyper-parameter and architecture search. *arXiv:2006.03656 [cs.CV]*, 2020.

A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, and E. Teppan. Industrial applications of answer set programming. *KI-Künstliche Intelligenz*, pages 1–12, 2018a.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient Hyperparameter Optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, volume 80, pages 1437–1446. Proceedings of Machine Learning Research, 2018b.

M. Feurer and F. Hutter. Hyperparameter Optimization. In F. Hutter, L. Kotthoff, and J. Vanschoren, editors, *Automated Machine Learning: Methods, Systems, Challenges*, chapter 1, pages 3–38. Springer, 2019. Available for free at http://automl.org/book.

C. Hvarfner, D. Stoll, A. Souza, L. Nardi, M. Lindauer, and F. Hutter. πBO: Augmenting acquisition functions with user beliefs for bayesian optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*, 2022. Published online: `iclr.cc`.

K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and Hyperparameter Optimization. In A. Gretton and C. Robert, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS'16)*, volume 51. Proceedings of Machine Learning Research, 2016.

K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian Optimisation with Continuous Approximations. In D. Precup and Y. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, pages 1799–1808. Proceedings of Machine Learning Research, 2017.

J. Kiili, E. Laaksonen, R. Turner, D. Eriksson, S. Park, M. Mccourt, Z. Xu, and I. Guyon. Black box optimization challenge, 2020.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian Hyperparameter Optimization on large datasets. *Electronic Journal of Statistics*, 11:4945–4968, 2017.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-based configuration evaluation for Hyperparameter Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*, 2017. Published online: `iclr.cc`.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

V. Nguyen, S. Schulze, and M. A. Osborne. Bayesian optimization for iterative learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors, *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, 2020.

C. Oh, E. Gavves, and M. Welling. BOCK : Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, pages 3865–3874, 2018.

F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, and B. Bischl. YAHPO gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *First Conference on Automated Machine Learning (Main Track)*, 2022. URL `https://openreview.net/forum?id=SWcg-TrUgc`.

J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter. NAS-bench-301 and the case for surrogate benchmarks for Neural Architecture Search. *arXiv:2008.09777v4 [cs.LG]*, 2020.

J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'12)*, pages 2960–2968. Curran Associates, 2012.

A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Bayesian optimization with a prior for the optimum. In *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part III*, volume 12977 of *Lecture Notes in Computer Science*, pages 265–296. Springer, 2021.

K. Swersky, J. Snoek, and R. Adams. Multi-task Bayesian optimization. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Proceedings of the 26th International Conference on Advances in Neural Information Processing Systems (NeurIPS'13)*, pages 2004–2012. Curran Associates, 2013.

K. Swersky, J. Snoek, and R. Adams. Freeze-thaw Bayesian optimization. *arXiv:1406.3896 [stats.ML]*, 2014.

Zi Wang, George E. Dahl, Kevin Swersky, Chansoo Lee, Zelda Mariet, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained gaussian processes for bayesian optimization, 2021. URL `https://arxiv.org/abs/2109.08215`.

L. Zimmer, M. Lindauer, and F. Hutter. Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3079 – 3090, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss societal and environmental impacts in Appendix B.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We publish code for reproducing our experiments at `https://anonymous.4open.science/r/mf-prior-exp-anon-submission-5093`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We report most hyperparameters in Appendix D.1 and Appendix D.2, and the rest can be found in our code (see 3.a).
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the standard error of the mean.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix C.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] While we do not yet upload the surrogate benchmarks that we created, we do provide code that creates them from existing publicly available data and we will provide trained surrogate models for the final version.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Resources used

All experiments in the paper were performed on cheap-to-evaluate surrogate benchmarks. We used several Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz to perform our experiments. Running one seed of one algorithm on one benchmark requires roughly 0.25 core hours. In total, the results we generated for our final experiments required 10 algorithms, 50 seeds, 22 benchmarks-prior pairs which totals 11000 runs which equates to 2750 core hours. We also trained surrogates for 2 metrics on 3 datasets for 4 hours and 8 cores, totalling another 192 core hours. We also estimate that some preliminary testing runs cost 735 core hours, a quarter of the total final experiment cost. This leads our total experimental costs to be at roughly 3677 core hours.

