Multi-objective Hyperparameter Optimization in the Age of Deep Learning

Soham Basu[1] **Danny Stoll**[1]

[1]University of Freiburg

Abstract While Deep Learning (DL) experts often have prior knowledge about which hyperparameter settings yield strong performance, very few Hyperparameter Optimization (HPO) algorithms exist that leverage such prior knowledge and none incorporate priors over multiple objectives. As DL practitioners often need to optimize not just one but many objectives, such as predictive performance, inference cost, fairness, or interpretability, this is a blind spot in the algorithmic landscape of HPO. To address this shortcoming, we introduce `PriMO`, the first HPO algorithm that integrates multi-objective user beliefs, and further, uses a novel initial design strategy tailored to DL tasks. `PriMO` achieves state-of-the-art performance across 8 DL benchmarks, comparing against prominent multi-objective baselines, and single-objective prior-based algorithms that we adapted to the multi-objective setting. Using statistical significance analysis, we underline `PriMO`'s performance gains under good prior conditions and superior recovery strength when prior knowledge is misleading.

1 Introduction

Modern Deep Learning (DL) pipelines [1–3] are highly sensitive to the choice of their hyperparameters, the manual tuning of which has become an increasingly time-consuming and costly task. Despite substantial advances in algorithms for Hyperparameter Optimization (HPO) [4–7], many researchers continue to rely on manual tuning [8], which allows intuitive incorporation of domain expertise and prior beliefs about the best performing hyperparameter settings.

While HPO researchers have formulated desiderata for HPO algorithms that include incorporating such user beliefs, existing research has focused exclusively on single-objective optimization [7, 9–11]. However, for DL, it is often necessary to optimize over several objectives, such as computational cost, training time, latency or fairness [12–15]. Thus, integrating prior knowledge into multi-objective optimization is a crucial research area that remains unexplored, and the desiderata in the existing literature incomplete. We therefore complete the desiderata for HPO algorithms for DL [6, 7, 16] as follows:

1. **Utilize cheap approximations**: Modern HPO algorithms must not only support optimization over multiple objectives, but should also be able to utilize cheap proxies of an objective function, if available, to speed up the optimization.

2. **Integration of MO expert priors**: Expert prior knowledge of hyperparameters is often available for real-world DL tasks. A modern multi-objective hyperparameter optimization (MO-HPO) algorithm must be able to integrate and properly utilize such beliefs over multiple objectives.

3. **Robustness to misleading priors**: Prior-based MO-HPO algorithms must also be able to meaningfully recover from misleading prior information.

4. **Strong anytime performance**: MO-HPO algorithms should identify promising configurations early in the optimization.

 © 2025 the authors, released under CC BY 4.0

Table 1: Comparison of PriMO and prominent baselines with respect to the identified desiderata. A ✓ indicates that the method satisfies the criterion; a ✗ indicates it does not. ✓* denote partial fulfillment or fulfillment with additional assumptions.

Criterion	RS	EA	RS+MOMF	MF-BO	MO-BO	BO+Prior	MF+ Prior	PriMO
Good anytime performance	✗	✗	✓	✓	✗	✗	✓	✓
Good final performance	✗	✓*	✓*	✓	✓	✓	✓	✓
Utilize cheap approximations	✗	✗	✓	✓	✗	✗	✓	✓
Multi-objective	✓	✓	✓	✗	✓	✗	✗	✓
Compute Efficient for MO	✗	✗	✗	✗	✓	✗	✗	✓
MO Expert priors	✗	✗	✗	✗	✗	✗	✗	✓

5. **Strong final performance**: The ultimate goal of HPO is to find the best performing configurations with larger budgets, as these are the most relevant for deployment.
6. **Efficiency**: MO-HPO algorithms must be compute-efficient, *i.e.*, under limited budget, they must find candidates that *significantly* improve the dominated Hypervolume.
7. **Simplicity**: MO-HPO algorithms should be conceptually simple and easy to apply to real-world DL problems. Such simplicity is essential for practitioners to understand and adopt these methods, as well as for HPO tools to implement them.

Table 1 shows that the existing HPO algorithms fulfill at most half of the criteria. To address this gap, we propose PriMO, which is the first HPO algorithm to incorporate expert knowledge over the optima of multiple objectives, while also employing a novel initial design to leverage cheap approximations of expensive objective functions. Our contributions are as follows:

1. We are the first to incorporate **user-provided prior distributions over multiple objectives**.
2. We employ a **novel initial design strategy** tailored to DL, which uses cheap approximations to speed up the optimization process for the subsequent model and prior-based strategy.
3. We empirically demonstrate **state-of-the-art performance** of PriMO across a variety of DL benchmarks, and perform statistical significance analysis to support our results.
4. We show that PriMO **effectively leverages good priors**, and is **robust to misleading priors**.

2 Problem Statement

To capture all the above desiderata, we consider the minimization of a vector-valued objective function f, while exploiting cheap approximations of its individual objectives and expert priors. In this section we define our problem setup. For related works and further background see Section 5 and Appendix A.

To extend expert priors over a single objective [11] to the MO setting, we consider a factorized prior as follows. For each objective f_i of the vector-valued function f, prior beliefs $\pi_{f_i}(\lambda)$ represent a probability distribution over the location of the optimum of f_i. Specifically, the prior will have a high value in regions that the user believes have an optimum. Formally, we define $\pi_{f_i}(\lambda) = \mathbb{P}\left(f_i(\lambda) = \min_{\lambda' \in \Lambda} f_i(\lambda')\right)$, yielding the compound prior $\Pi_f(\lambda) = \left\{\pi_{f_i}(\lambda)\right\}_{i=1}^{m}$.

To also leverage cheap approximations of the individual objectives, let $\hat{f}_i(\lambda, z)$ denote the low-fidelity proxy for f_i, where configuration λ is evaluated at the fidelity level z, where $f_i(\lambda) = \hat{f}_i(\lambda, z_{max})$. Therefore, our goal is to solve

$$\arg\min_{\lambda \in \Lambda} f(\lambda) = \arg\min_{\lambda \in \Lambda} \left(\hat{f}_1(\lambda, z_{max}), ..., \hat{f}_n(\lambda, z_{max})\right), \quad \text{guided by } \Pi_f(\lambda), \tag{1}$$

using inexpensive evaluations of f, while addressing the challenge that the priors may be misleading.

3 PriMO: Prior Informed Multi-objective Optimizer

In this section we introduce the first multi-objective (MO) HPO algorithm, PriMO, that leverages MO user priors and fulfills all the desiderata of modern HPO. We discuss how PriMO, a Bayesian Optimization algorithm, makes use of MO user priors via its acquisition function and also introduce its novel initial design strategy. We provide additional details and pseudo code in Appendix B.

We weight the acquisition function of the BO with the the PDF of the selected priors, raised to an exponent $\gamma = exp\left(-n_{\text{BO}}^2/n_d\right)$. To formulate an acquisiton function, we convert the vector-valued objective function into a single-objective optimization problem, using a linear scalarization function [17] with randomly sampled weights, which is not only simple but also scalable with the number of objectives. Furthermore, to aid in recovery from misleading priors, we additionally incorporate a simple exploration parameter ϵ, which controls how often we augment the acquisition function with the prior. Thus, with priors over n objectives, the acquisition function becomes

$$\alpha_{\epsilon\pi}(\lambda, \mathcal{D}_m) \triangleq \begin{cases} \alpha(\lambda, \mathcal{D}_m), & \text{with prob. } \epsilon \\ \alpha(\lambda, \mathcal{D}_m) \cdot \pi_j(\lambda)^{exp\left(-n_{\text{BO}}^2/n_d\right)}, & \text{with prob. } 1-\epsilon,\ j \sim \mathcal{U}(1, \ldots, n) \end{cases} . \tag{2}$$

To leverage cheap approximations of the objective functions, we propose a novel initial design strategy that is similar to multi-fidelity optimization to sample initial seed points to speed up the optimization process. First, we set a threshold of (equivalent) full function evaluations based on the initial design size, for which PriMO's initial design strategy is run. Once the threshold is reached, only the samples at the maximum fidelity are selected for the BO. Next, we choose one of the priors over multiple objectives, uniformly at random, during each iteration. The sampled initial points then kickstart the BO along with the decaying prior-augmented acquisition function.

4 Experiments

To fulfill the desiderata outlined in the Introduction, we address the following research questions:

RQ1: Does PriMO outperform prominent baselines in terms of anytime and final performance?

RQ2: Does PriMO's novel initial design result in performance gains compared to random sampling?

RQ3: Can PriMO effectively leverage multi-objective expert priors?

RQ4: Does PriMO recover from misleading priors and maintain its robustness?

RQ5: Are all components of PriMO necessary and helpful?

State-of-the-art performance. The relative rankings (Figure 1, left) show that PriMO effectively leverages good-good priors, is the best algorithm initially and significantly outperforms all baselines, except πBO+RW, towards the end of the optimization (**RQ1**). We notice that compared to BO+RW and πBO+RW which use random sampling and prior-based sampling for their initial designs respectively, PriMO's initial relative ranking is significantly better (**RQ2**), which is further utilized by the MO-prior-weighted BO phase, leading to strong final performance (**RQ3**). In Appendix J we perform a significance analysis that confirms these results. We show detailed experimental results using dominated Hypervolume and Pareto front plots in Appendix G.

Robustness to prior conditions. Under overall-bad priors in Figure 1 (middle), we see that PriMO is already the best early on and achieves the best relative ranking at the end of the optimization run. This highlights PriMO's ability to effectively recover in the presence of misleading priors (**RQ4**). The relative ranking plot in Figure 1 (right) shows that throughout the optimization run, PriMO remains the best ranked HPO algorithm under all (overall-all) prior conditions.

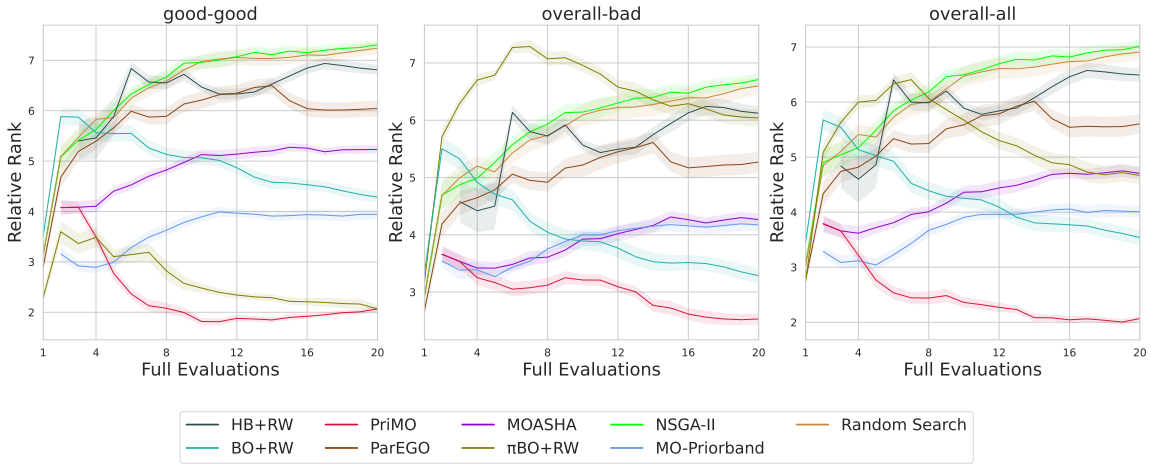

Figure 1: Relative ranking averaged across 9 benchmarks comparing PriMO against the baselines, under various prior conditions. **Left**: Relative ranks when priors for both objectives are helpful. **Middle**: Relative ranks when the prior for one or both objectives are misleading. **Right**: Relative ranks under all priors averaged.

Ablation study. We perform an ablation study in Appendix I.1 that shows that the components of PriMO indeed work best in conjunction with each other and support the design of PriMO (**RQ5**).

