

# MULTI-OBJECTIVE HYPERPARAMETER OPTIMIZATION IN THE AGE OF DEEP LEARNING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

While Deep Learning (DL) experts often have prior knowledge about which hyperparameter settings yield strong performance, only few Hyperparameter Optimization (HPO) algorithms can leverage such prior knowledge and none incorporate priors over multiple objectives. As DL practitioners often need to optimize not just one but many objectives, this is a blind spot in the algorithmic landscape of HPO. To address this shortcoming, we introduce **PrIM0**, the first HPO algorithm that can integrate multi-objective user beliefs. We show **PrIM0** achieves state-of-the-art performance across 8 DL benchmarks in the multi-objective *and* single-objective setting, clearly positioning itself as the new go-to HPO algorithm for DL practitioners.

## 1 INTRODUCTION

Modern Deep Learning (DL) pipelines (Vaswani et al., 2017; Jumper et al., 2021; Brown et al., 2020) 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) (Bergstra et al., 2011; Li et al., 2017; Falkner et al., 2018; Mallik et al., 2023), many researchers continue to rely on manual tuning (Bouthillier & Varoquaux, 2020), 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 (Ramachandran et al., 2020; Souza et al., 2020; Hvarfner et al., 2022; Mallik et al., 2023). However, for DL, it is often necessary to optimize over several objectives, such as computational cost, training time, latency or fairness (Izquierdo et al., 2021; Schmucker et al., 2020; Salinas et al., 2021; Schneider et al., 2023). Thus, integrating prior knowledge into multi-objective optimization is a crucial research area that remains unexplored. We therefore adapt the desiderata for HPO algorithms for DL (Falkner et al., 2018; Mallik et al., 2023; Franceschi et al., 2025) as follows:

Table 1: Comparison of our algorithm **PrIM0** to prominent categories of multi-objective algorithms with respect to the identified desiderata. The algorithmic categories include Evolutionary algorithms (EA, *e.g.* NSGA-II, SMS-EMOA), multi-objective multi-fidelity algorithms (MOMF, *e.g.*, MOASHA, MO-HyperBand, HyperBand with Random Weights), and multi-objective Bayesian optimization (MO-BO, *e.g.*, BO with random weights, BO with EHVI, ParEGO). A ✓ indicates that the method satisfies the criterion; a ✗ indicates it does not. (✓) denotes partial fulfillment or fulfillment with additional assumptions.

Criterion	RS	EA	MOMF	MO-BO	PrIM0
Utilize cheap approximations	✗	✗	✓	✗	✓
Integrate multi-objective expert priors	✗	✗	✗	✗	✓
Strong anytime performance	✗	✗	✓	✗	✓
Strong final performance	✗	(✓)	(✓)	✓	✓

054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084  
085  
086  
087  
088  
089  
090  
091  
092  
093  
094  
095  
096  
097  
098  
099  
100  
101  
102  
103  
104  
105  
106  
107

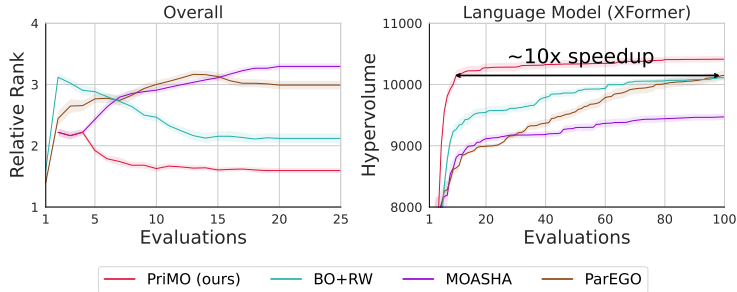


Figure 1: Comparison of PriMO and prominent multi-objective algorithms. [Left] Mean relative ranks across 8 DL benchmarks under all prior conditions averaged. [Right] Mean dominated Hypervolume for tuning the hyperparameters of a language model, demonstrating that PriMO can leverage a good prior to offer speedups of up to  $\sim 10x$ .

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. **Integrate multi-objective expert priors:** Expert prior knowledge of hyperparameters is often available for real-world DL tasks. A modern HPO algorithm must be able to utilize such beliefs over multiple objectives to speed up the optimization and be able to meaningfully recover from misleading prior information.
3. **Strong anytime performance:** Multi-objective HPO algorithms must be compute-efficient, *i.e.*, under limited budget, they must find candidates that significantly improve the dominated Hypervolume.
4. **Strong final performance:** The ultimate goal of HPO is to find the best performing configurations. As budgets grow larger, the algorithms should yield strong solutions.

Table 1 shows that existing HPO algorithms satisfy 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 and also leverages cheap approximations of expensive objective functions. Our **main contributions** are as follows:

- We are the first to consider expert priors for multiple objectives (Section 2) and show that naively adapting existing algorithms is not a robust solution (Section 3).
- We introduce PriMO, a Bayesian optimization algorithm that integrates multi-objective expert priors in its acquisition function and exploits cheap proxy tasks in its initial design (Section 4). As such, PriMO is the first HPO algorithm to meet all the requirements of multi-objective HPO for practical DL (Table 1) and empirically yields up to 10x speedups over existing algorithms (Figure 1).
- We empirically demonstrate state-of-the-art performance of PriMO across a variety of DL benchmarks in the multi-objective *and* single-objective setting (Section 5.3). Furthermore, we show that PriMO is robust to different priors strengths (Section 5.4) and, in an ablation study, we verify that all components of PriMO are helpful and necessary (Section 5.6).