B Societal and environmental impact

Here, we discuss the potential societal and environmental impacts our work can have.

Environmental Beyond the experiments we ran, our work can have an impact on the environment as follows. The contributed algorithm `PriorBand` is designed to help reduce compute requirements for finding good performing DL pipelines and thus help reduce carbon emissions for HPO in DL. However, as `PriorBand` facilitates the adoption of HPO, there could also be a rebound effect.

Societal Our paper and the contributed algorithm `PriorBand` is designed to assist a wide range of DL users in their tasks. The ability of `PriorBand` to tune DL models under affordable compute can hopefully enable DL to be used by more people and even non-experts without access to much compute. The societal impact depends on which task and DL pipeline `PriorBand` is applied to.

C Licenses

- Our implementations - **MIT License**
- JAHS-Bench-201 benchmark [Bansal et al., 2022] - **MIT License**
- YAHPO-Gym benchmark [Zimmer et al., 2021] - **Apache License 2.0**
- Learning curve benchmark [Zimmer et al., 2021] - **Apache License 2.0**
- PD1 [Wang et al., 2021] - **Apache License 2.0**
- BOHB [Falkner et al., 2018a] from HpBandSter - **BSD 3-Clause License**

D Experiment details

Experiment setup All algorithms were run for the same 50 seeds for each benchmark. We cut off optimization traces after the equivalent of $10 \times z_{\text{max}}$ for a benchmark. Aside from the synthetic Hartmann function experiments (Appendix D.1.1), all others are experiments using surrogates for DL models that use *epochs* as the fidelity variable. For each experiment, the minimum budget (z_{min}) was chosen such that `HyperBand` discretizes the fidelity space into 4 fidelity levels for $\eta = 3$. The incumbent was chosen to be the configuration with the best seen performance across any fidelity.

D.1 Benchmarks

The benchmarks we use are provided by our collected suite of multi-fidelity benchmarks (`mf-prior-bench`) that treats priors as first class citizen. We include our own synthetic Hartmann functions (D.1.1) extened to the multi-fidelity setting. We wrap JAHS-Bench-201 [Bansal et al., 2022] (D.1.2) and Yahpo-Gym [Pfisterer et al., 2022] (D.1.4) and provide new surrogate benchmarks for large models for image and language tasks, trained from optimization data obtained from the benchmark from HyperBO [Wang et al., 2021] (D.1.3).

D.1.1 Multi-fidelity synthetic Hartmann (MFH)

The multi-fidelity Hartmann functions follow the design of [Kandasamy et al., 2017], where, for $[0, 1]$-scaled z, the fidelity is parameterized as

$$g(\boldsymbol{x}, z) = \sum_{i=1}^{4} (\alpha_i - \alpha_i^{'}(z; b)) \exp\left(-\sum_{j=1}^{D} A_{ij}(x_j - P_{ij})^2\right) \tag{2}$$

where for Hartmann-3,

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = 10^{-4} \times \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix},$$

and for Hartmann-6,

$$A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad P = 10^{-4} \times \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

where $\alpha_i^{'}(z; b)) = b(1 - z_i)$. In the original paper, the variable which accounts for the bias between fidelities, b, is set to 0.1. To account for the fact that we only consider a single fidelity variable, we set $z_i = z, \forall i$, and increase the bias terms significantly to create realistic task correlations. For the good fidelity, we set $b = 2.5$, and for the bad fidelity task, we set $b = 4$. The tasks incorporated half-normally distributed noise with $\sigma = 5(1 - z)$ for the bad fidelity task and $\sigma = 2(1 - z)$ for the good fidelity correlation.

name	type	values	info
X_0	continuous	$[0.0, 1.0]$	
X_1	continuous	$[0.0, 1.0]$	
X_2	continuous	$[0.0, 1.0]$	
z	log integer	$[3, 100]$	fidelity

Table 1: Synthetic Multi-Fidelity Hartmann search space in 3 dimensions.

name	type	values	info
X_0	continuous	$[0.0, 1.0]$	
X_1	continuous	$[0.0, 1.0]$	
X_2	continuous	$[0.0, 1.0]$	
X_3	continuous	$[0.0, 1.0]$	
X_4	continuous	$[0.0, 1.0]$	
X_5	continuous	$[0.0, 1.0]$	
z	log integer	$[3, 100]$	fidelity

Table 2: Synthetic Multi-Fidelity Hartmann search space in 6 dimensions.