5 Related Work

MO optimization traditionally considers large budgets [18–21] however, DL is very expensive and can only make use of few evaluations. To make HPO for DL feasible, specialized strategies have been explored. Exploiting user priors, have been explored in a few works, but only for the single-objective optimization case. Most similar to our approach (albeit for single-objective optimization) is Priorband [7], which, in addition, to exploiting user priors also makes use of cheap approximations, in contrast to us, they use these throughout the optimization and not as an initial design. We explored adapting Priorband to the MO setting, but found it does not perform well (Section 4). πBO [11], like PriMO, also augments the acquisition function with the priors, although it does so for a single objective only, can not utilize cheap approximations, and adapting it directly to the MO setting does not perform well under misleading priors (Section 4). While exploiting expert priors is novel for multi-objective HPO, cheap proxies have been explored [13, 14, 22], again, not as an initial design and our approach outperforms these by a wide margin (Section 4). We discuss background and related work in more detail in Appendix A.

6 Conclusion and Limitations

PriMO distinguishes itself as the first algorithm to integrate MO priors, leading to state-of-the-art performance. As such, PriMO is, to date, the only HPO algorithm that fulfills all the desiderata of modern HPO, making it fit for efficient optimization under constrained budgets for practical DL.

Limitations. In line with previous work [7, 10, 11], we only consider Gaussian distribution for our priors, although PriMO supports priors with any distribution. While it may be more beneficial in the MO setting to generate priors based on an approximate Pareto front, this remains a non-trivial challenge. Additionally, instead of linear scalarization, an approach such as *Hypervolume scalarization* [21] could be beneficial to PriMO as it has provable guarantees.

Acknowledgements. Robert Bosch GmbH is acknowledged for financial support. We also acknowledge funding by the European Union (via ERC Consolidator Grant DeepLearning 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.

Funded by the European Union

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*. Curran Associates, Inc., 2017.

[2] J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. A. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In Larochelle et al. [23], pages 1877–1901.

[4] 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 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'11)*, pages 2546–2554. Curran Associates, 2011.

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

[6] 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, 2018.

[7] N. Mallik, C. Hvarfner, E. Bergman, D. Stoll, M. Janowski, M. Lindauer, L. Nardi, and F. Hutter. PriorBand: Practical hyperparameter optimization in the age of deep learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*. Curran Associates, 2023.

[8] X. Bouthillier and G. Varoquaux. Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020. Research report [hal-02447823], Inria Saclay Ile de France, 2020.

[9] A. Ramachandran, S. Gupta, S. Rana, C. Li, and S. Venkatesh. Incorporating expert prior in Bayesian optimisation via space warping. *Knowledge-Based Systems*, 195, 2020.

[10] A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Prior-guided Bayesian optimization. In Larochelle et al. [23].

[11] C. Hvarfner, D. Stoll, A. Souza, L. Nardi, M. Lindauer, and F. Hutter. πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. In *The Tenth International Conference on Learning Representations (ICLR'22)*. ICLR, 2022. Published online: iclr.cc.

[12] S. Izquierdo, J. Guerrero-Viu, S. Hauns, G. Miotto, S. Schrodi, A. Biedenkapp, T. Elsken, D. Deng, M. Lindauer, and F. Hutter. Bag of baselines for multi-objective Joint Neural Architecture Search and Hyperparameter Optimization. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.

[13] R. Schmucker, M. Donini, V. Perrone, M. Zafar, and C. Archambeau. Multi-objective multi-fidelity hyperparameter optimization with application to fairness. In R. Calandra, J. Clune, E. Grant, J. Schwarz, J. Vanschoren, F. Visin, and J. Wang, editors, *NeurIPS 2020 Workshop on Meta-Learning*, 2020.

[14] David Salinas, Valerio Perrone, Olivier Cruchant, and Cedric Archambeau. A multi-objective perspective on jointly tuning hardware and hyperparameters, June 2021. URL http://arxiv.org/abs/2106.05680. ICLR Workshop on Neural Architecture Search.

[15] Lennart Schneider, Bernd Bischl, and Janek Thomas. Multi-objective optimization of performance and interpretability of tabular supervised machine learning models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 538–547, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590380. URL https://doi.org/10.1145/3583131.3590380.

[16] Luca Franceschi, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. Hyperparameter Optimization in Machine Learning, April 2025. URL http://arxiv.org/abs/2410.22854. arXiv:2410.22854 [stat].

[17] Min Yoon, Yeboon Yun, and Hirotaka Nakayama. *Sequential Approximate Multiobjective Optimization Using Computational Intelligence*. Vector Optimization. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-88909-0 978-3-540-88910-6. doi: 10.1007/978-3-540-88910-6. URL https://link.springer.com/10.1007/978-3-540-88910-6.

[18] K. Deb. Multi-objective optimization. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 403–449. Springer, 2013.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. ISSN 1941-0026. doi: 10.1109/4235.996017. URL https://ieeexplore.ieee.org/document/996017.

[20] J. D. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1): 50–66, 2006.

[21] Daniel Golovin and Qiuyi Zhang. Random Hypervolume Scalarizations for Provable Multi-Objective Black Box Optimization, June 2020. URL http://arxiv.org/abs/2006.04655. arXiv:2006.04655 [cs].

[22] R. Schmucker, M. Donini, M. Zafar, D. Salinas, and C. Archambeau. Multi-objective asynchronous successive halving. *arXiv:2106.12639 [stat.ML]*, 2021.

[23] H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors. *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020. Curran Associates.

[24] 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.

[25] 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.

[26] N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyperband 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.

[27] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems 2*, volume 2, 2020.

[28] N. Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, September 1994. ISSN 1063-6560. doi: 10.1162/evco.1994.2.3.221. URL https://doi.org/10.1162/evco.1994.2.3.221. _eprint: https://direct.mit.edu/evco/article-pdf/2/3/221/1492770/evco.1994.2.3.221.pdf.

[29] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/TEVC.2007.892759.

[30] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — A comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, pages 292–301, Berlin, Heidelberg, 1998. Springer. ISBN 978-3-540-49672-4. doi: 10.1007/BFb0056872.

[31] Yaochu Jin and Janusz Kacprzyk, editors. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, Berlin, Heidelberg, 2006. ISBN 978-3-540-30676-4 978-3-540-33019-6. doi: 10.1007/3-540-33019-4. URL http://link.springer.com/10.1007/3-540-33019-4.

[32] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A Flexible Framework for Multi-Objective Bayesian Optimization using Random Scalarizations, June 2019. URL http://arxiv.org/abs/1805.12168. arXiv:1805.12168 [cs].

[33] M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. PhD thesis, Universität Dortmund, 2005.

[34] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[35] Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. Multi-Objective Bayesian Global Optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, 44:945–956, February 2019. ISSN 2210-6502. doi: 10.1016/j.swevo.2018.10.007. URL https://www.sciencedirect.com/science/article/pii/S2210650217307861.

[36] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization, October 2020. URL http://arxiv.org/abs/2006.05078. arXiv:2006.05078 [stat].

[37] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In J. Ceberio, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20)*, page 533–541. ACM Press, 2020.

[38] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted $\mathcal{S}$-Metric Selection. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 784–794, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-87700-4. doi: 10.1007/978-3-540-87700-4_78.

[39] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[40] S. Belakaria, A. Deshwal, and J. Doppa. Max-value entropy search for multi-objective Bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, and R. Garnett, editors, *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'19)*. Curran Associates, 2019.

[41] D. Hernández-Lobato, J. Hernández-Lobato, A. Shah, and R. Adams. Predictive Entropy Search for Multi-objective Bayesian Optimization. In M. Balcan and K. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning (ICML'17)*, volume 48, pages 1492–1501. Proceedings of Machine Learning Research, 2016.

[42] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[43] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Multi-Fidelity Multi-Objective Bayesian Optimization: An Output Space Entropy Search Approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(06):10035–10043, April 2020. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v34i06.6560. URL https://arxiv.org/pdf/2011.01542.

[44] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Information-theoretic multi-objective bayesian optimization with continuous approximations, 2020. URL https://arxiv.org/abs/2009.05700.

[45] Faran Irshad, Stefan Karsch, and Andreas Döpp. Leveraging Trust for Joint Multi-Objective and Multi-Fidelity Optimization. *Machine Learning: Science and Technology*, 5(1):015056, March 2024. ISSN 2632-2153. doi: 10.1088/2632-2153/ad35a4. URL http://arxiv.org/abs/2112.13901. arXiv:2112.13901 [cs].

[46] Danny Stoll, Neeratyoy Mallik, Eddie Bergman, Simon Schrodi, Samir Garibov, Tarek Abou Chakra, Timur Carstensen, Maciej Janowski, Gopalji Gaur, Anton Merlin Geburek, Daniel Rogalla, Carl Hvarfner, Ru Binxin, and Frank Hutter. Neural Pipeline Search (NePS), April 2025. URL https://github.com/automl/neps. original-date: 2021-06-15T19:15:11Z.

[47] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/419f72cbd568ad62183f8132a3605a2a-Abstract-Conference.html.

[48] D. Salinas, M. Seeger, A. Klein, V. Perrone, M. Wistuba, and C. Archambeau. Syne Tune: A library for large scale hyperparameter tuning and reproducible research. In I. Guyon, M. Lindauer, M. van der Schaar, F. Hutter, and R. Garnett, editors, *Proceedings of the First International Conference on Automated Machine Learning*. Proceedings of Machine Learning Research, 2022.

[49] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, December 2020. URL http://arxiv.org/abs/1910.06403. arXiv:1910.06403 [cs].

[50] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. YAHPO Gym – An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization, July 2022. URL http://arxiv.org/abs/2109.03670. arXiv:2109.03670 [cs].

[51] Zi Wang, George E. Dahl, Kevin Swersky, Chansoo Lee, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained Gaussian Processes for Bayesian Optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024. URL http://jmlr.org/papers/v25/23-0269.html.

[52] Jeong Soo Park. *Tuning complex computer codes to data and optimal designs*. text, University of Illinois at Urbana-Champaign, 1991. URL https://hdl.handle.net/2142/21851.

[53] Y. Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

[54] Eddie Bergman, Neeratyoy Mallik, Carl Hvarfner, and Soham Basu. mf-prior-bench, March 2025. URL https://github.com/automl/mf-prior-bench.

[55] S. Zagoruyko and N. Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the 27th British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

[56] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR'16)*, pages 770–778. Computer Vision Foundation and IEEE Computer Society, IEEE, 2016.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[59] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://aclanthology.org/2021.tacl-1.4/.

[60] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. URL https://arxiv.org/abs/1312.3005.

[61] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

[62] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 workshop on statistical machine translation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3001. URL https://aclanthology.org/W15-3001/.

[63] L. Zimmer, M. Lindauer, and F. Hutter. Auto-Pytorch: Multi-fidelity metalearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3079–3090, 2021.

[64] J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2014.

[65] Stefan Riezler and Michael Hagmann. *Validity, Reliability, and Significance: Empirical Methods for NLP and Data Science*. Synthesis Lectures on Human Language Technologies. Springer International Publishing, Cham, 2022. ISBN 978-3-031-01055-2 978-3-031-02183-1. doi: 10.1007/978-3-031-02183-1. URL https://link.springer.com/10.1007/978-3-031-02183-1.

[66] Anton Geburek, Neeratyoy Mallik, Danny Stoll, Xavier Bouthillier, and Frank Hutter. Lmems for post-hoc analysis of hpo benchmarking, 2024.

[67] John W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5 2: 99–114, 1949. URL https://api.semanticscholar.org/CorpusID:806596.

Submission 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] Yes, Section 3 details the algorithm design and Section 4 presents and discusses the performance of `PriMO` against multi-objective baselines, providing empirical proof of fulfilling the claims made (Table 1).

   (b) Did you describe the limitations of your work? [Yes] See Section 6.

   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? (see `https://2022.automl.cc/ethics-accessibility/`) [Yes]

2. If you ran experiments…

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources, etc.)? [Yes] See Appendix F and Appendix L.

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning details and results, etc.)? [Yes] See Appendix C, D, E, and F.

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] See Appendix F.

   (d) Did you report the uncertainty of your results (e.g., the standard error across random seeds or splits)? [Yes] See Section 4 and Appendix G.

   (e) Did you report the statistical significance of your results? [Yes] See Appendix J.

   (f) Did you use enough repetitions, datasets, and/or benchmarks to support your claims? [Yes]

   (g) Did you compare performance over time and describe how you selected the maximum runtime? [Yes]

   (h) 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 L.

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] See Appendix I.1 and I.2.

3. With respect to the code used to obtain your results…

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all dependencies (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation instructions, and execution commands (either in the supplemental material or as a URL)? [Yes] See Appendix K.