## 2 MULTI-OBJECTIVE HPO WITH EXPERT PRIORS AND CHEAP APPROXIMATIONS

To capture all the above desiderata, we propose the novel problem formulation of minimizing a vector-valued objective function  $f$ , while exploiting cheap approximations of its individual objectives and expert priors. For background on HPO for DL and the multi-objective case see Appendix B, and for a comparison to related problem formulations see Section 6.

**Introducing multi-objective expert priors** To extend expert priors over a single objective (Hvarfner et al., 2022) to the multi-objective 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

$$\pi_{f_i}(\lambda) = \mathbb{P} \left( f_i(\lambda) = \min_{\lambda' \in \Lambda} f_i(\lambda') \right), \quad (1)$$

yielding the compound prior  $\Pi_f(\lambda) = \{\pi_{f_i}(\lambda)\}_{i=1}^n$ , i.e., the set of prior beliefs over the optima of the individual functions that comprise  $f$ . For a discussion on our assumptions on priors see Section 7 and for details on the priors we consider in our experiments see Section 5.1. Further, we provide a discussion on the sources of prior knowledge in Appendix D.

**Integrating multi-objective expert priors and cheap approximations** To also leverage cheap approximations of the individual objectives, let  $\hat{f}_i(\lambda, z)$  denote the low-fidelity proxy for  $f_i$ , where hyperparameters  $\lambda$  are evaluated at the fidelity level  $z$ , where  $f_i(\lambda) = \hat{f}_i(\lambda, z_{max})$ . Therefore, our goal is to solve

$$\arg \min_{\lambda \in \Lambda} f(\lambda) = \arg \min_{\lambda \in \Lambda} \left( \hat{f}_1(\lambda, z_{max}), \dots, \hat{f}_n(\lambda, z_{max}) \right), \quad \text{guided by } \Pi_f(\lambda), \quad (2)$$

using inexpensive evaluations of  $f$ , while addressing the challenge that the priors may be misleading. Since the solution in multi-objective optimization is not a single optimum, but rather a Pareto front of trade-offs between objectives, our formulation seeks to guide the optimization process toward promising regions of this front.

### 3 POOR PERFORMANCE OF THE NAIVE SOLUTION

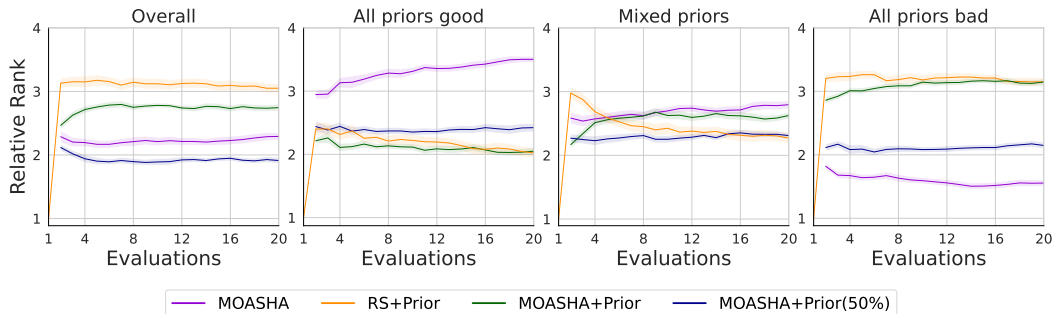


Figure 2: Mean relative ranks  $\pm 1$  standard error across benchmarks and seeds under various prior conditions for randomly sampling from the priors, MOASHA, and adaptations of it that utilize multi-objective expert priors. See Section 5.1 for details on the evaluation protocol.

In this section we study the effect of solving Equation 2 by naively adapting a multi-objective algorithm that can already utilize cheap approximations. We find that this naive solution does not perform robustly across different prior strengths (Figure 2), calling for a better-designed algorithm.

Specifically, we modify the random sampling in multi-objective asynchronous successive halving (MOASHA) (Schmucker et al., 2021), so that configurations are sampled from one of the priors  $\pi_{f_i}$  chosen randomly at every iteration or 50% of iterations. In the presence of good prior knowledge, always sampling from the prior considerably outperforms standard MOASHA, but under misleading priors leads to drastically poor performance. 50% prior-sampling performs better than MOASHA overall, but is unable to effectively utilize good priors as well as 100% prior sampling or even prior-based random search. Therefore, we propose PriMO, to benefit from *good* priors while having the ability to recover from *bad* ones.

## 4 PRIMO: PRIOR INFORMED MULTI-OBJECTIVE OPTIMIZER

In this section, we introduce the first multi-objective HPO algorithm, `PrIMo` (Algorithm 3), that leverages multi-objective user priors and fulfills all the desiderata of modern HPO.

We discuss how `PrIMo`, a Bayesian Optimization algorithm, makes use of multi-objective user priors via its acquisition function (Section 4.1) and cheap approximations of the objective functions with its initial design (Section 4.2). As `PrIMo` yields state-of-the-art for the multi-objective and single-objective setting (Section 5), we discuss how single-objective problems imply a special case of `PrIMo` (Section 4.3). We provide additional details in Appendix C.