D.1.2 JAHS-Bench-201

The JAHS-Bench-201 [Bansal et al., 2022] is a benchmark consisting of surrogates trained on 140 million data points of Neural Networks train on 3 datasets, namely CIFAR10, Colorectal-Histology and Fashion-MNIST. They extend the search space beyond the original tabular search space of NAS-Bench-201 [Dong et al., 2020] consisting of purely discrete architectural choices, introducing both hyperparameters and multiple fidelities to create the first multi-multi-fidelity benchmark for deep learning hyperparameter optimization. Each of the three datasets share equal search spaces while we fix the fidelity parameters, depth `N` and width `W`, to their maximum. We further limit `Resolution` to a fixed value of 1.0 out of the three original choices $\{0.25, 0.5, 1.0\}$. The surrogates provided by JAHS-Bench-201 do not explicitly model the monotonic constraint that as `epoch` increase, so should the training cost. In practice, this was found to be insignificant but we state so for completeness. For these benchmarks, optimizers minimize `1 - valid_acc`.

name	type	values	info
Activation	categorical	{ReLU,Hardswish,Mish}	
LearningRate	continuous	$[0.001, 1.0]$	log
N	constant	5	
Op1	categorical	{0,1,2,3,4}	
Op2	categorical	{0,1,2,3,4}	
Op3	categorical	{0,1,2,3,4}	
Op4	categorical	{0,1,2,3,4}	
Op5	categorical	{0,1,2,3,4}	
Op6	categorical	{0,1,2,3,4}	
Optimizer	constant	SGD	
Resolution	constant	1.0	
TrivialAugment	categorical	{True,False}	
W	constant	16	
WeightDecay	continuous	$[1e\text{-}05, 0.01]$	log
epoch	integer	$[3, 200]$	fidelity

Table 3: The JAHS-Bench-201 search space for all 3 datasets, CIFAR10, Colorectal-Histology and Fashion-MNIST.

D.1.3 PD1 (HyperBO)

The PD1 benchmarks consists of surrogates trained on the learning curves of large architectures, spanning both language and computer vision tasks. The original tabular data is obtainable from the output generated by HyperBO [Wang et al., 2021] and enable us to test our methods and baselines for low-budget settings, where multi-fidelity methods are most applicable. This tabular data consists of 4 collections of optimization records, one where the authors provide a grid like view of their search space and ones that their optimizer selected as well as their initial testing run before their full runs. To maximize the data available to the surrogate, we utilize all of this data but take care to drop duplicated runs from their test runs.

The original data is a mixed of several datasets, models and their parameters for which we do some preprocessing. All data-preprocessing is available as part of `mf-prior-bench` and consists of:

- Splitting the raw data by all available $\{datasetname, model, batchsize\}$ subsets.
- Identify which columns are hyperparameters by those being marked as such and consist of more than one unique value.
- Drop all columns which are not hyperparameters or metrics.
- Drop all `NaN` values for which no metrics are recorded.
- Drop duplicated entries, keeping those from the final experimental runs.

Any further pre-processing required will be specified with each dataset. Once the datasets are prepared, we then train a single surrogate XGBoost model [Chen and Guestrin, 2016] per metric. This training was performed using DEHB [Awad et al., 2021], optimizing for the mean R2 loss of 5 fold cross-validation for a total of 4 hours, 8 CPU cores and the seed set to 1. The training procedure can be found as part of `mf-prior-bench` but we will soon integrate the surrogates as part of `mf-prior-bench` . The benchmarks utilized for our work are listed below with the numbers (A, B, C, D) the number of entries after **A)** considering all data available, **B)** doing dataset specific cleaning, **C)** dropping `NaN` rows and finally, **D)** dropping duplicate entries.

For these benchmarks, optimizers aim to minimize the `valid_error_rate`.