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] In the experiment repository provided in Appendix K.

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] See Appendix K.

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] See Appendix K.

4. If you used existing assets (e.g., code, data, models)…

   (a) Did you cite the creators of used assets? [Yes]

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] See Appendix M.

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you created/released new assets (e.g., code, data, models)…

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] See Appendix M.

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] See Appendix K.

6. 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]

7. If you included 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]

A Background and more related work

A.1 Hyperparameter optimization for Deep Learning

Multi-fidelity optimization. The high computational cost of DL model evaluations has motivated research in multi-fidelity optimization. Multi-fidelity (MF) [24] optimizers use *cheap proxies* to approximate promising candidates and speed up the search. Bandit-based methods [5, 25] are the most popular in the Automated Machine Learning community for multi-fidelity optimization. These have been further extended by replacing their Random Search (RS) component with evolutionary [26] and model-based Falkner et al. [6] search, and increasing efficiency for large-scale parallelization Li et al. [27].

Instead of optimizing the expensive objective function f as a blackbox, multi-fidelity optimization leverages evaluations of f at lower fidelities. For example, when training a Neural network with a particular hyperparameter configuration for 100 epochs, a lower-fidelity proxy would be the validation score obtained by training the model with the same hyperparameter configuration for 15 epochs. More formally, for a hyperparameter configuration $\lambda \in \Lambda$ at a fidelity level $z \in Z$ where $Z \coloneqq \{z_{min}, ..., z_{max}\}, |Z| = m$ is the fidelity space, a cheap proxy function of f is defined as $\hat{f}(\lambda, z)$. Therefore, when $z = z_{max}$ (the maximum fidelity), the proxy function $\hat{f}$ converges to the true objective function f. Hence, $f = \hat{f}(\lambda, z_{max})$.

In an optimization setup with *continuations*, the function evaluation $\hat{f}(\lambda, z)$ for a configuration λ at fidelity z can be continued up to a fidelity z' to yield $\hat{f}(\lambda, z')$, given $z < z'$. For example, let us assume that we would like to train a network with a hyperparameter configuration λ for a total of 200 epochs, and have already trained it with λ for 50 epochs. Then we can simply continue training with λ for 150 more epochs instead of restarting from scratch. For such a continual setup, we define *equivalent function evaluations* as z/z_{max}.

Prior-based optimization for a single objective. Prior-based single objective optimization can be defined as:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} f(\lambda), \quad \text{guided by } \pi(\lambda), \tag{3}$$

where prior $\pi(\lambda)$ is a probability distribution over the location of the optimum of the objective function f.

PrBO [10] combines expert prior distributions $P_g(\lambda)$ and $P_b(\lambda)$ with respective models $M_g(\lambda)$ and $M_b(\lambda)$ in a Tree-structured Parzen Estimator (Bergstra et al. [4]) (TPE)-based approach to construct *pseudo-posteriors* $g(\lambda)$ and $l(\lambda)$ respectively. The candidates are then chosen from these pseudo-posteriors by maximizing the EI as described in Bergstra et al. [4]. πBO [11] directly augments the acquisition function α with the unnormalized user-specified prior distribution $\pi(\lambda)$ which decays over time, controlled by a parameter β: $\alpha_\pi^n(\lambda) = \alpha(\lambda) \cdot \pi(\lambda)^{\frac{\beta}{n}}$, where n refers to the nth iteration. Unlike PrBO, πBO generalizes to acquisition functions other than EI and offers convergence guarantees. However, as we saw see in Section 4, πBO's longer dependence on the Priors has major downsides. `PriMO` addresses this issue using a novel MO-priors-based augmentation of the BO component that we introduced in Section 3.

Mallik et al. [7] introduce Priorband which extends the integration of expert priors to multi-fidelity optimization. Priorband uses a novel *ensemble sampling policy (ESP)* $\mathcal{E}_\pi$, which combines random sampling $\mathcal{U}(\cdot)$, prior-based sampling $\pi(\cdot)$ and incumbent-based sampling $\hat{\lambda}(\cdot)$, with their proportions denoted by $p_\mathcal{U}$, p_π and $p_{\hat{\lambda}}$ respectively. Initially, $\hat{\lambda}(\cdot)$ is inactive. Given the constraint $p_\mathcal{U} + p_\pi = 1$, $\mathcal{E}_\pi$ selects from $\mathcal{U}(\cdot)$ and $\pi(\cdot)$ according to $p_\mathcal{U}$ and p_π. When $\hat{\lambda}(\cdot)$ becomes active, p_π is split into p_π and $p_{\hat{\lambda}}$ according to weighted scores $\mathcal{S}_\pi$ and $\mathcal{S}_{\hat{\boldsymbol{\lambda}}}$, calculated by first computing the likelihood of the top performing configurations under $\pi(\cdot)$ and $\hat{\lambda}(\cdot)$, which capture how much *trust* should be placed on each.

While the aforementioned algorithms efficiently integrate user priors in the HPO problem, they only apply to the single-objective optimization case. To the best of our knowledge, we are the first to incorporate priors over multiple objectives, whilst also employing a novel initial design strategy to leverage cheap proxies of the objective function.

A.2 Multi-objective optimization

For many real-world problems we are often interested in optimizing not one, but multiple, potentially competing objectives. MO [19, 20, 28, 29] deals with optimizing a *vector-valued objective function* $f(\lambda)$ composed of n distinct objective functions, where $f : \lambda \rightarrow \mathbb{R}^n$, $\lambda \in \mathbb{R}$. Without loss of generality, we assume minimization of all objectives. More formally, the MO problem can be defined as:

$$\arg\min_{\lambda \in \Lambda} f(\lambda) = \arg\min_{\lambda \in \Lambda} (f_1(\lambda), f_2(\lambda), ..., f_n(\lambda)) \quad . \tag{4}$$

Pareto optimality. Typically, there does not exist a single best solution for MO problems that minimizes all the objectives simultaneously. Rather, there exists a set of solutions, consisting of points in the domain Λ.

Given two candidates $\lambda_1, \lambda_2 \in \Lambda$, we say that λ_2 dominates λ_1 *if and only if* $f(\lambda_2) < f(\lambda_1)$. Formally, we write $\lambda_2 \prec \lambda_1$. For $f(\lambda_2) \leq f(\lambda_1)$, we write $\lambda_2 \preceq \lambda_1$ and say that λ_1 weakly dominates λ_2.

For a vector-valued function f, we say that λ_2 Pareto dominates λ_1, i.e. $\lambda_2 \prec \lambda_1$ under two conditions:

- $\forall i \in \{1, ..., n\} : f_i(\lambda_2) \leq f_i(\lambda_1)$, and,
- $\exists k \in \{1, ..., n\} : f_k(\lambda_2) < f_k(\lambda_1)$.

A candidate λ that is not dominated by any other candidate λ' is called *Pareto Optimal*, and the set of Pareto Optimal candidates is known as the *Pareto Set* $\mathcal{P}$, defined as:

$$\mathcal{P} := \left\{ \lambda \in \Lambda \,\middle|\, \nexists \lambda' \in \Lambda \text{ with } f(\lambda') < f(\lambda) \right\} \quad . \tag{5}$$

The set of solutions, i.e., set the corresponding values of an MO function for each of the Pareto Optimal candidates is called the *Pareto Front*. Formally, a Pareto front is defined as:

$$\mathcal{F} = \left\{ f(\lambda) \in \mathbb{R}^n \,\middle|\, \lambda \in \Lambda, \nexists \lambda' \in \Lambda \text{ with } f(\lambda') < f(\lambda) \right\} \quad . \tag{6}$$

Hypervolume indicator. The true Pareto front of a real-world MO problem is generally unknown. Thus, the goal of MO Optimization algorithms is to return a set of non-dominated candidates from which we can obtain an *approximated Pareto front*. To assess the quality of this approximation, the *S-Metric* or *Hypervolume (HV) Indicator* [30] is the most frequently used measure as it does not require prior knowledge of the true Pareto front.

Given a reference point r and an approximate Pareto set $\mathcal{A}$, the Hypervolume Indicator $\mathcal{H}$ is defined as:

$$\mathcal{H}_r(\mathcal{A}) = \mu\left(\{x \in \mathbb{R}^n | \exists a \in A : a \leq x \cap x \leq r\}\right) \quad , \tag{7}$$

where μ is the Lebesgue measure. Throughout this work, for our experiments, we will be using the *Hypervolume Improvement* (*HVI*) metric as a cumulative performance indicator for MO algorithms with respect to function evaluations. Given a new set of candidates γ, an existing Pareto set $\mathcal{P}$ and a reference point r, the *HVI* is formally defined as:

$$HVI(P, r, \gamma) = \mathcal{H}_r(P \cup \gamma) - \mathcal{H}_r(P) \quad . \tag{8}$$

A.3 Multi-objective optimization for Deep Learning

For DL, it is often necessary to optimize not only the validation error (or validation accuracy) but also a cost metric, such as the inference time of a Neural Network or Floating Point Operations per Second. It is easy to imagine that a cost metric would be cheap to evaluate since it is a simple observation, unlike an objective such as accuracy (which would require the network to be trained first) [12]. Additionally, we might also be interested in a third objective like fairness or interpretability of the DL model. However, from a DL perspective, optimizing predictive performance typically (but not always) comes at the cost of degrading other objectives. In the context of Machine Learning, multi-objective algorithms for HPO [31] have been adapted mainly from the general MO literature.

Scalarization-based Bayesian Optimization. Scalarization-based multi-objective Bayesian Optimization (MO-BO) approaches proposed by [17, 20, 21, 32] use a function: $s : \mathbb{R}^n \times \alpha \mapsto \mathbb{R}$ that maps the vector-valued MO function into a scalar value, thus effectively converting the MO problem into a single-objective optimization problem. These approaches vary in the choice of the scalarization function [20] or the distribution from which the weights are sampled [17, 32]. Knowles [20] introduces *ParEGO*, which uses a Tchebycheff norm over the objective values as opposed to a linear weighted sum approach. These methods are highly scalable and easy to implement, which is why we employ random scalarizations during the BO phase of `PriMO`.

Multi-objective Bayesian Optimization using acquisition function modifications. Other MO-BO approaches directly modify the acquisition function in BO to account for multiple objectives. Emmerich [33] proposed the Expected Hypervolume Improvement (EHVI) acquisition function wherein a surrogate model is fitted for each objective separately, and then the Expected Improvement (EI) [34] of the HV contribution is calculated. Several improvements to calculate EHVI have been proposed, such as in Yang et al. [35] and Daulton et al. [36]. EHVI is also used in Ozaki et al. [37] to extend the TPE [4] to MO TPE. Ponweiser et al. [38] introduced the S-Metric Selection-based EGO (SMS-EGO) which, instead of using EHVI, selects new candidates by directly maximizing the HV contribution based on the predictions of the surrogate model, using the Lower Confidence Bound [39] acquisition function. Izquierdo et al. [12] modified EHVI by fitting surrogate models only on the expensive objectives, such as validation accuracy. MO Information-theoretic acquisition functions, such as maximum entropy search [40] (MESMO) and predictive entropy search [41] (PESMO), aim to reduce the entropy of the location of the Pareto front.

Evolutionary algorithms. Evolutionary MO Algorithms mutate configurations from a diverse initial population to identify promising candidates closer to the Pareto front. Deb et al. [19] proposed the popular Non-dominated Sorting Genetic Algorithm (NSGA-II) which uses non-dominated sorting [28] to rank candidates from multiple non-dominated fronts and conducts survival selection (tie-breaking) using crowding-distance sort [19]. S-Metric Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) [42] also employs non-dominated sorting from [28] and [19] for the initial ranking of candidates, but then uses each candidate's contribution to the dominated HV for survival selection. Evolutionary methods, however, are quite compute-inefficient, requiring a high budget to significantly improve the dominated HV. Compute efficiency is one of the desiderata we identify in Table 1 and therefore is a key aspect of `PriMO`.

Multi-objective multi-fidelity optimization. Izquierdo et al. [12] extended SMS-EMOA to the MF domain by augmenting it with SH rungs. Furthermore, they introduced MO-BOHB, which replaced the TPE component with MO-TPE. Schmucker et al. [13] adapted HyperBand (HB) to MO using a randomly scalarized objective value (HB+RW) to select and promote promising configurations. Salinas et al. [14] and Schmucker et al. [22] further build on [13] by modifying the promotion strategy of HB and ASHA respectively, using non-dominated sorting for the initial ranking of candidates, and a greedy `epsilon-net` (ϵ-net) strategy for exploration.