### 4.1 INTEGRATING MULTI-OBJECTIVE EXPERT PRIORS INTO BAYESIAN OPTIMIZATION

We first choose one of the priors over multiple objectives uniformly at random during each iteration. We weight the acquisition function of BO (Algorithm 2) with the PDF of the selected prior, raised to an exponent  $\gamma = \exp(-n_{\text{BO}}^2/n_d)$ . Unlike  $\pi$ BO, where  $\gamma = \frac{10}{n}$ , with  $n$  referring to the  $n^{\text{th}}$  iteration, we reduce the overdependence on the prior by setting  $\gamma$  to be inversely proportional to the square of the number of BO samples. To formulate an acquisition function, we convert the vector-valued objective function into a single-objective optimization problem, using a linear scalarization function (Yoon et al., 2009) with randomly sampled weights, which is not only simple but also scalable with the number of objectives  $n$ :

$$\min_{\lambda \in \Lambda} \sum_{i=1}^n w_i \hat{f}_i(\lambda, z_{max}) \quad w_i \sim \mathcal{U}, w_i > 0, \sum_{i=1}^n w_i = 1. \quad (3)$$

Furthermore, to aid in recovery from misleading priors, we incorporate a simple exploration parameter  $\epsilon$ , which controls how often we augment the acquisition function with the prior. Thus, for `PrIMo`'s  $\epsilon$ -BO, with priors over  $n$  objectives, the acquisition function becomes

$$\alpha_{\epsilon\pi}(\lambda, \mathcal{D}) \triangleq \begin{cases} \alpha(\lambda, \mathcal{D}), & \text{with prob. } \epsilon \\ \alpha(\lambda, \mathcal{D}) \cdot \pi_{f_j}(\lambda)^{\exp(-n_{\text{BO}}^2/n_d)}, & \text{with prob. } 1 - \epsilon, j \sim \mathcal{U}(1, \dots, n). \end{cases} \quad (4)$$

### 4.2 AN INITIAL DESIGN TO UTILIZE CHEAP APPROXIMATIONS

To leverage cheap approximations of the objective functions, we propose an initial design strategy (Algorithm 1) that exploits the strengths of multi-fidelity algorithms. Specifically, we use a multi-fidelity algorithm in `PrIMo` to generate strong initial seed points at the maximum fidelity  $z_{max}$  to speed up the optimization in the BO phase afterward.

First, we set a threshold of (equivalent) full function evaluations based on the initial design size. Once this threshold is reached, only maximum fidelity evaluations  $\{(\lambda, \hat{f}(\lambda, z_{max}))\}$  are included in the dataset  $\mathcal{D}$  for use in BO. Next, we choose one of the priors over multiple objectives uniformly at random during each iteration and the sampled initial points then aid the BO along with the decaying prior-augmented acquisition function (Equation 4). We chose to use multi-objective asynchronous successive halving (MOASHA) in our initial design due to its strong performance early on and since as an infinite-horizon optimizer, it is budget invariant, resulting in a single continued optimization run without being restricted to discrete Successive Halving brackets.

216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227**Algorithm 1** Initial design strategy

---

```

1: function init( $n_{\text{init}}, \Lambda, \eta, z_{\text{min}}, z_{\text{max}}, f, \mathbf{w}$ )
2:    $b \leftarrow 0, \mathcal{D} \leftarrow \emptyset$ 
3:   while  $b < n_{\text{init}}$  do
4:      $\lambda, z \leftarrow \text{moasha}(\Lambda, \eta, z_{\text{min}}, z_{\text{max}})$ 
5:      $\mathbf{y} \leftarrow f(\lambda, z)$ 
6:     if  $z = z_{\text{max}}$  then
7:        $\mathbf{y} \leftarrow \mathbf{w}^\top \mathbf{y}$ 
8:        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda, \mathbf{y})\}$ 
9:      $b \leftarrow b + \frac{z}{z_{\text{max}}}$ 
10:  return  $\mathcal{D}$ 

```

---

**Algorithm 2** BO step with multi-obj. priors

---

```

1: function moprior_bo( $\Lambda, \mathcal{D}, \Pi_f, n_{\text{BO}}, \epsilon$ )
2:   Select prior  $\pi_{f_j}$ , where  $j \sim \mathcal{U}(1, \dots, n)$ 
3:    $\gamma \leftarrow \exp(-n_{\text{BO}}^2/n_d)$ 
4:    $u \sim \mathcal{U}(0, 1)$ 
5:   if  $u < \epsilon$  then
6:      $\tilde{\alpha}(\lambda) := \alpha(\lambda, \mathcal{D})$ 
7:   else
8:      $\tilde{\alpha}(\lambda) := \alpha(\lambda, \mathcal{D}) \cdot \pi_{f_j}(\lambda)^\gamma$ 
9:    $\lambda \leftarrow \arg \max_{\lambda \in \Lambda} \tilde{\alpha}(\lambda)$ 
10:  return  $\lambda$ 

```

---

228  
229**Algorithm 3** PriMO

---

```

1: Input: Objective  $f$ , search space  $\Lambda$  with dimension  $n_d$ , priors  $\Pi_f = \{\pi_{f_i}(\lambda)\}_{i=1}^n$ , initial design size  $n_{\text{init}}$ , reduction factor  $\eta$ , fidelity range  $[z_{\text{min}}, z_{\text{max}}]$ , budget  $B$  and exploration parameter  $\epsilon$ .
2: function PriMO(Input)
3:   Sample weights  $\mathbf{w} \sim \mathcal{U}(0, 1)^n$  and normalize
4:    $\mathcal{D} \leftarrow \text{init}(n_{\text{init}}, \Lambda, \eta, z_{\text{min}}, z_{\text{max}}, f, \mathbf{w})$ 
5:    $b \leftarrow n_{\text{init}}, n_{\text{BO}} \leftarrow 0$ 
6:   while  $b < B$  do
7:      $\lambda_{\text{new}} \leftarrow \text{moprior\_bo}(\Lambda, \mathcal{D}, \Pi_f, n_{\text{BO}}, \epsilon)$ 
8:      $n_{\text{BO}} \leftarrow n_{\text{BO}} + 1$ 
9:      $\mathbf{y} \leftarrow f(\lambda_{\text{new}}, z_{\text{max}})$ 
10:     $\mathbf{y} \leftarrow \mathbf{w}^\top \mathbf{y}$ 
11:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda_{\text{new}}, \mathbf{y})\}$ 
12:     $b \leftarrow b + 1$ 
13:  return  $\mathcal{P}_f(\mathcal{D})$ 

```

---

242  
243

## 4.3 THE SINGLE-OBJECTIVE SETTING

244  
245  
246  
247  
248  
249  
250

We also adapt PriMO to the single-objective setting as a special case of the original multi-objective design. The initial design strategy of PriMO replaces MOASHA with the ASHA scheduler. Instead of selecting a prior at random, as in the multi-objective case, we sample from the prior over the single objective. The  $\epsilon$ -greedy prior-augmented Bayesian Optimization phase remains unchanged.