- The **lm1b_transformer_2048** dataset required dropping entries that recorded train costs above $10,000$, where most runs maximally achieved < 400. There were 1059 of these runs

with some additionally recording `NaN` for their valid error rate. The number of available samples during the preprocessing steps were (69974, 68915, 68915, 68915)

name	type	values	info
lr_decay_factor	continuous	$[0.010543, 0.9885653]$	
lr_initial	continuous	$[1e\text{-}05, 9.986256]$	
lr_power	continuous	$[0.100811, 1.999659]$	
opt_momentum	continuous	$[5.9e\text{-}05, 0.9989986]$	
epoch	integer	$[1, 74]$	fidelity

Table 4: The `lm1b_transformer_2048` search space.

- The **translate_xformer_64** contained a substantial amount of duplicated configurations with only 2738 unique configurations before any cleaning was done. Coupled with the 19 epochs recorded, we would expect a perfect set of non-failing recorded runs to have 52022 entries available. Further dropping `NaN` rows seems to have dropped this down to a further 1822 unique configurations with any recorded validation error. The number of available samples during the preprocessing steps were (2608019, 2608019, 2607078, 35544)

name	type	values	info
lr_decay_factor	continuous	$[0.0100221257, 0.988565263]$	
lr_initial	continuous	$[1.00276e\text{-}05, 9.8422475735]$	
lr_power	continuous	$[0.1004250993, 1.9985927056]$	
opt_momentum	continuous	$[5.86114e\text{-}05, 0.9989999746]$	
epoch	integer	$[1, 19]$	fidelity

Table 5: The `translatewmt_xformer_64` search space.

D.1.4 Yahpo-Gym

Tha Yahpo-Gym [Pfisterer et al., 2022] collection is a large collection of multi-fidelity surrogates across a wide range of tasks, including traditional machine learning models with dataset size as a fidelity as well as Neural Network benchmarks such as LCBench and NAS-Bench-301 [Siems et al., 2020]. For our experiments, we stick to 2 randomly selected LCBench tasks from 34 from OpenML [Bischl et al., 2019]. These are provided by a surrogates trained on a shared search space between these tasks. We ignore the rest of the available benchmarks from Yahpo-Gym as the others consist of mostly conditional search spaces which we do not account for in this work. For these benchmarks, we minimize `1 - val_balanced_accuracy`. These two tasks were chosen as they repesent the majority of the LCBench tasks (see fig 5 and most compliment the space of possible correlation curves as ssen in figure 3.

D.2 Baselines

Hyperband We implement our own version of `HyperBand`[Li et al., 2018] to allow for the input of priors. We verified our implementation with the `HyperBand` implementation provided in BOHB [Falkner et al., 2018b]. We use $\eta = 3$ for all experiments with the minimum and maximum budget coming as an input from the problem to solve, in this case, benchmarks.

Bayesian optimization is a prominent framework for Hyperparameter Optimization [Snoek et al., 2012, Kiili et al., 2020], hence we choose its GP implementation as a model-based competitor. We incorporate expert priors to the optimization following the πBO algorithm [Hvarfner et al., 2022]. In a low-budget setting, model-based search proves challenging for high-dimensional search spaces (e.g. JAHS-Bench-201) as common practices require the number of initial random observations equal to

name	type	values	info
batch_size	integer	$[16, 512]$	log
learning_rate	continuous	$[0.00010000000000000009, 0.10000000000000002]$	log
max_dropout	continuous	$[0.0, 1.0]$	
max_units	integer	$[64, 1024]$	log
momentum	continuous	$[0.1, 0.99]$	
num_layers	integer	$[1, 5]$	
weight_decay	continuous	$[1e\text{-}05, 0.1]$	
epoch	integer	$[1, 52]$	fidelity

Table 6: The `lcbench` search space.

the search space dimensionality. For our $10\times$ setup, to allow model-based search in BO and πBO we set their initial design size to 5.

BOHB [Falkner et al., 2018b] incorporates multi-fidelity `HyperBand` into the Bayesian optimization framework by building KDE models on each fidelity level to efficiently guide the search. The official implementation has no direct way of accepting a distribution over optimal configuration and incorporate it into search. We keep the other default settings in tact.