MF-OSEMO (Belakaria et al. [43]) and iMOCA(Belakaria et al. [44]) extend the information-theoretic method MESMO to discrete and continuous fidelities, respectively. Irshad et al. [45] propose a novel modification to the EHVI acquisition function which optimizes a multi-objective function and the fidelity of the data source jointly. They achieve this by defining a *trust-based cost objective* which is directly proportional to the fidelity level. However, these MOMF-BO algorithms are quite computationally expensive, requiring vast amounts of resources and longer optimization runtimes. Although they integrate cheap approximations of the objective function, their high overall computational costs make them unsuitable for DL.

Apart from a few notable exceptions, MO algorithms have been largely been used for general optimization problems. Their usage in practical DL applications have been relatively limited compared to single-objective optimization, and only a handful of studies exist where MO optimizers are benchmarked on real-world DL tasks. We aim to bridge this gap between general multi-objective optimization and multi-objective hyperparameter optimization by demonstrating `PriMO`'s effectiveness in both synthetic MO problems, as well as DL benchmarks.

B Algorithm details

Algorithm 1 Initial design strategy

function `init_design`($n_{\text{init}}, \Lambda, \eta, z_{min}, z_{max}, f, \mathbf{w}$)
  Initialize: $b \leftarrow 0, \mathcal{D} \leftarrow \emptyset$
  while $b < n_{\text{init}}$ **do**
    $\lambda, z \leftarrow$ `moasha`($\Lambda, \eta, z_{min}, z_{max}$)
    $\mathbf{y} \leftarrow f(\lambda, z)$
    if $z = z_{max}$ **then**
      $\mathbf{y} \leftarrow \mathbf{w}^\top \mathbf{y}$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda, \mathbf{y})\}$
    end if
    $b \leftarrow b + \frac{z}{z_{max}}$
  end while
  return $\mathcal{D}$
end function

Algorithm 2 Bayesian Optimization with multi-objective priors

function `moprior_bo`($\Lambda, \eta, \mathcal{D}, \Pi_n, n_{\text{BO}}, \epsilon$)
  Select prior $\pi_j(\lambda)$, where $j \sim \mathcal{U}(1, \ldots, n)$
  $\gamma \leftarrow \exp(-n_{\text{BO}}^2 / n_d)$
  $u \sim \mathcal{U}(0, 1)$
  if $u < \epsilon$ **then**
    $\tilde{\alpha}(\lambda) \leftarrow \alpha(\lambda, \mathcal{D})$
  else
    $\tilde{\alpha}(\lambda) \leftarrow \alpha(\lambda, \mathcal{D}) \cdot \pi_j(\lambda)^\gamma$
  end if
  $\lambda \leftarrow \arg\max_{\lambda \in \Lambda} \tilde{\alpha}(\lambda, \mathcal{D})$
  return λ
end function

Algorithm 3 `PriMO`

Input: Vector-valued objective function f, search space Λ with dimension n_d, MO priors $\Pi_n = \{\pi_i(\lambda)\}_{i=1}^n$, initial design size n_{init}, reduction factor η, fidelity range $[z_{min}, z_{max}]$, budget B and exploration parameter ϵ.
function `PriMO`(Input)
  Sample weights $\mathbf{w} \sim \mathcal{U}(0, 1)^n$ and normalize
  $\mathcal{D} \leftarrow$ `init_design`($n_{\text{init}}, \Lambda, \eta, z_{min}, z_{max}, f, \mathbf{w}$)
  Initialize: $b \leftarrow n_{\text{init}}, n_{\text{BO}} \leftarrow 0$
  while $b < B$ **do**
    $\boldsymbol{\lambda}_{new} \leftarrow$ `moprior_bo`($\Lambda, \eta, \mathcal{D}, \Pi_n, n_{\text{BO}}, \epsilon$)
    $n_{\text{BO}} \leftarrow n_{\text{BO}} + 1$
    $\mathbf{y} \leftarrow f(\boldsymbol{\lambda}_{new}, z_{max})$
    $\mathbf{y} \leftarrow \mathbf{w}^\top \mathbf{y}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\lambda}_{new}, \mathbf{y})\}$
    $b \leftarrow b + 1$
  end while
  return $\mathcal{P}_f(\mathcal{D})$
end function

The pseudo-codes for all the major constituents for `PriMO` are provided in Algorithm 1, Algorithm 2 and Algorithm 3. All these parts of `PriMO` were implemented in the NePS [46] package. For `PriMO`'s initial design strategy we used MOASHA which we implemented in NePS, borrowing the

code for the ϵ-net MO promotion strategy from the Syne Tune repository, which is the original implementation by its authors [22]. We set $\eta = 3$ and the initial design size to 5 which we will see in Appendix I.2 to be the best choice.

For the base acquisition function in the prior-augmented BO, we used `qLogNoisyEI` from BoTorch as it has been proven to significantly outperform ordinary EI implementations [47]. The NePS package already contains code for the `WeightedAcquisition` function for πBO, which we borrow for `PriMO`.

C Construction of priors

For the construction of priors, we closely follow the procedure described by Mallik et al. [7]. Our priors are hyperparameter settings, perturbed by a Gaussian noise with a σ depending on the prior quality. In all our experiments we use two kinds of priors for every objective - *good* and *bad* priors. The good priors represent areas of the hyperparameter space where we expect the corresponding objective to have a value close to its *optimum*. The bad priors represent *inaccurate* configurations which yield poor values for the objective function. The hyperparameter configurations for these priors are generated using the methods listed below:

- Class "**good**" priors: To generate *good* priors, we begin by uniformly sampling 100,000 hyperparameter configurations at random using a fixed global seed for all prior generation runs. We then evaluate these configurations on the corresponding benchmark at the highest available fidelity, z_{max}. Afterwards, we rank the configurations based on the objective values derived from their evaluations. Since we always aim to minimize each objective, for objectives intended to be maximized, we take their negative values to find the minimum. The configuration that yields the best objective value is perturbed by a Gaussian noise with $\sigma = 0.01$. This slight perturbation reflects a realistic scenario where prior knowledge is good or near-optimal, but never precisely so.
- Class "**bad**" priors: Similar to the good prior case, for the *bad* priors, we sort the configurations based on the corresponding objective value. From this, we select the configuration with the worst seen value and do not perturb it any further. This forms our *bad* prior configuration.

After locating the hyperparameter configurations that constitute these priors, we create a Gaussian distribution over each, $\mathcal{N}(\lambda, \sigma^2)$, where $\sigma = 0.25$ for all priors.

D Baselines

We compare `PriMO` against a host of prominent MO baselines representing different classes of optimization algorithms for MO. These include scalarized Bayesian Optimization approaches like BO with random weights (BO+RW) and ParEGO [20], multi-fidelity optimizers such as HyperBand with Random Weights (HB+RW) [13] and multi-objective asynchronous successive halving [22] (MOASHA), and an evolutionary algorithm – NSGA-II [19].

Prior-based baselines (RS + Prior, MOASHA + Prior, πBO [11], Priorband [7]) are modified to randomly chose and sample from one of the MO priors at each iteration. We further augment πBO with random scalarizations and modify Priorband's ensemble sampling policy using *scalarized incumbents for MO* to build MO-Priorband. It is important to note here that for all scalarized BO algorithms, we set the initial design size to the number of search space dimensions. The implementation and hyperparameter setting of all baselines used in this paper are individually detailed below:

Bayesian Optimization with Random Weights (BO+RW). BO with random weights is a popular MO baseline which converts the MO function into a SO optimization problem. We extend the BO implementation in the NePS [46] package by scalarizing the multivariate objective function f with randomly chosen weights for every seed, at the beginning of the optimization process. The initial design

size of the BO is set to be the same as the dimensionality of the corresponding benchmark's search space. The BO implementation in the NePS library uses the `q-Log-Noisy Expected Improvement` acquisition function from BoTorch, which has been shown to perform significantly better than ordinary Expected Improvement implementations ([47]).

ParEGO. Just like BO+RW, ParEGO is another BO baseline with the Chebyshev norm as the scalarization function. We use the ParEGO implementation from the SMAC3 package and leave the initial design design size of the BO as the package default (search space dimensions).

NSGA-II. NSGA-II is an EA algorithm which uses non-dominated sorting to identify promising configurations and crowding-distance sort as a tie-breaker. It is a popular baseline but EAs are quite sample-inefficient and hence not super practical for DL as a standalone optimization algorithm. Thus, we use NSGA-II as a representative EA baseline and borrow its implementation from the Nevergrad package. The parameters of the algorithm are set to the defaults values defined in Nevergrad.

HyperBand with Random Weights (HB+RW). HB is a common bandit-based baseline for all MF benchmarking studies. The NePS package provides an implementation of HB which allows for continuations, that we modify with random weights in the same way as BO+RW above. For all our experiments, we set the η to 3.

Multi-objective Aynchronous Successive Halving (MOASHA). MOASHA is an infinite horizon MO optimizer and currently one of the state-of-the-art baselines for multi-objective optimization, using bandit-based ASHA as the base. Like ASHA, MOASHA can also run very efficiently on HPO setups with many parallel workers, reducing idle-time. However, even for single worker setups, MOASHA is able to leverage its asynchronous promotion strategy to achieve competitive performance [22], and that is what we employ for the experiments in this paper. We implement our own version of MOASHA in the NePS package using the official code for ϵ-net from Syne Tune [48]. Just like HB+RW above, we set $\eta = 3$ in MOASHA.

πBO with random weights (πBO+RW). πBO is a SO, blackbox optimization algorithm which augments the acquisition function with user-specified priors. We use the πBO implementation from the NePS package and extend it to MO with random weight just like BO+RW above. For use with MO priors, we modify πBO to randomly chose and sample from one of the MO priors at each iteration. The original πBO paper [11] uses $\gamma = \frac{\beta}{n}$ to denote the power to which the prior PDF term is raised when multiplied by the values of the acquisition function, where n refers to the n-th iteration and the value of β is set to 10. In the NePS package, however, γ is completely different and is set to e^{-n_{BO}/n_d}, where n_{BO} refers to the number of BO samples and n_d indicates the dimensions of the search space. Like BO+RW, we set the initial design size to n_d, and sample from a randomly chosen prior for each of the initial points.

MO-Priorband. Priorband integrates cheap proxies unlike πBO to achieve good anytime performance. It employs an ESP strategy for sampling proportionately from the *priors*, the *incumbent* and at *random*. We extend Priorband to the MO domain by first replacing the MF component with an MOMF component. Then, to calculate the *top_k* configurations, we scalarize the MO vectors using weights, randomly chosen during each iteration. Additionally, to integrate MO priors, MO-Priorband chooses one of the available priors at random at each iteration. We note that a MOASHA base and scalarization-based incumbent modification works better for MO-Priorband, than a HB base (used in the original Priorband algorithm) and Pareto front incumbent such as ϵ-net, respectively. Additionally, we set MO-Priorband's $\eta = 3$, just like in MOASHA.

E Benchmarks

In this paper we use 9 benchmarks from 3 different families representing image classification, language translation and learning curves for Deep Neural Networks. We chose the synthetic MOMF-Park benchmark from BoTorch (Balandat et al. [49]), 4 LCBench benchmarks from the Yahpo-Gym Suite [50] and 4 from the PD1 (HyperBO [51]) set of benchmarks. We use the MOMF-Park benchmark's 2 objectives and select the corresponding *validation error* and *cost* objectives from the LCBench and PD1 benchmarks. Therefore, in this section, we study the effect of the prior combination *good-good*, the average over bad prior combinations *overall-bad* (*bad-bad* and *bad-good*), as well as the average over all three of these combinations *overall-all*. Our main experiments in Section 4 include the synthetic benchmark - MOMF-Park, and surrogate benchmarks LCBench-126026, LCBench-146212, LCBench-168330, LCBench-168868 from Yahpo-Gym and cifar-100, imagenet, translate-wmt-xformer, lm1b-transformer from the PD1 suite.