251

252

## 4.4 DISCUSSION

253

254

255

256

257

258

**Runtime Analysis** We use scalarization to transform the multi-objective problem to a single-objective one in each BO step, therefore our runtime behaviour corresponds to classic BO. In our  $\epsilon$ -BO, we use a Gaussian Processes which has the asymptotic time complexity of  $\mathcal{O}(n^3)$ . We note that the runtime of a single BO step is negligible in comparison to the model evaluation cost (for an example see Table 2).

259

260

Table 2: Comparison of the average HPO sampling times of PriMO and BO+RW, and the evaluation cost of language modeling with a large transformer on the 1B word benchmark.

261

262

263

264

265

266

267

268

269

Algorithm	Avg. HPO sampling time (s)	Avg. model evaluation time (s)
PriMO	10.28 $\pm$ 1	8306.68
BO+RW	9.80 $\pm$ 1	8740.58

**Behavior under highly correlated priors** When the priors for two objectives are strongly positively correlated, the overall dependence on their quality increases. If the priors are helpful for both objectives, PriMO’s performance will improve, while the converse is true if the priors for both objectives are misleading. However, under mixed prior conditions, their

effects may cancel out or the prior over one objective may suppress the effect of the other depending on the strength of the correlation between them.

## 5 EXPERIMENTS

To empirically demonstrate that `PrIMO` fulfills the desiderata outlined in the [Introduction](#), we address the following research questions.

**RQ1:** Does `PrIMO` outperform strong multi-objective baselines in terms of anytime and final performance?

**RQ2:** Does `PrIMO` maintain state-of-the-art performance in the single-objective setting?

**RQ3:** Can `PrIMO` effectively leverage multi-objective expert priors?

**RQ4:** Does `PrIMO` recover from misleading priors and maintain its robustness?

**RQ5:** Can `PrIMO` effectively leverage cheap approximations with its initial design strategy?

**RQ6:** Are all components of `PrIMO` necessary and helpful?

After providing details on our experimental setup and baselines (Section 5.1 and 5.2), we show `PrIMO`'s state-of-the-art performance in the multi-objective and single-objective setting in Section 5.3 (answering **RQ1**, **RQ2**, and **RQ3**). We then discuss its robustness across all prior conditions in Section 5.4 (answering **RQ4**), and, finally, provide an ablation study in Section 5.6 (answering **RQ5** and **RQ6**). [In Appendix I we provide detailed case-level analysis based on Pareto fronts, and hypervolume plots under bad and overall prior combinations.](#)

### 5.1 EXPERIMENTAL SETUP

**Evaluation protocol** We base our multi-objective evaluation on the mean dominated hypervolume across 25 seeds and report relative rankings of the algorithms across budgets. Each optimizer-benchmark-seed combination was run for 20 equivalent full function evaluations, corresponding to typical budgets in practical Deep Learning. We give more details on our evaluation protocol in [Appendix H](#), provide additional experiments and analysis in [Appendix I](#), and conduct a statistical significance analysis in [Appendix J](#).

**Benchmarks** We use 8 benchmarks representing image classification, language translation and learning curves for Deep Neural Networks. We chose 4 `LCBench` benchmarks from the `Yahpo-Gym Suite` ([Pfisterer et al., 2022](#)) and 4 from the `PD1` ([Wang et al., 2024](#)) set of benchmarks. We select the corresponding validation error and training cost metrics as objectives. In [Appendix G](#) we provide full details on the 8 benchmarks.

**Priors** We study the effect of different prior conditions: both objectives have good priors, both objectives have bad priors, the average over mixed good and bad prior combinations, as well as the average over all combinations. For generating priors, we follow the protocols in the literature on single-objective optimization ([Appendix E](#)).

### 5.2 BASELINES

We give an overview of all our baselines here and provide additional details in [Appendix F](#).

**Multi-objective baselines from the literature** 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 ([Knowles, 2006](#)), multi-fidelity optimizers such as HyperBand with Random Weights (HB+RW) ([Schmucker et al., 2020](#)) and multi-objective asynchronous successive halving (MOASHA) ([Schmucker et al., 2021](#)), and an evolutionary algorithm – NSGA-II ([Deb et al., 2002](#)).