D.3 Correlation Curves

We provide a further look at the details of the benchmarks by providing correlation curves, where each point of a curve shows the spearman correlation of 25 configurations from the current fidelity to the last. These can vary quite significantly depending on the number of samples taken and the sample of configurations chosen. By increasing the number of samples, the correlation curve will tend to converge, but does not illustrate the variety of noisy curves that a given algorithm will experience during an optimization procedure.

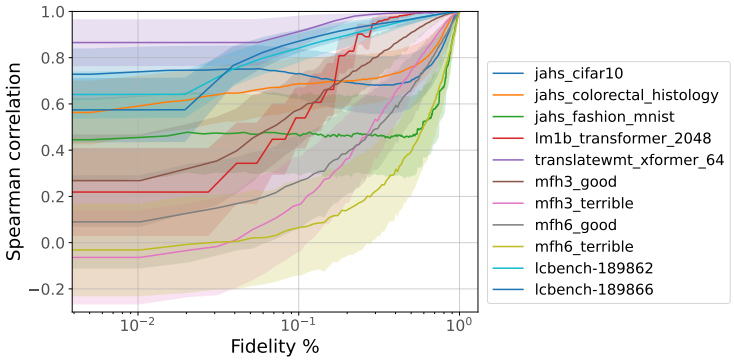
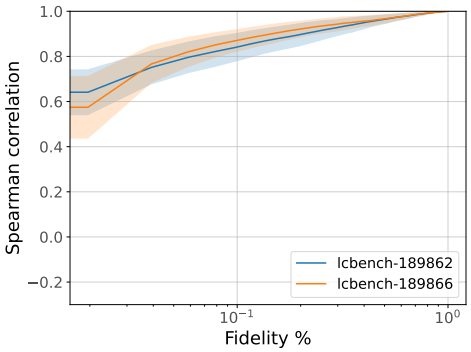


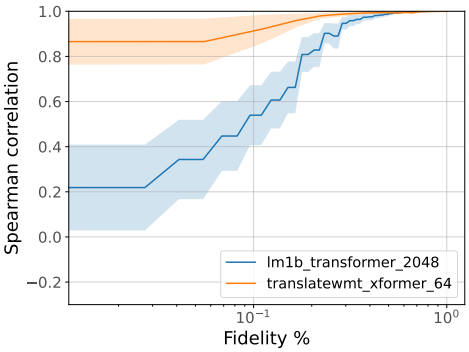
Figure 3: The correlation curves for all benchmarks used. The x-axis represents proportion of the total fidelity while the y-axis shows the correlation from the given fidelity percentage to the final fidelity.

To illustrate this fully, we perform a Monte Carlo estimation strategy, where we repeatedly generate these curves and calculate their mean, continuing until the mean has converged to being within 0.001 euclidean distance of the previous. We plot the final mean along with the standard deviation for each of the benchmarks.

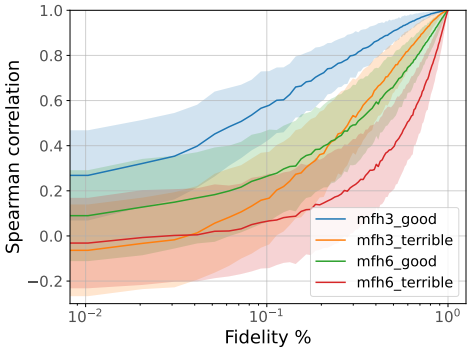
We also provide a look in figure 5 at how the correlation curves of the two LCBench tasks we selected compare to the other 34 that were available as part of Yahpo-Gym [Pfisterer et al., 2022]. We highlight that the correlation between even just 10% of the total fidelity budget and the final fidelity exhibit a very strong correlation with most having a correlation above 0.8.