E.1 Multi-objective Multi-fidelity Park Problem

The single-objective, 4-dimensional Park 1 and Park 2 functions from Park [52] are first individually modified by Irshad et al. [45] to incorporate a fidelity parameter s, allowing for multi-fidelity evaluations. These are then merged to create an MO problem. The MOMF-Park test problem is available in BoTorch, which we wrapped using `hpoglue` to create a benchmark. The individually modified functions are:

$$P_1(\boldsymbol{x}^{'}, s) = A(s)\left[\frac{T_1 + T_2 - B(s)}{22}\right] - 0.8 \quad , \tag{9}$$

the modified Park 1 function, where,

$$T_1 = 0.5\left[x_1 + 0.001(1-s)\right].\left[\sqrt{1 + (x_2 + x_3^2)\frac{x_4}{x_1^2}}\right] \quad ,$$

$$T_2 = (x_1 + 3x_4)\exp[1 + \sin(x_3)] \quad ,$$

and,

$$P_2(\boldsymbol{x}^{'}, s) = A(s)\left[\frac{5 - \frac{2}{3}\exp(x_1 + x_2) - (x_4)\sin(x_3)A(s) + x_3 - B(s)}{4}\right] - 0.7 \quad , \tag{10}$$

the modified Park 2 function. For both these functions, $A(s) = (0.9 + 0.1s)$ and $B(s) = 0.1(1-s)$.

In our experiments we indicate the Park 1 function as the objective `value1` and Park 2 as `value2`. The reference point used for calculating the dominated Hypervolume in our experiments is listed in Table 2. Although Irshad et al. [45] assume a continuous fidelity space $s \in (0, 1)$, we use a discrete space $z \in [1, 100]$ to ensure numerical stability in bandit-based optimizers like HyperBand with Random Weights and MOASHA. Table 3 gives a detailed overview of the Hyperparameter space of the MOMF-Park Benchmark.

Table 2: Reference values for `value1` and `value2` objectives in the MOMF-Park benchmark, for calculating the Hypervolume Improvement. Please note that the original MOMF-Park benchmarks are designed to maximize their objectives. We negate the values in all our optimization runs.

Benchmark Name	`value1` (max)	`value2` (max)
`MOMFPark`	1.0	1.0

Table 3: Search space of the synthetic multi-objective multi-fidelity Park benchmark, including the discretized fidelity space z.

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
x_0	float		$[0.0, 1.0]$	continuous	
x_1	float		$[0.0, 1.0]$	continuous	
x_2	float		$[0.0, 1.0]$	continuous	
x_3	float		$[0.0, 1.0]$	continuous	
z	integer		$[1, 100]$	discrete	fidelity

E.2 PD1 (HyperBO)

PD1 from HyperBO (Wang et al. [51]) is a collection of XGBoost surrogates trained on the learning curves of near state-of-the-art DL models on a diverse array of practical downstream DL tasks including image classification, language modeling and language translation. Overall, PD1 contains 24 benchmarking tasks, with each consisting of a task dataset, a DL model, and a broad search space for Nesterov Momentum (Nesterov [53]).

From these 24, we select 4 benchmarks from `mf-prior-bench` (Bergman et al. [54]) providing a well-rounded representation of DL models and the aforementioned tasks. For each of these benchmarks, we select the `valid_error_rate` as the error objective and `train_cost` as the cost objective. All of these benchmarks have a single fidelity `epoch`. We list the static reference points for calculating the HVI for the PD1 benchmarks in Table 4. The individual benchmarks are further detailed below:

1. **cifar100-wide_resnet-2048** benchmark contains the optimization trace of a `WideResnet` (Zagoruyko and Komodakis [55]) model on the `CIFAR-100` (Krizhevsky [56]) dataset with a batch size of 2048. The hyperparameter space of this benchmark is given in Table 5.

2. **imagenet-resnet-512** surrogate is trained on the learning curve of a `ResNet50` (He et al. [57]) on the `ImageNet` (Russakovsky et al. [58]) dataset with a batch size of 512. See Table 6 for the detailed search space of this benchmark.

3. **lm1b-transformer-2048** is a surrogate trained on the HPO runs of a transformer model (Roy et al. [59]) on the *One Billion Word* statistical language modeling benchmark (Chelba et al. [60]). Table 7 lists the search space of the benchmark.

4. **translatewmt-xformer-64** surrogate is trained on the HPO runs of an `xformer` (Lefaudeux et al. [61]) transformer model on the `WMT15 German-English` text translation dataset (Bojar et al. [62]). For the detailed search space, see Table 8.

Table 4: Reference values for `valid_error_rate` and `train_cost` objectives across PD1 benchmarks for HVI calculation.

Benchmark Name	`valid_error_rate` (max)	`train_cost` (max)
`cifar100-wide_resnet-2048`	1.0	30
`imagenet-resnet-512`	1.0	5000
`lm1b-transformer-2048`	1.0	1000
`translatewmt-xformer-64`	1.0	20000

E.3 LCBench surrogate benchmarks (YAHPO-Gym)

Yahpo-Gym (Pfisterer et al. [50]) is a large collection of multi-objective multi-fidelity surrogate benchmarks trained on a wide array of tasks with fidelities including epochs as well as dataset

Table 5: Hyperparameter search space table of the `cifar-100-wide_resnet-2048` benchmark, including the hyperparameter ranges and fidelity bounds of epoch, as given in `mf-prior-bench` .

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`lr_decay_factor`	float		$[0.010093, 0.989012]$	continuous	
`lr_initial`	float	✓	$[0.000010, 9.779176]$	continuous	
`lr_power`	float		$[0.100708, 1.999376]$	continuous	
`opt_momentum`	float	✓	$[0.000059, 0.998993]$	continuous	
epoch	integer		$[1, 52]$	discrete	fidelity

Table 6: Approximate hyperparameter search space table of the `imagenet-resnet-512` benchmark, including hyperparameter ranges and fidelity bounds of epoch. Exact ranges are provided by `mf-prior-bench` .

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`lr_decay_factor`	float		$[0.010294, 0.989753]$	continuous	
`lr_initial`	float	✓	$[1e-5, 9.774312]$	continuous	
`lr_power`	float		$[0.100225, 1.999326]$	continuous	
`opt_momentum`	float	✓	$[5.9e-5, 0.998993]$	continuous	
epoch	integer		$[1, 99]$	discrete	fidelity

Table 7: Hyperparameter search space of the `lm1b-transformer-2048` benchmark, with the fidelity epoch as given in `mf-prior-bench` .

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`lr_decay_factor`	float		$[0.010543, 0.9885653]$	continuous	
`lr_initial`	float	✓	$[1e-5, 9.986256]$	continuous	
`lr_power`	float		$[0.100811, 1.999659]$	continuous	
`opt_momentum`	float	✓	$[5.9e-5, 0.9989986]$	continuous	
epoch	integer		$[1, 74]$	discrete	fidelity

Table 8: Search space and fidelity epoch of the `translatewmt-xformer-64` benchmark, as given in `mf-prior-bench` .

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`lr_decay_factor`	float		$[0.0100221257, 0.988565263]$	continuous	
`lr_initial`	float	✓	$[1.00276e-5, 9.8422475735]$	continuous	
`lr_power`	float		$[0.1004250993, 1.9985927056]$	continuous	
`opt_momentum`	float	✓	$[5.86114e-5, 0.9989999746]$	continuous	
epoch	integer		$[1, 19]$	discrete	fidelity

fractions. Yahpo-Gym also contains surrogates for the LCBench (Zimmer et al. [63]) set of benchmarks that consists of surrogates trained on the learning curves of DL models, on several OpenML (Vanschoren et al. [64]) datasets. Out of these, we choose 4 task OpenML IDs for the experiments in this paper – 126026, 146212, 168330 and 168868. The fidelity for these tasks is epoch and we select the `val_cross_entropy` and `time` objectives for our experiments. Table 9 lists the maximum bounds used as the reference points for calculating the Hypervolume Improvement, for each of the selected LCBench task IDs. All LCBench benchmarks share a common search space, detailed in Table 10.

Table 9: Reference values for `val_cross_entropy` and `time` objectives across selected LCBench tasks, for HVI calculation.

Task ID	`val_cross_entropy` (max)	`time` (max, seconds)
126026	1.0	150
146212	1.0	150
168330	1.0	5000
168868	1.0	200

Table 10: Hyperparameter search space table of the `yahpo-lcbench` benchmarks. This includes the hyperparameter ranges and types as typically defined in the YAHPO-Gym benchmark suite.

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`batch_size`	integer	✓	$[16, 512]$	discrete	
`learning_rate`	float	✓	$[1e{-}4, 0.1]$	continuous	
`momentum`	float		$[0.1, 0.99]$	continuous	
`weight_decay`	float		$[1e{-}5, 0.1]$	continuous	
`num_layers`	integer		$[1, 5]$	discrete	
`max_units`	integer	✓	$[64, 1024]$	discrete	
`max_dropout`	float		$[0.0, 1.0]$	continuous	
`epoch`	integer		$[1, 52]$	discrete	fidelity

F Evaluation protocol and analysis

We report the mean dominated HV along with standard error bars against function evaluations across 25 seeds, for each HPO algorithm on all benchmarks. The HV is computed with respect to a static reference point set for each benchmark (Appendix E). Additionally, for every baseline, we report the Pareto front aggregated across all seeds per benchmark in line with existing literature [12, 13, 22], with the primary (error) objective on the x-axis and the cost objective along the y-axis. Each optimizer-benchmark-seed combination was run for 20 *equivalent* function evaluations. For blackbox optimizers like BO+RW, NSGA-II and ParEGO, every optimization iteration is equal to a function evaluation since they evaluate f at the maximum fidelity z_{max}. For optimizers such as MOASHA, HB+RW and `PriMO` that use cheap proxies of the objective, we calculate equivalent function evaluations as z/z_{max} where z is the fidelity at which f is evaluated at a given iteration. We note here that for all MF optimizers, we leveraged continuations and plot the HV only when an equivalent full function evaluation has been performed, *i.e.*, when the benchmark is evaluated at its highest fidelity z_{max}. Further details about the experiment repository, resources and licenses of packages used in this paper can be found in Appendices K, L and M respectively.

G Additional experiments and discussion

In this section, we present and discuss the performance of `PriMO` against prominent baselines, to highlight its novelty and robustness across a variety of benchmarks and multiple seeds, under various prior conditions. First, we present a naive approach for incorporating multi-objective priors. We then evaluate the performance of `PriMO` when integrating good-good priors, and assess its robustness under misleading prior conditions. In each of these two sections, we compare `PriMO` against the strongest non-prior-based MO baselines, including RS, HB+RW, MOASHA, NSGA-II, ParEGO, and BO+RW. We further engage in a detailed discussion of the study results in both sections, highlighting the key attributes of `PriMO` that contribute to its strong performance.

G.1 Naive solution for incorporating multi-objective priors does not work

For single-objective optimization, Mallik et al. [7] provide empirical evidence that the naive approach of simply augmenting existing algorithms with priors is not the best solution. In Figure 2 we study

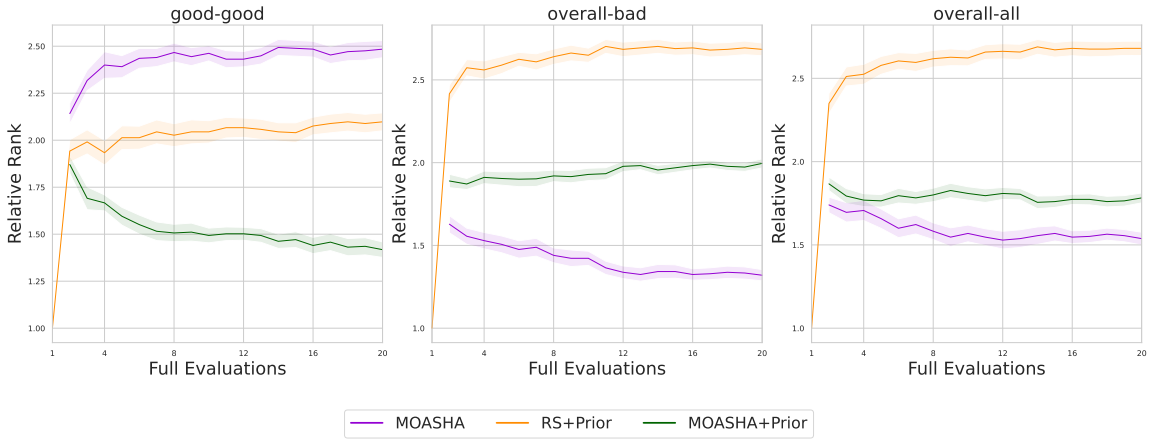

Figure 2: Comparing MOASHA, a good MO baseline against the naive methods for incorporating expert priors - MOASHA + Prior and RS + Prior. In each iteration, both the prior-augmented optimizers sample from one of the 2 (randomly selected) MO priors.

the effect of solving Equation (1) using a naive approach for MO. Instead of RS in MOASHA, configurations are sampled from one of the MO priors chosen randomly at every iteration. As we would expect, this is not the most robust solution. In the presence of good prior knowledge, MOASHA + Prior is able to identify good configurations and considerably outperforms the non-prior MOASHA, but leads to drastically poor performance with misleading priors. Therefore, we propose `PriMO`, to benefit from *good* MO prior beliefs while simultaneously having the ability to recover from *bad* priors.