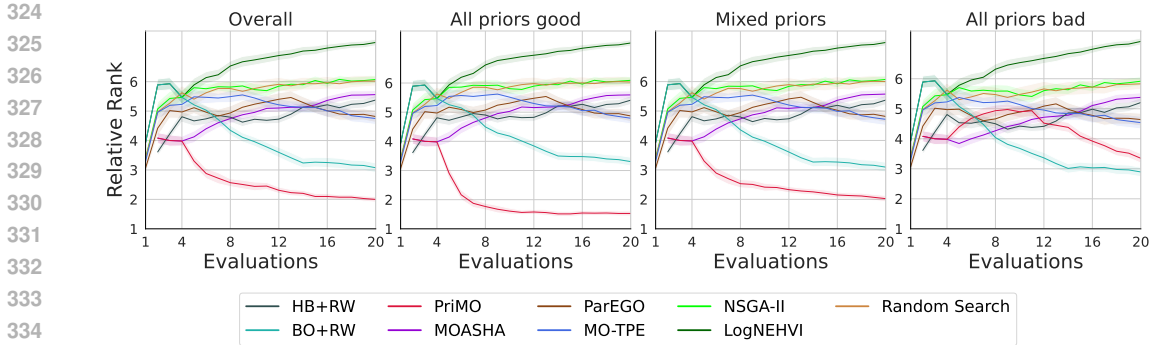


Figure 3: Mean relative ranks  $\pm 1$  standard error of **PriMO** and prominent multi-objective algorithms across benchmarks and seeds under various prior conditions.

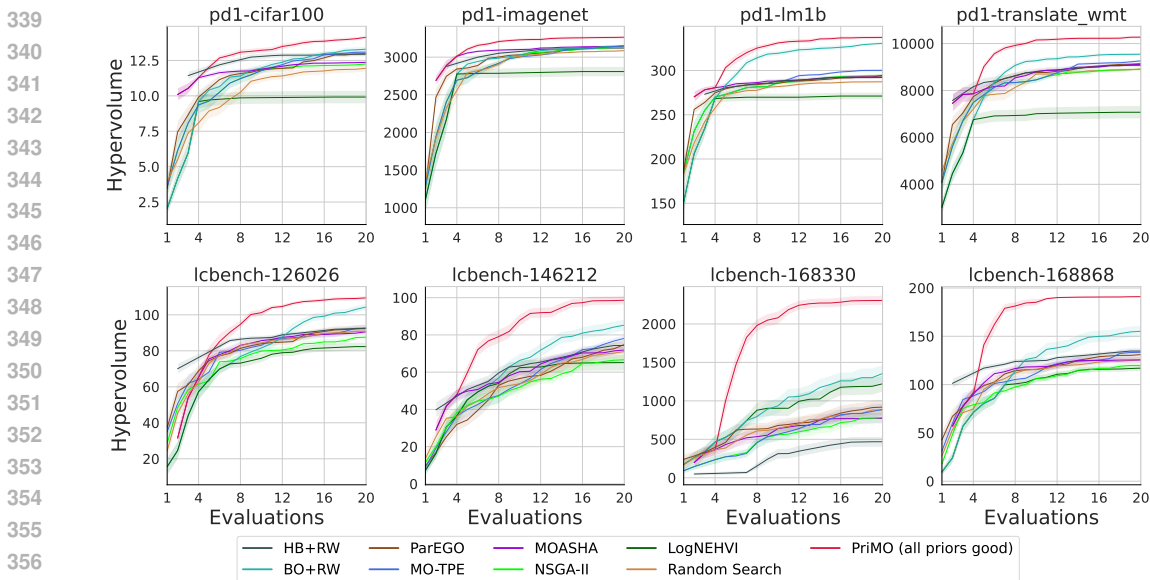


Figure 4: Mean dominated Hypervolume  $\pm 1$  standard error of **PriMO** and prominent multi-objective algorithms across seeds for each benchmark. **PriMO** is under all good priors setting here. See Appendix I for additional Hypervolume plots.

**Additional multi-objective baselines we constructed** While no multi-objective approaches exist in the literature that leverage expert priors, we augment single-objective approaches to provide strong prior-based baselines. We modify such single-objective approaches (RS + Prior, MOASHA + Prior,  $\pi$ BO (Hvarfner et al., 2022), Priorband (Mallik et al., 2023)) to randomly chose and sample from one of the multi-objective priors at each iteration. We further augment  $\pi$ BO with random scalarizations and modify Priorband’s ensemble sampling policy using scalarized incumbents for MO to build MO-Priorband.

**Single-objective baselines** For our experiments in the single-objective setting, we compare **PriMO** against BO and HyperBand (Li et al., 2017), and existing single-objective algorithms that can leverage expert priors, i.e., Priorband-BO and  $\pi$ BO.

### 5.3 PRIMO ACHIEVES STATE-OF-THE-ART PERFORMANCE

**Multi-objective setting** Figure 3 demonstrates that, overall, **PriMO** maintains the strongest anytime performance and achieves the best final performance in terms of relative rankings across all benchmarks (**RQ1**). Under good priors the relative ranking gap

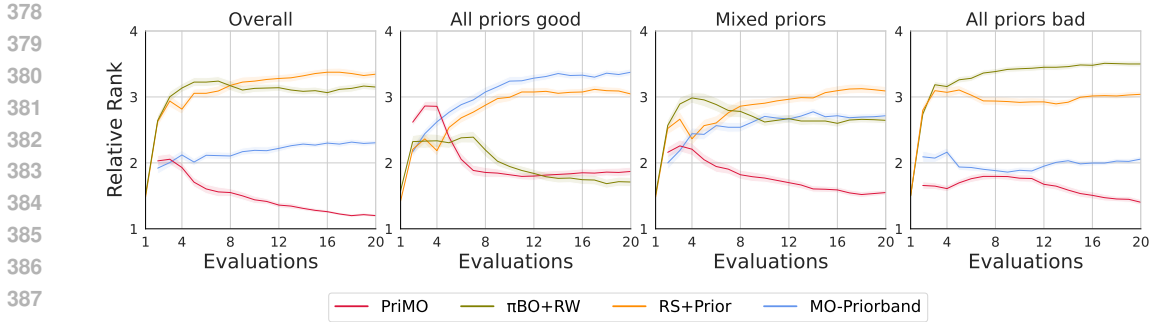


Figure 5: Mean relative ranks  $\pm 1$  standard error of baselines we constructed to use multi-objective priors and PriMO across benchmarks and seeds under various prior conditions.