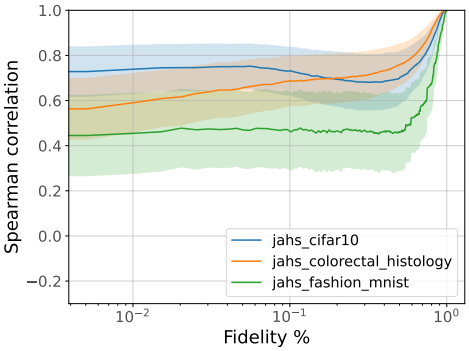
(a) LCBench



(b) PD1

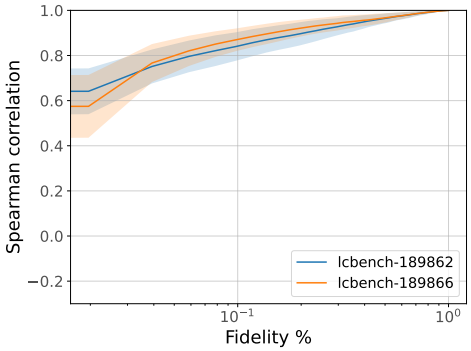


(c) Multi-fidelity synthetic Hartmann (MFH)

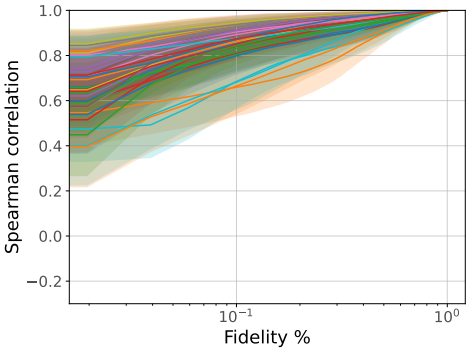


(d) JAHS-Bench-201

Figure 4: The classes of benchmarks used, separated out for comparison.



(a) Selected LCBench tasks



(b) All LCBench tasks

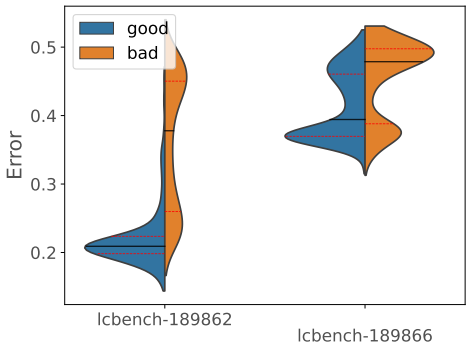
Figure 5: The correlation curves of the 2 selected LCBench tasks, 189862 and 189866 when compared to the total 34 tasks available.

D.4 Generating Priors

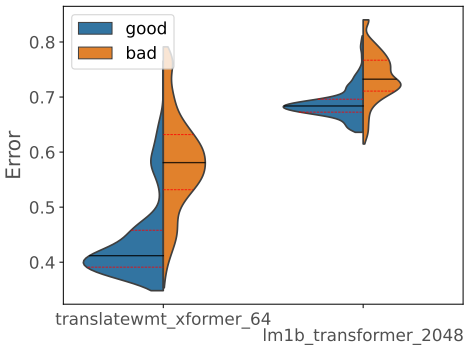
To generate both good and bad priors, we randomly sample 25 configurations from each benchmark. The two configurations with the lowest and highest error at the maximum fidelity are used as the good and bad prior respectively. For the Hartmann benchmarks, we instead *calculate* a good prior using the fact we have analytical access to compute the optimum configuration. To generate a *good* prior from this, we simply perturb the optimum, adding uniform noise in $[-0.25, 0.25]$ to each dimension.

To generate both good and bad prior *distributions*, we construct a multi-variate normal where the mean is centered at the prior configuration's hyperparameters and the standard deviations are set to 25% of each hyperparameter's total range. For example, if using a prior with a value of $x = 5$, $x \in [1, 10]$, the distribution arising from this prior would sample x from $\mathcal{N}(5, 2.5)$.

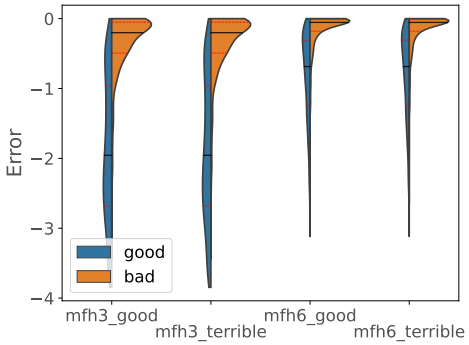
In figure 6 we present a violin plot, comparing how using the good and bad configurations as a distributional prior affects the errors of 1250 sampled configurations. The plot shows that for every benchmark, the density of errors for configurations sampled from the good prior distribution achieves a lower mean error when compared to drawing from the bad prior distribution. The one close exception is seen with the `jahs_fashion_mnist` benchmark, where both the good and bad prior distributions achieve a similar mean error. We note however, that still, using the good prior achieves a much tighter quartile range over that of using the bad prior.