G.2 State-of-the-art performance of PriMO under good priors

Initial observations. . Figure 3 demonstrates that `PriMO` starts off really strong and by about the 4th (equivalent) function evaluation, is already the best performing algorithm across all benchmarks, based on the mean dominated HV. This is in part due to the novel initial design strategy which provides a strong head-start even before the BO phase has even begun. For the first few evaluations, we notice that the performance coincides with that of MOASHA, which is by design expected. From Figure 3 it is clear that the points sampled by `PriMO`'s initial design are already better compared to those sampled randomly for BO+RW. Using these configurations, the BO phase of `PriMO` maintains its **strong anytime performance**. Here, `PriMO` is able to effectively utilize the priors, which augment the acquisition function, and achieve **state-of-the-art final performance** across all benchmarks.

Closely examining Figure 4 leads to the conclusion that `PriMO` typically samples non-dominated configurations, some of which in a few of the benchmarks are only weakly dominated by BO+RW. Nevertheless, we clearly notice that `PriMO` provides a better coverage of the overall Pareto front.

Further discussion. `PriMO` benefits from good-good priors that, after the end of its initial design, provides an additional boost in its performance. We observe this effect throughout the optimization run across all benchmarks, where the dominated HV consistently improves. By the final evaluation, `PriMO` outperforms all baselines. Designed with the practical DL use case in mind, where most practitioners operate on modest budgets, `PriMO` reduces its dependence on priors after approximately 10 BO samples, governed by our chosen γ setting (see Section 3).

G.3 Robustness of PriMO under bad prior conditions

Initial observations. We present the results of `PriMO` under the average of the bad prior combinations - bad-good and bad-bad, denoted as overall-bad.

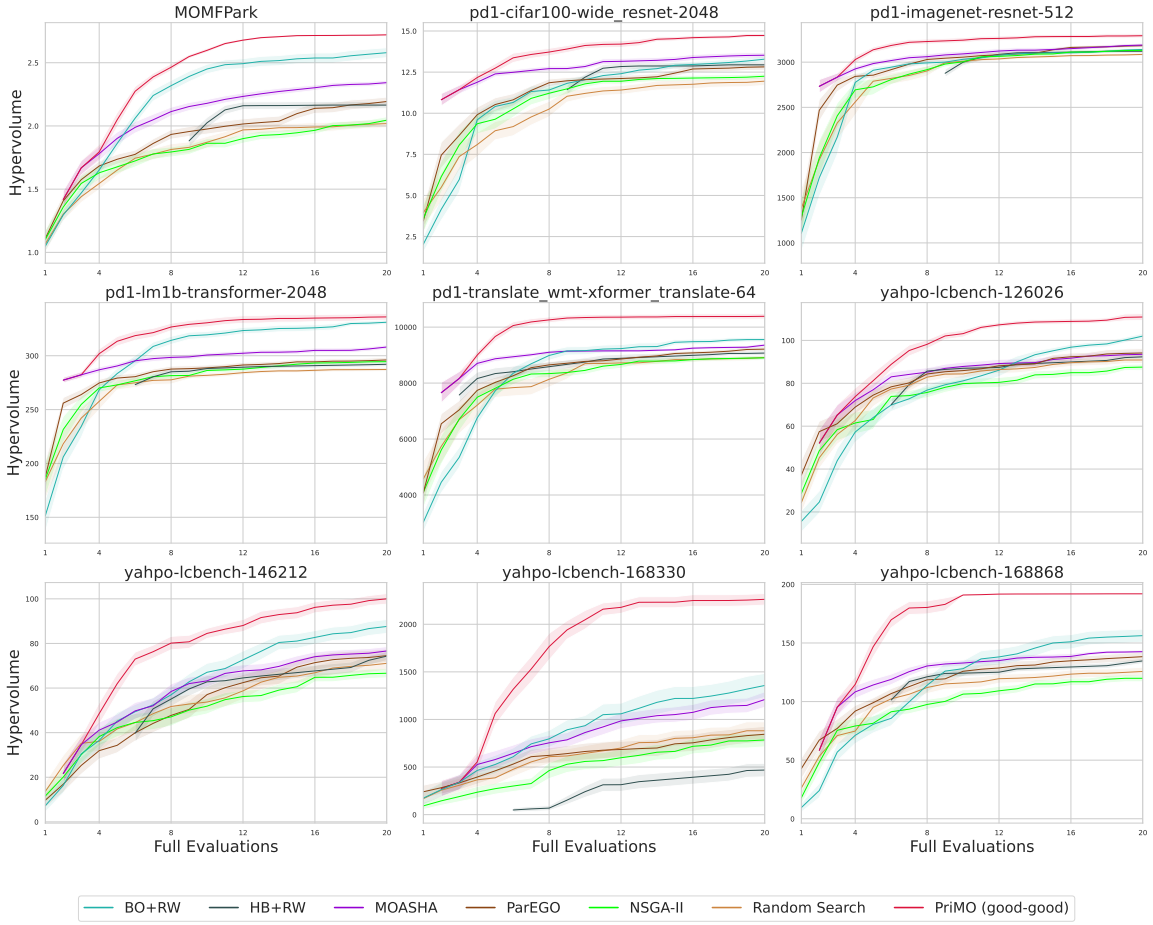

Figure 3: Shown here is the comparison in mean dominated Hypervolume with respect to function evaluations across 25 seeds and 9 benchmarks, between PriMO and some state-of-the-art non-prior MO baselines - Bayesian Optimization with Random Weights (BO+RW), MOASHA, HyperBand with Random Weights (HB+RW), ParEGO, NSGA-II and Random Search. PriMO is under the good-good prior setting here.

In Figure 5 we see that, with the notable exception of the LCBench-168330 benchmark, PriMO starts off really strong again. Initially outperformed on a few benchmarks by BO+RW, as the optimization process continues, we clearly see PriMO's strong recovery from these inaccurate priors, resulting in a competitive final performance.

The Pareto front plot in Figure 6 shows that, on average, PriMO is the best algorithm along with BO+RW as they locate more non-dominated points compared to the rest of the baselines.

Further discussion. We attribute PriMO's strong recovery under misleading priors to our design of the *decaying MO-prior-weighted acquisition*, influenced by two key parameters – β and ϵ.

The aggressive β setting ensures the prior's influence diminishes rapidly — an important property for practical DL scenarios where HPO is not expected to be run for long. Additionally, the parameter ϵ in the acquisition function controls how much the prior contributes while it is still active, thus encouraging exploration of the search space. Together, these two effects ensure that PriMO does not become overly dependent on the prior and, under inaccurate priors, can still effectively explore and discover better hyperparameter configurations than its counterparts.

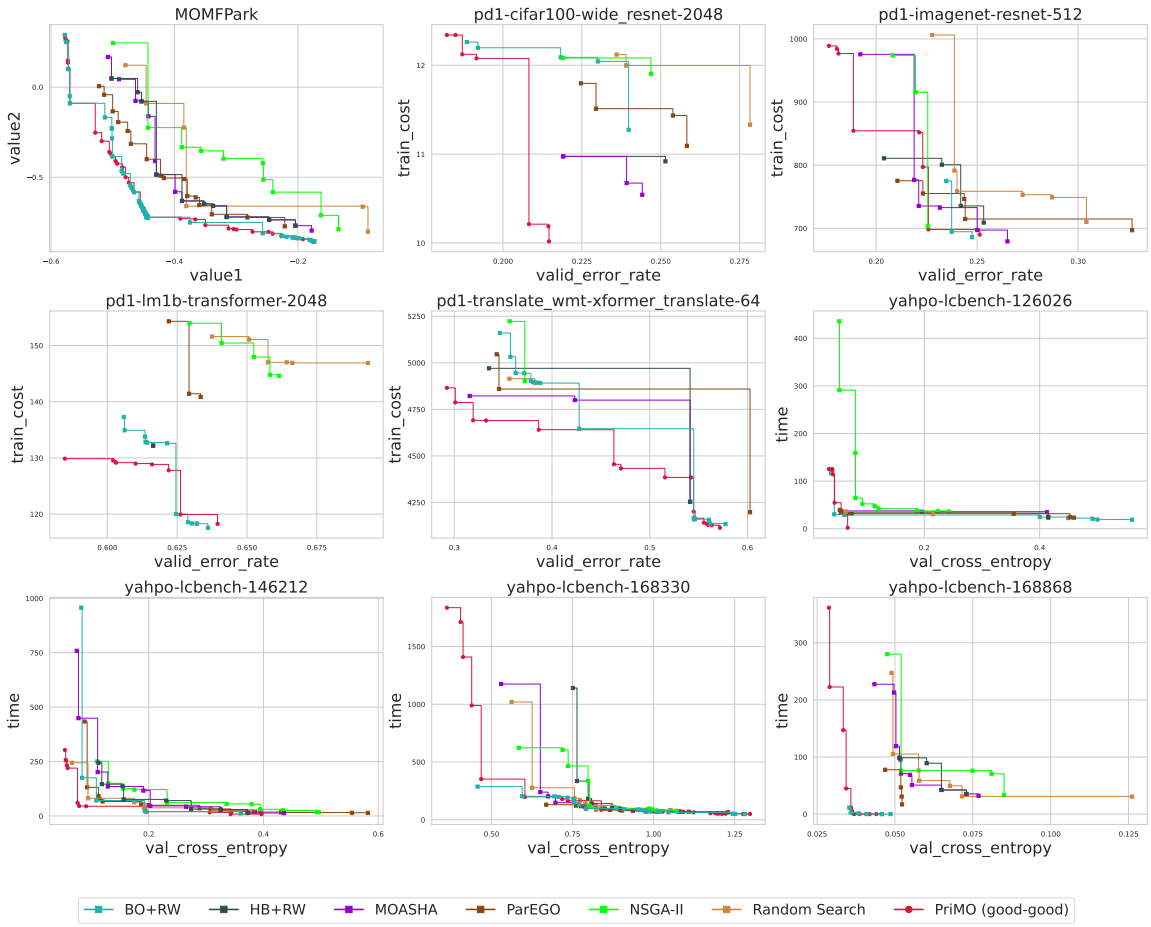

Figure 4: Shown here are the Pareto fronts obtained by PriMO, compared to other non-prior MO baselines under good-good prior conditions.

G.4 A note on compute efficiency

PriMO stands out as being *extremely compute-efficient*, on average, achieving significant performance gains with minimal HPO evaluations, *i.e.*, with a low compute budget. We see this under good-good prior conditions in the HV plots across most benchmarks in Figure 3. Given that we set PriMO's initial design size to 5, an asynchronous MF optimizer like MOASHA (in a continual setup) effectively requires only about 3.5 equivalent function evaluations, which on average results in 3 configurations sampled at z_{max}. Therefore, compared to other BO algorithms whose initial design size we set to the number of dimensions, PriMO effectively uses fewer max-fidelity configurations to fit the GP in the BO phase. Despite fewer samples, PriMO already achieves much better performance in the beginning compared to all baselines, due to the use of its initial design strategy.

In summary, these findings support our claim that PriMO is a robust and general purpose multi-objective hyperparameter optimization algorithm designed for real-world DL workloads, fulfilling all the desiderata outlined in Table 1.