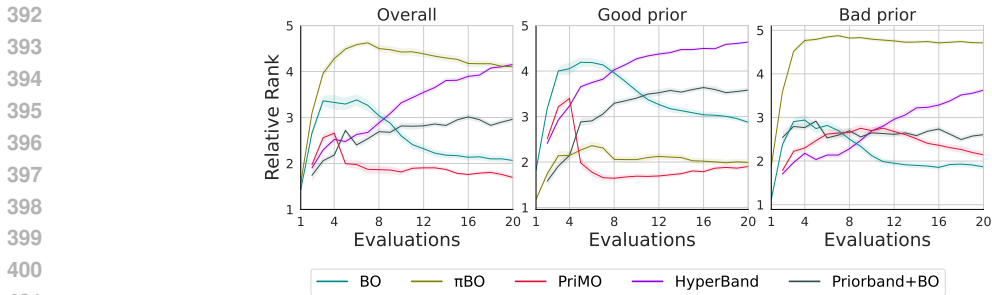


Figure 6: Mean relative ranks  $\pm 1$  standard error in the single-objective setting across benchmarks and seeds under various prior conditions.

between PriMO and the second best optimizer, BO+RW, is even more pronounced. PriMO shows strong starts across all benchmarks, with respect to the mean dominated Hypervolume under good priors (Figure 4), and is the best performing algorithm across all benchmarks early on. We attribute this to the initial design which provides a strong head-start before the BO phase. Using the configurations sampled by the initial design, the BO phase of PriMO maintains its strong anytime performance, and is able to effectively utilize the priors, achieving state-of-the-art final performance across all benchmarks (RQ3). Figure 5 clearly shows that, overall, PriMO is anytime better compared to prior-based MO adapted baselines and outperforms MO-Priorband by a wide margin. Under good priors, PriMO and  $\pi$ BO+RW are usually 2 of the best optimizers on average. 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  $\gamma$  setting (see Section 4).

**Single-objective setting** Figure 6 shows PriMO’s state-of-the-art performance in the single-objective setting. PriMO is overall the best choice for single-objective HPO demonstrating the strongest anytime and final performance (RQ2). Under good priors, Priorband and  $\pi$ BO show stronger starts as is expected with their prior-based initial sampling strategies, but are quickly outperformed by PriMO within a few full function evaluations.

#### 5.4 PRIMO IS ROBUST TO PRIOR CONDITIONS

Under all bad priors in Figure 3 we see that after a poor initial performance, PriMO shows remarkable recovery and by the end of the optimization budget, nearly catches up with BO+RW, resulting in a competitive final performance (RQ4). Mixed and overall prior conditions show similar trends where PriMO is the best performing algorithm early on and ranks significantly better than all other baselines by the final iteration. Compared to MO-adapted prior-based baselines in Figure 5 (all priors bad), we observe that PriMO not only has good anytime performance, but also significantly outperforms all baselines by the end of the optimization run across most benchmarks. Thus,  $\pi$ BO+RW, despite being able to leverage

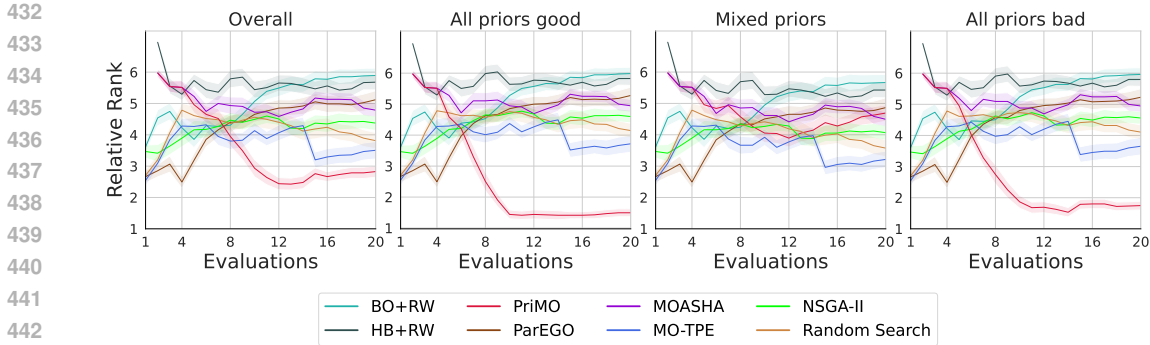


Figure 7: Mean relative ranks  $\pm 1$  standard error of **PriMO** and prominent multi-objective algorithms across the 3 HW-GPT-Bench tasks under various prior conditions. For an explanation of **PriMO**'s strong performance under all priors bad, see Section 5.5.

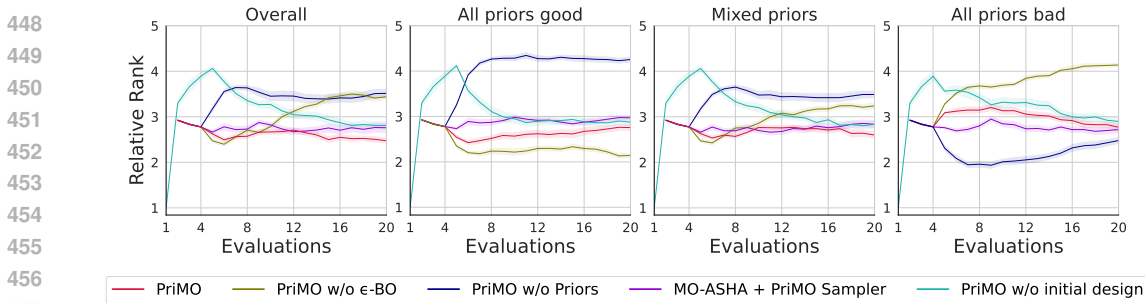


Figure 8: Mean relative ranks  $\pm 1$  standard error of various ablations of **PriMO** across benchmarks and seeds under various prior conditions.

good priors well, is quite prone to misleading priors and, overall, significantly (Appendix J) less robust compared to **PriMO**.