(a) LCBench



(b) PD1



(c) Multi-fidelity synthetic Hartmann (MFH)

(d) JAHS-Bench-201

Figure 6: A violin plot showing showing the errors of 1250 samples from both the good and bad prior distribution. Each side of a *violin* has a distribution of equal area, the black line inside the violin shows the distribution's mean and the red dotted lines the 25%/75% quartiles.

E Additional experimental results

In Figure 7 we present the results for RQ3 in Section 3.

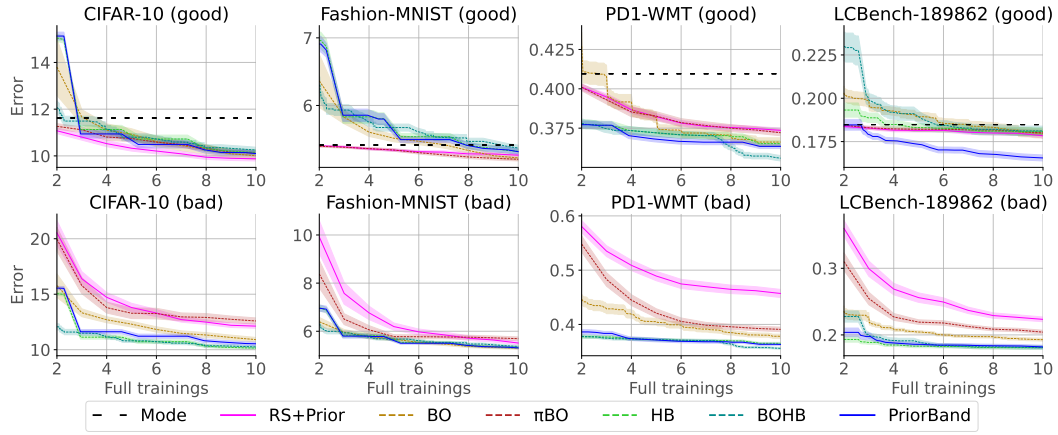


Figure 7: Comparative performance of `PriorBand`. We show the mean validation error over 50 seeds and the standard error band.

F Algorithm Details

F.1 Incumbent sampling

For the incumbent-based sampling, sample neighbors uniformly within a hyper-sphere of the incumbent. The radius is determined by a modified euclidean distance from the incumbent to its closest neighbor. Each continous and integer-valued variable is normalized to $[0, 1]$-range. For categorical variables, a scaled Hamming distance is utilized. For two categorical variable x_i^c and x_j^c, the distance between them is defined as

$$d(c_i, c_j) = \begin{cases} 1, & \text{if } x_i^c = x_j^c \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

and this metric is scaled by $1/\sqrt{|x^c|}$ to avoid having the average distance increase with a larger number of categorical options. As such, we can obtain a distance measure for each categorical variable in $\mathcal{X}$, and, along with the normalized continuous and integer-valued distance measures, we are able to compute euclidian distances across the search space.

To perform uniform sampling from the hyper-sphere around the incumbent, we utilize rejection sampling. As such, points are drawn from the entirety of the search space, and are discarded until we obtain a sample that is within the desired distance from $\boldsymbol{x}^*$.

F.2 Ablations

We run multiple ablations on PriorBand, and separate each of the novel components between runs. The following variants are run:

- `PriorBand` - the proposed approach with sampling as outlined in Algorithm 2.
- `PriorBand` without prior - the proposed approach, but replacing the samples from π with samples drawn uniformly at random.
- `PriorBand` without incumbent sampling - the proposed approach, but with no samples drawn around the incumbent. This equates to always to setting the incumbent number of samples to zero in Algorithm 2.