H Observed speedups

We ran PriMO against the baselines over a longer budget of 100 evaluations. We notice significant speedups offered by PriMO across most benchmarks, with the highest being approximately a **15x** speedup on the translate-wmt-xformer benchmark (Figure 7).

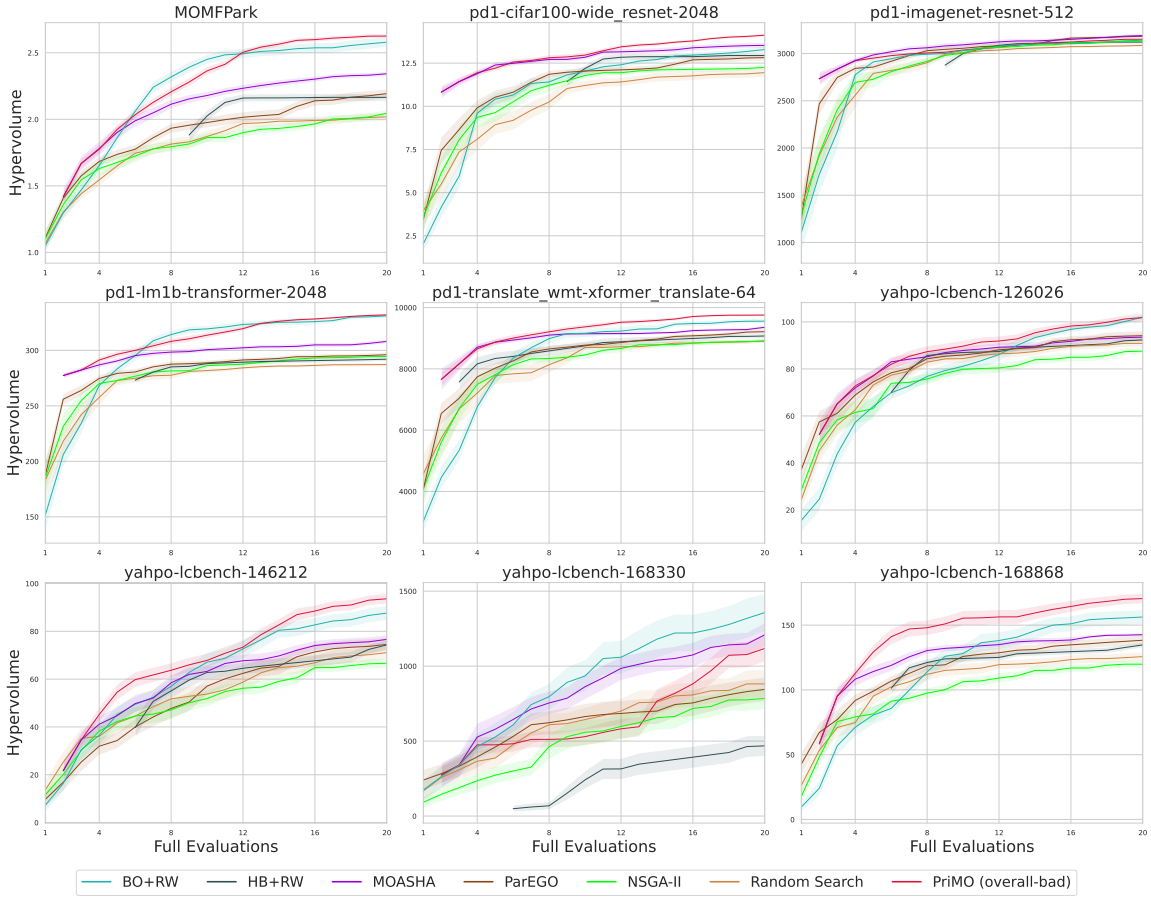

Figure 5: Mean dominated HV across 9 benchmarks, showcasing consistent performance and good recovery (from bad priors) of `PriMO` against some common non-prior baselines. `PriMO` is under the overall-bad (bad-bad and bad-good) prior combination here.

I Ablations for PriMO

I.1 General ablations

Here, we dive into the various design ablations of `PriMO` to answer **RQ5**.

Experiment outline. Here, we consider different design ablations of `PriMO` under the overall-all priors average. Specifically, we study how the various parts of `PriMO` work individually and in combination with each other. For this, we also include the previously used baselines BO+RW and πBO+RW which, in a way, can also be considered ablated versions of `PriMO`. We divide `PriMO` into its constituent components, namely – MOMF initial design, Priors and Model (including the random weights), and label it as such. We denote BO+RW as just *Model* since it only comprises the Model + random weights component of `PriMO`, while πBO+RW is indicated as Priors + Model. The vanilla variant of `PriMO` without Priors is just an MOMF initial design with the final BO phase. We denote this as MOMF initial design + Model. Finally, the model component is replaced with Random Search which is designated MOMF initial design + RS.

Observations. Figure 8 presents the rankings of different algorithm designs under good-good, overall-bad, and overall-all priors. A standalone MOMF initial design with random sampling performs significantly worse after its initial head start from the MOMF component across most benchmarks, as expected. Interestingly, we also observe that Priors + Model – *i.e.* πBO – performs

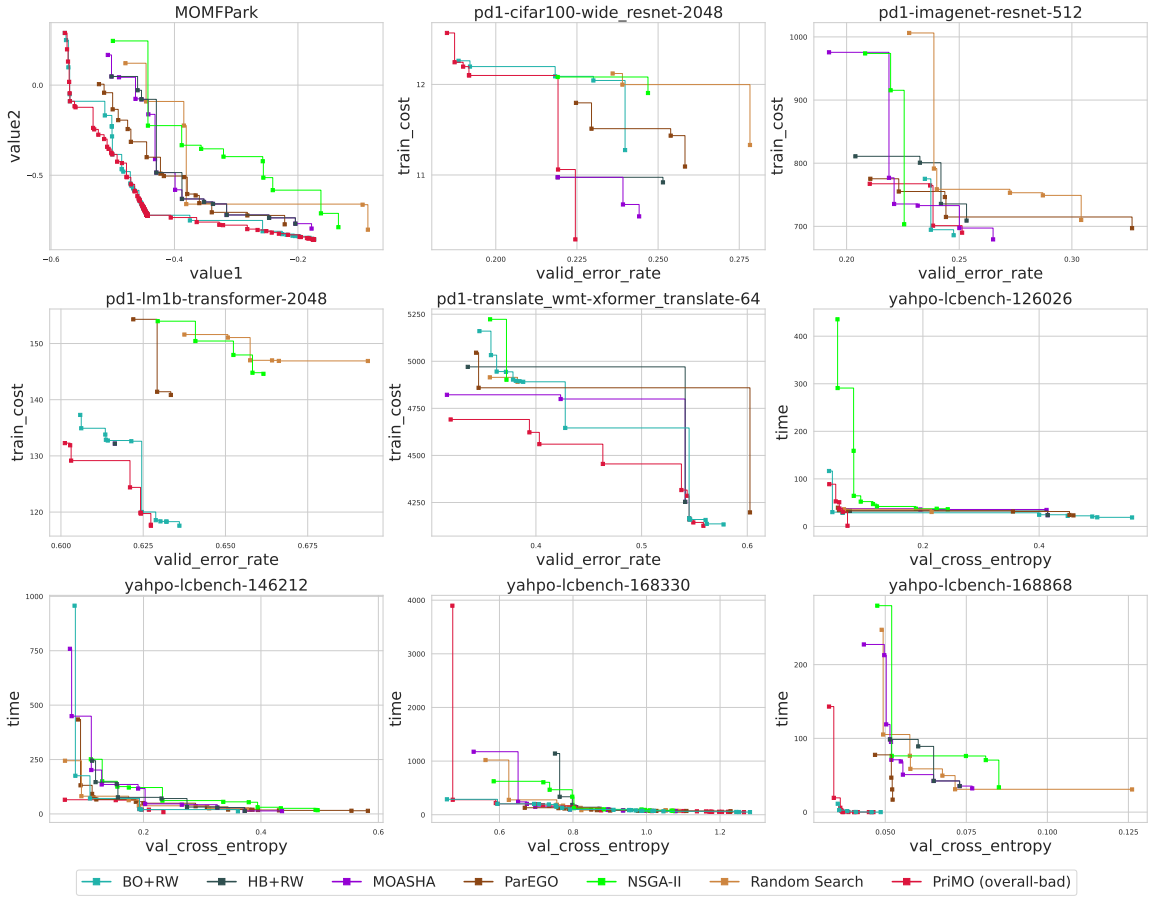

Figure 6: We compare the non-dominated solutions obtained by `PriMO` and some popular MO baselines. `PriMO` is under overall-bad priors and yet showcases its recovery strength by finding the best Pareto fronts across most benchmarks.

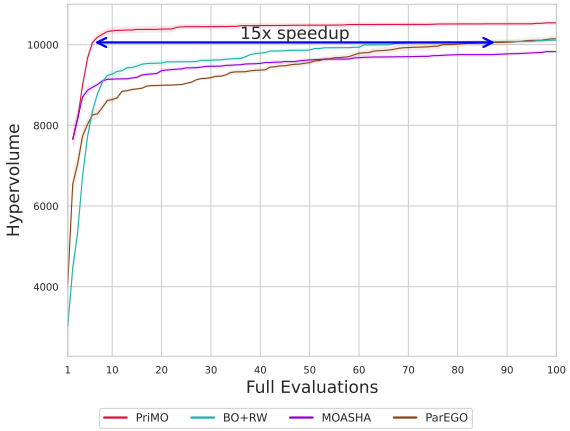

Figure 7: Shown here is a typical plot of the dominated Hypervolume against function evaluations on a DL benchmark, demonstrating that `PriMO` offers speedups of up to ~15x, compared to some of the strongest MO baselines that we consider in our study.

notably worse than MOMF initial design + RS under both overall-bad and overall-all priors, and

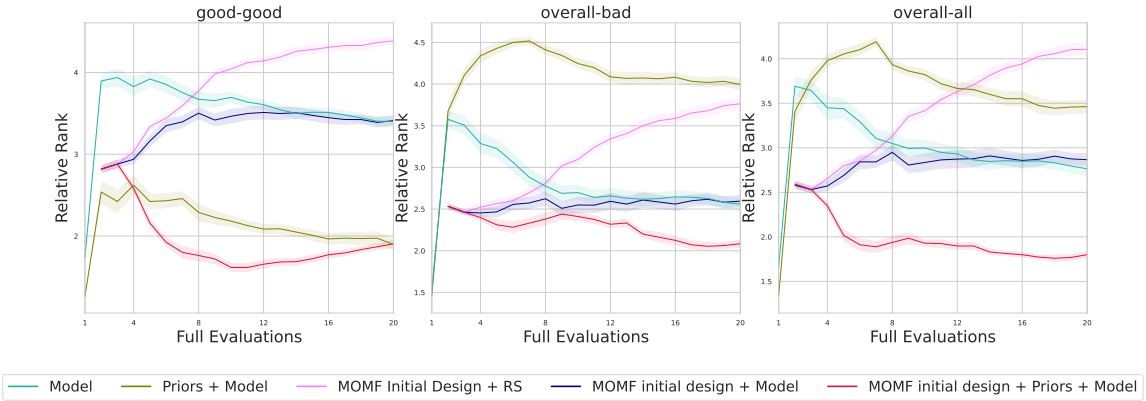

Figure 8: Overall relative ranking plots design ablations of PriMO under all prior combinations. **Left**: ranking under good-good prior conditions. **Middle**: relative ranks under overall-bad priors averaging over bad-bad and bad-good prior conditions. **Right**: relative ranks under all prior conditions averaged.

only begins to improve over MOMF initial design + RS in the final evaluations under overall-all priors. We find that the MOMF initial design gives a substantial early boost to BO, as MOMF + BO+RW starts much stronger than vanilla BO+RW. This initial advantage tends to persist for a significant number of evaluations. Overall, while the MOMF initial design provides meaningful early speedups to BO, it does not sustain strong performance in the long run unless paired with the full MO priors-augmented BO. These findings, taken together, support our final design choice for PriMO (**RQ5**).