### 5.5 CASE STUDY: GPT2

We conduct a case study on the GPT2-based Neural Architecture Search benchmark HW-GPT-Bench (Sukthanker et al., 2024). In Figure 7, we compare the performance of **PriMO** on HW-GPT-Bench against our baselines and find that **PriMO** achieves the strongest performance. We notice that **PriMO** performs surprisingly strong when all priors are misleading for their respective objective. Table 3 shows that a bad prior for perplexity is strongly positively correlated with the FLOPS objective. Therefore, a bad prior for perplexity serves as a good prior for FLOPS.

Table 3: Spearman’s rank correlation between objectives and bad priors on HW-GPT-Bench.

	Bad prior for FLOPS	Bad prior for perplexity
FLOPS	-0.06	0.56
Perplexity	0.11	-0.65

### 5.6 ALL COMPONENTS OF **PriMO** ARE HELPFUL

We consider different design ablations of **PriMO** in Figure 8 to answer **RQ5** and **RQ6** and, overall, find that all components of **PriMO** are important. We divide **PriMO** into its constituent components, namely – the initial design, MO-Priors and the  $\epsilon$ -BO (including the random weights), and label each design ablation with respect to the component(s) that were removed from **PriMO**.

We find that the initial design strategy gives a substantial early boost, as all ablations of PriMO using the initial design start off much stronger than PriMO without the initial design (RQ5). This initial advantage tends to persist for most ablations until the BO phase for PriMO without the initial design. 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  $\epsilon$ -BO.

PriMO without Priors and PriMO without  $\epsilon$ -BO are two of the worst performing ablations overall, highlighting the importance of a prior-based BO design, coupled with the  $\epsilon$ -greedy optimization strategy in PriMO’s BO. We further notice that MOASHA, when augmented with PriMO’s  $\epsilon$ -BO sampler is surprisingly robust under all prior conditions, although never the most competitive design. These findings, taken together, support our final design choice for PriMO (RQ6).

## 6 RELATED WORK

Multi-objective optimization traditionally considers large budgets (Deb, 2013; Deb et al., 2002; Knowles, 2006; Golovin & Zhang, 2020); in DL, however, budgets are constrained. Thus, to make HPO for DL feasible, specialized strategies have been proposed. We discuss the strategies most related to ours here and provide a more expansive discussion in Appendix B.

**User priors for single-objective optimization** The integration of expert 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 (Mallik et al., 2023), 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 multi-objective setting, but found it does not perform well (Section 5).  $\pi$ BO (Hvarfner et al., 2022), 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 5).

**Exploiting cheap approximations for multi-objective optimization** While exploiting expert priors is novel for multi-objective HPO, cheap proxies have been explored (Schmucker et al., 2020; Salinas et al., 2021; Schmucker et al., 2021). However, in contrast to our approach, not as an initial design. In our ablation study, we show that integrating cheap approximations as an initial design performs better overall (Section 5.6).

## 7 LIMITATIONS

In line with previous work (Souza et al., 2020; Hvarfner et al., 2022; Mallik et al., 2023), we only consider Gaussian distributions for our priors, **where the mean corresponds to a configuration that is expected to perform well**, although PriMO supports priors with any distribution. While it may be more beneficial in the multi-objective setting to generate priors based on an approximate Pareto front, this remains a non-trivial challenge. However, it is unclear how experts would define priors over a Pareto front directly, and PriMO already achieves state-of-the-art performance using simple priors. Additionally, instead of linear scalarization, an approach such as Hypervolume scalarization (Golovin & Zhang, 2020) could be beneficial to PriMO as it has provable guarantees to converge to non-convex Pareto fronts.

## 8 CONCLUSION

PriMO distinguishes itself as the first algorithm to integrate multi-objective expert 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 Deep Learning.

540 REPRODUCIBILITY STATEMENT

541  
542 To ensure reproducibility, we adhere to and include a reproducibility checklist in Appendix A.  
543 In our justifications for the checks, we link to all relevant sections, appendices, and supple-  
544 mentary materials. We considered the checklists in use by NeurIPS and AutoML-Conf, but  
545 chose the latter as it is more comprehensive.