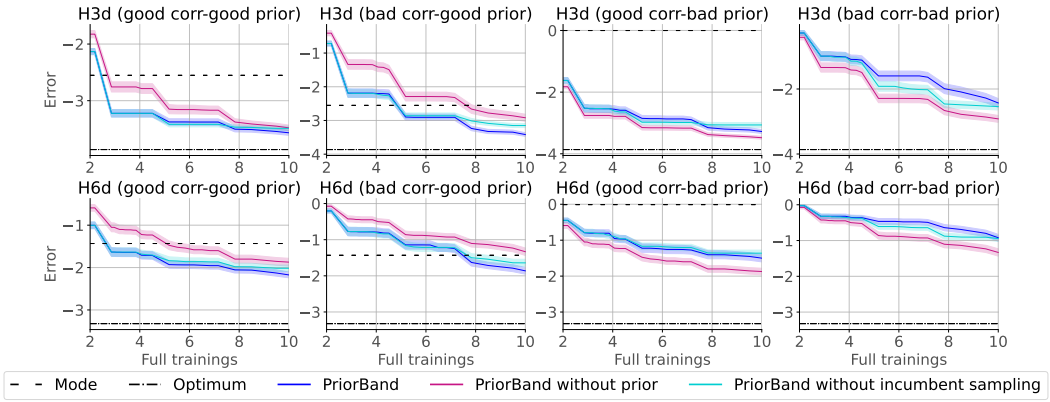


Figure 8: Ablations on the synthetic Hartmann-3 and Hartmann-6 functions.

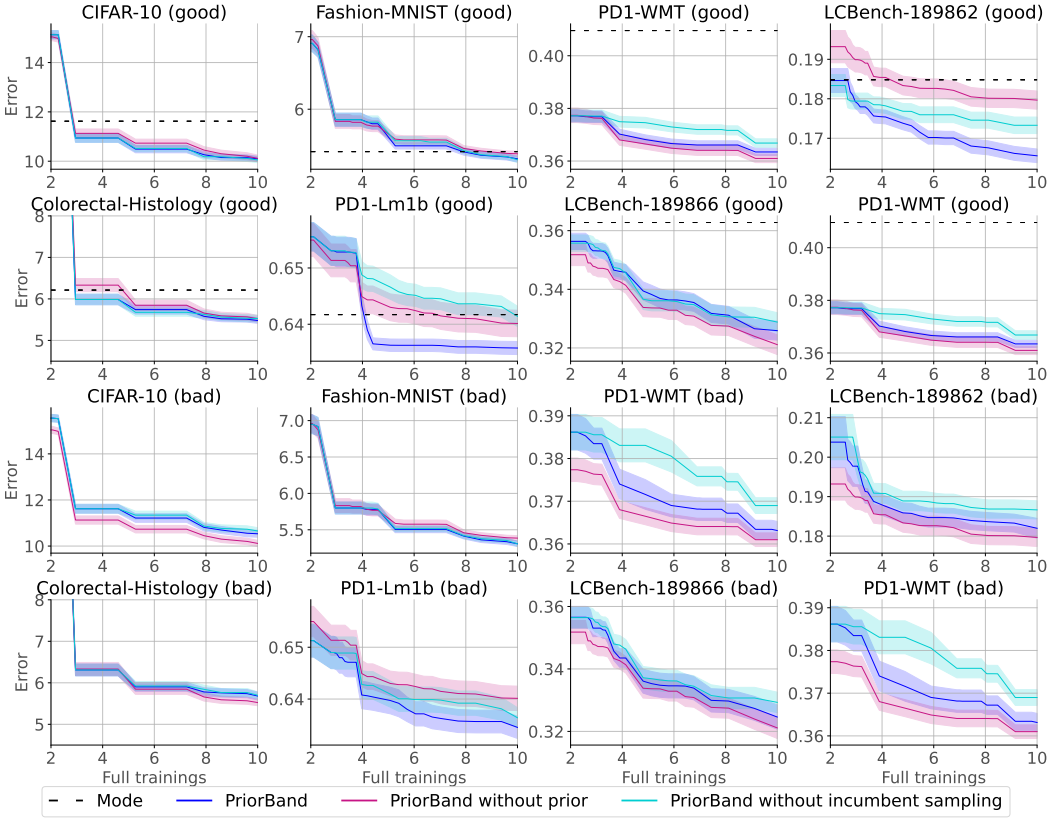


Figure 9: Ablations on the real world benchmarks.