I.2 Ablation study for initial design size

Experiment overview. In this appendix, we evaluate how varying the initial design (init) size of PriMO affects optimization performance. Specifically, we compare PriMO with init sizes of 5, 7, and 10, using BO+RW as a baseline across all benchmarks. We include multiple prior combinations—good-good, overall-bad, and overall-all - to assess how the init size influences PriMO's performance under good priors and its ability to recover from poor ones. As expected, a smaller initial design size leads to a shorter MOASHA phase and a longer BO phase, and vice versa. However, the duration of prior influence is unaffected by the init size, as it is governed solely by the γ hyperparameter which we fix as a design choice. We also tested an init size of 3, but this setting failed to return any configurations evaluated at z_{max} in several runs - likely because the shorter MOMF phase prevented its ASHA base from promoting candidates to the highest fidelity rung.

Results and discussion. Figure 9 (left) compares the different init sizes under good-good priors. An init size of 5 performs best early on, as it activates the prior-augmented BO phase sooner than the others. This head start is exploited by the longer BO phase that follows, further improving relative ranking. Under overall-bad priors (Figure 9) (middle), the differences across init sizes are less pronounced initially, but init = 5 again emerges as the most robust - showing the strongest recovery and best final performance. This can be attributed to the longest BO phase, where the influence of the misleading prior decays sooner, allowing more room for the optimizer to recover. The relative ranking under the overall-all setting (Figure 9) (right) reflects a similar trend, with init = 5 clearly achieving the best final relative rank across all benchmarks.

J Significance analysis

In this appendix, we perform statistical significance tests using Linear Mixed Effect Models (LMEMs) to verify the results obtained in our experiments. Our choice for using LMEMs is supported by

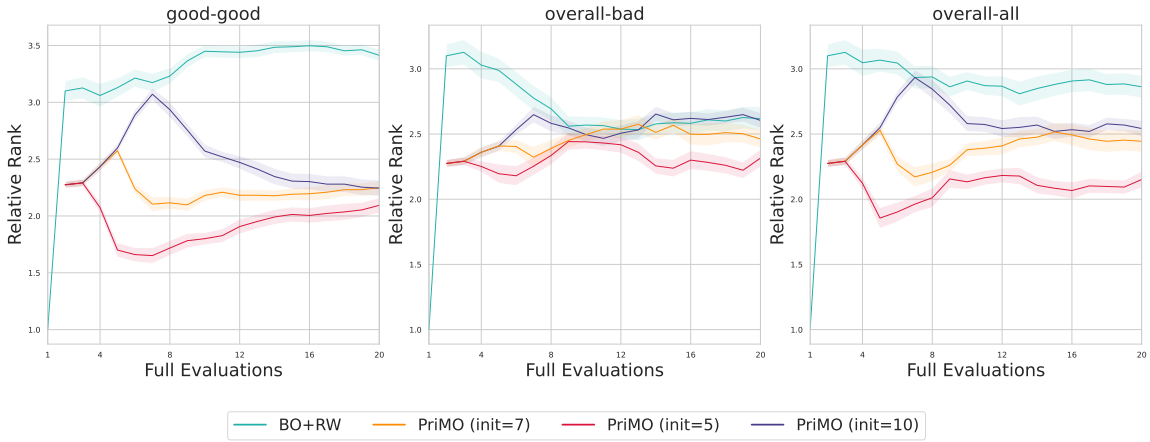

Figure 9: Overall relative ranking plots comparing `PriMO` with 3 initial design size choices and a BO+RW baseline. **Left**: ranking under good-good prior conditions. **Middle**: relative ranks under overall-bad priors averaging over bad-bad and bad-good prior conditions. **Right**: relative ranks under all prior conditions averaged.

Riezler and Hagmann [65] who proposed LMEM-based significance testing for Natural Language Processing tasks. Further, Geburek et al. [66] argued for the usage of LMEM-based significance analysis for HPO benchmarking.

J.1 Data preparation and sanity checks

To prepare the data for the significance analysis, we computed and used normalized Hypervolume regret scores, as the scale of HV can vary considerably across benchmarks. After aggregating the normalized HV regret values at each function evaluation, we conducted sanity checks to ensure statistical validity. We then performed a post-hoc analysis and used Critical Difference (CD) diagrams to compare the **early** and **final** performance of `PriMO` against all other algorithms.

Seed independency check. We fitted two LMEMs:

$$\texttt{normalized_hv_regret} \sim \texttt{algorithm} \quad , \tag{11}$$

and,

$$\texttt{normalized_hv_regret} \sim \texttt{(0 + algorithm | seed)} \quad , \tag{12}$$

on the data and performed a Generalized Likelihood Ratio Test (GLRT) to verify that the seed is not a significant effect.

Benchmark informativeness. Using GLRT, we compared the likelihoods of the LMEMs:

$$\texttt{normalized_hv_regret} \sim \texttt{1} \quad , \tag{13}$$

and,

$$\texttt{normalized_hv_regret} \sim \texttt{algorithm} \quad , \tag{14}$$

which confirmed that our benchmarks are informative, as the second model (Equation (14)) was shown to be significantly better. This further indicates that there are indeed significant differences between the performance of algorithms across all benchmarks, justifying the use of CD diagrams for comparison.

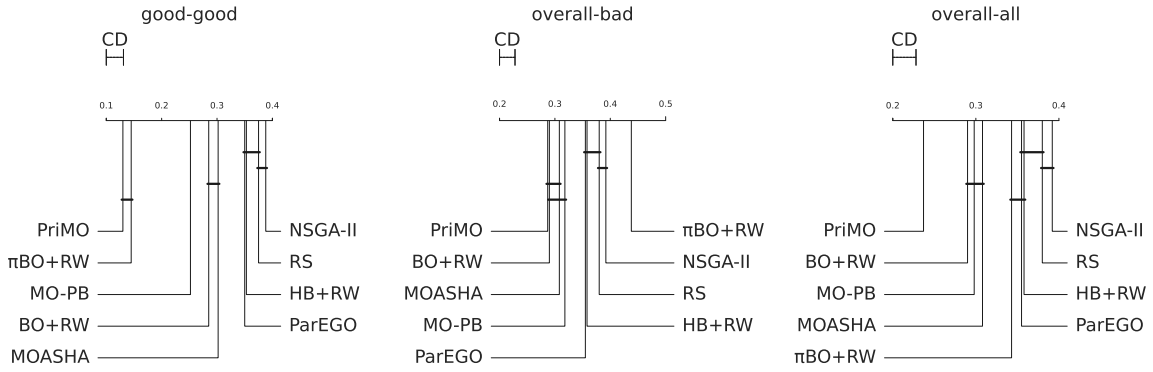

Figure 10: Critical Difference diagrams at **10 evaluations** comparing **early performance** of `PriMO` against the baselines – BO+RW, πBO+RW, MO-Priorband (MO-PB), MOASHA, RS(RS), ParEGOand NSGA-II, under various prior conditions. **Left**: CD diagram for good-good prior conditions. **Middle**: CD diagram for overall-bad priors. **Right**: CD diagram for overall-all – all prior combinations averaged.

J.2 Critical difference diagrams

We perform pairwise Tukey HSD (Tukey [67]) tests using LMEMs to obtain individual p-values for each comparison. Using this, we plot the CD diagrams.

Here, we consider the statistical differences in the early and final performance between `PriMO` and other algorithms. Figure 10 shows CD plots for 10 function evaluations, *i.e.*, halfway through our entire allocated budget. Figure 10 (left) shows that `PriMO` is able to efficiently leverage good priors very early during the optimization, and significantly better than all baselines except πBO+RW. Under overall-bad priors, Figure 10 (middle) demonstrates `PriMO`'s recovery strength and it is already the best ranked algorithm after 10 evaluations, being significantly better than more than half of the other algorithms. However, averaging all prior conditions in Figure 10 (right), we observe a **significant difference** between `PriMO` and all other optimizers. `PriMO` is shown to be the best ranked algorithm with **significantly strong early performance**.

In Figure 11, we show the CD diagrams for 20 function evaluations, *i.e.*, at the end of our optimization budget. As observed in our relative ranking plots before, there is negligible critical difference between `PriMO` and πBO+RW under good-good priors, while both are significantly better than other algorithms (Figure 11 left). Figure 11 (middle) verifies the final performance of `PriMO` under overall-bad priors, highlighting a strong recovery, where, with the notable exception of BO+RW, `PriMO` is shown to be significantly better than all baselines. Finally, under overall-all prior conditions in Figure 11 (right), `PriMO` is clearly shown to be the algorithm with the highest rank, indicating the **strongest final performance**. Thus, Figures 10 and 11 confirm our relative ranking plots and statistically verify `PriMO`'s **state-of-the-art** performance, proving that overall, `PriMO` is **significantly better** than all other algorithms used in our study.

K Code repository

The `Python` code for generating the priors and running the experiments presented in this paper is publicly available in the GitHub repository: `mo_mf_priors`. This repository also contains the code used to generate all plots, along with a comprehensive `README.md` file that provides reproducibility guidelines, explains the output data structure, and outlines the steps required to run all baselines on the benchmarks used in this work. The priors over the objectives for the various benchmarks are also included. Additionally, the raw results from the all optimization runs of `PriMO` used in this paper are hosted at this URL.

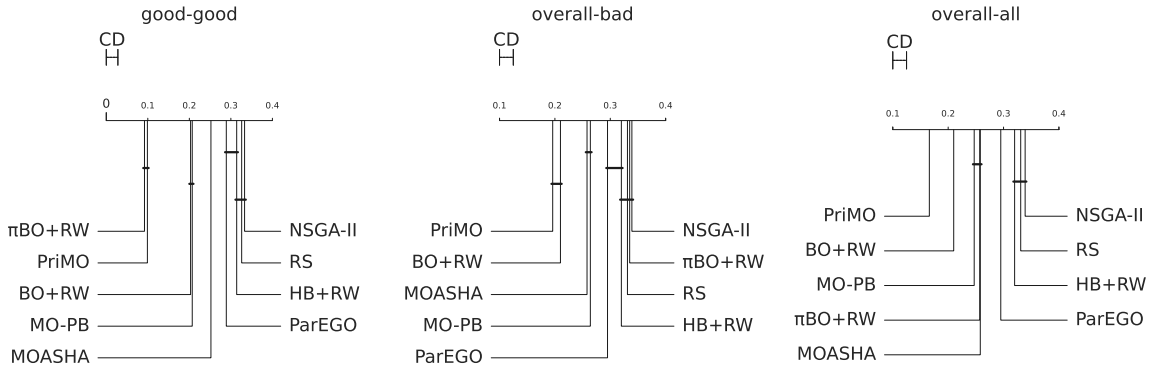

Figure 11: CD diagrams at **20 evaluations** comparing **final performance** of `PriMO` against other non-prior and prior-guided (adapted to MO) baselines, under various prior conditions. **Left**: CD diagram for good-good prior conditions. **Middle**: CD diagram for overall-bad priors. **Right**: CD diagram for overall-all – all prior combinations averaged.

L Resources used

We ran all the algorithms in this paper on inexpensive surrogate and synthetic benchmarks. To perform all our experiments, we only used a CPU compute cluster and 30 cores of Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz. For runs up to 20 function evaluations, each seed of an HPO algorithm on a single benchmark took approximately 0.02 CPU hours, or 0.6 core hours on average. While MF optimizers such as MOASHA and HB+RW completed in just a few seconds (∼0.15 core hours), model-based baselines such as BO+RW and πBO+RW required significantly longer on average – typically over 5 minutes (∼2.5 core hours).

For the main experiments in Section 4, we ran 9 optimizers in total – 6 non-prior and 3 prior-based, including `PriMO`. Each prior-based optimizer was evaluated under 3 different prior combinations. Each run lasted 20 evaluations and we evaluated each optimizer on 9 benchmarks across 25 seeds. In total, this amounted to ∼67.5 CPU hours, or ∼2025 core hours to generate the results presented in Section 4.

M Licenses

- Experiments repository `mo_mf_priors`: **BSD 3-Clause License**
- NePS package (including our algorithm implementation): **Apache License, Version 2.0**
- `hpoglue`: **BSD 3-Clause License**
- `mf-prior-bench`: **Apache License, Version 2.0**
- Yahpo-Gym : **Apache License, Version 2.0**
- HyperBO PD1: **Apache License, Version 2.0**
- BoTorch: **MIT License**
- Nevergrad: **MIT License**
- SMAC: **BSD 3-Clause License**
- Syne Tune (code for ϵ-net): **Apache License, Version 2.0**
- `lmem-significance`: **MIT License**