547 REFERENCES

- 548  
549 K. Abe, Y. Wang, and S. Watanabe. Tree-structured parzen estimator can solve black-box  
550 combinatorial optimization more efficiently, 2025. URL <https://arxiv.org/abs/2507.08053>.
- 551  
552 S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Un-  
553 expected Improvements to Expected Improvement for Bayesian Optimization.  
554 *Advances in Neural Information Processing Systems*, 36:20577–20612, Decem-  
555 ber 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/hash/419f72cbd568ad62183f8132a3605a2a-Abstract-Conference.html](https://proceedings.neurips.cc/paper_files/paper/2023/hash/419f72cbd568ad62183f8132a3605a2a-Abstract-Conference.html).
- 556  
557 N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyperband for scalable, robust  
558 and efficient Hyperparameter Optimization. In Z. Zhou (ed.), *Proceedings of the 30th*  
559 *International Joint Conference on Artificial Intelligence (IJCAI’21)*, pp. 2147–2153, 2021.
- 560  
561 S. Belakaria, A. Deshwal, and J. Doppa. Max-value entropy search for multi-objective  
562 Bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alche Buc,  
563 E. Fox, and R. Garnett (eds.), *Proceedings of the 33rd International Conference on*  
564 *Advances in Neural Information Processing Systems (NeurIPS’19)*. Curran Associates,  
565 2019.
- 566  
567 S. Belakaria, A. Deshwal, and J. R. Doppa. Information-theoretic multi-objective bayesian  
568 optimization with continuous approximations, 2020a. URL <https://arxiv.org/abs/2009.05700>.
- 569  
570 S. Belakaria, A. Deshwal, and J. R. Doppa. Multi-Fidelity Multi-Objective Bayesian  
571 Optimization: An Output Space Entropy Search Approach. *Proceedings of the AAAI*  
572 *Conference on Artificial Intelligence*, 34(06):10035–10043, April 2020b. ISSN 2374-3468,  
573 2159-5399. doi: 10.1609/aaai.v34i06.6560. URL <https://arxiv.org/pdf/2011.01542>.
- 574  
575 E. Bergman, N. Mallik, C. Hvarfner, and S. Basu. mf-prior-bench, March 2025. URL  
<https://github.com/automl/mf-prior-bench>.
- 576  
577 J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter op-  
578 timization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger  
579 (eds.), *Proceedings of the 25th International Conference on Advances in Neural Information*  
580 *Processing Systems (NeurIPS’11)*, pp. 2546–2554. Curran Associates, 2011.
- 581  
582 N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on  
583 dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669,  
584 2007.
- 585  
586 Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris  
587 Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post,  
588 Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 workshop on  
589 statistical machine translation. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann,  
590 Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina  
591 (eds.), *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 1–  
592 46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:  
593 [10.18653/v1/W15-3001](https://aclanthology.org/W15-3001/). URL <https://aclanthology.org/W15-3001/>.
- 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.

- 594 T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan,  
595 P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan,  
596 R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin,  
597 S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and  
598 D. Amodei. Language models are few-shot learners. In *Larochelle et al. (2020)*, pp.  
599 1877–1901.
- 600 C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One  
601 billion word benchmark for measuring progress in statistical language modeling, 2014.  
602 URL <https://arxiv.org/abs/1312.3005>.
- 603 S. Daulton, M. Balandat, and E. Bakshy. Differentiable Expected Hypervolume Improvement  
604 for Parallel Multi-Objective Bayesian Optimization, October 2020. URL <http://arxiv.org/abs/2006.05078>. arXiv:2006.05078 [stat].
- 605 K. Deb. Multi-objective optimization. In *Search Methodologies: Introductory Tutorials in*  
606 *Optimization and Decision Support Techniques*, pp. 403–449. Springer, 2013.
- 607 K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic  
608 algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197,  
609 April 2002. ISSN 1941-0026. doi: 10.1109/4235.996017. URL <https://ieeexplore.ieee.org/document/996017>.
- 610 M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization Assisted by*  
611 *Gaussian Random Field Metamodels*. PhD thesis, Universität Dortmund, 2005.
- 612 S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient Hyperparameter Opti-  
613 mization at scale. In J. Dy and A. Krause (eds.), *Proceedings of the 35th International*  
614 *Conference on Machine Learning (ICML’18)*, volume 80, pp. 1437–1446. Proceedings of  
615 Machine Learning Research, 2018.
- 616 L. Franceschi, M. Donini, V. Perrone, A. Klein, C. Archambeau, M. Seeger, M. Pontil,  
617 and P. Frasconi. Hyperparameter Optimization in Machine Learning, April 2025. URL  
618 <http://arxiv.org/abs/2410.22854>. arXiv:2410.22854 [stat].
- 619 A. M. Geburek, N. Mallik, D. Stoll, X. Bouthillier, and F. Hutter. LMEMs for post-hoc  
620 analysis of HPO benchmarking. In *AutoML Conference 2024 (Workshop Track)*, 2024.  
621 URL <https://openreview.net/forum?id=AAASG6BNv1>.
- 622 D. Golovin and Q. Zhang. Random Hypervolume Scalarizations for Provable Multi-  
623 Objective Black Box Optimization, June 2020. URL <http://arxiv.org/abs/2006.04655>.  
624 arXiv:2006.04655 [cs].
- 625 K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In  
626 *Proceedings of the International Conference on Computer Vision and Pattern Recognition*  
627 *(CVPR’16)*, pp. 770–778. Computer Vision Foundation and IEEE Computer Society, IEEE,  
628 2016.
- 629 D. Hernández-Lobato, J. Hernández-Lobato, A. Shah, and R. Adams. Predictive Entropy  
630 Search for Multi-objective Bayesian Optimization. In M. Balcan and K. Weinberger  
631 (eds.), *Proceedings of the 33rd International Conference on Machine Learning (ICML’17)*,  
632 volume 48, pp. 1492–1501. Proceedings of Machine Learning Research, 2016.
- 633 C. Hvarfner, D. Stoll, A. Souza, L. Nardi, M. Lindauer, and F. Hutter.  $\pi$ BO: Augmenting  
634 Acquisition Functions with User Beliefs for Bayesian Optimization. In *The Tenth Interna-*  
635 *tional Conference on Learning Representations (ICLR’22)*. ICLR, 2022. Published online:  
636 [iclr.cc](http://iclr.cc).
- 637 F. Irshad, S. Karsch, and A. Döpp. Leveraging Trust for Joint Multi-Objective and Multi-  
638 Fidelity Optimization. *Machine Learning: Science and Technology*, 5(1):015056, March  
639 2024. ISSN 2632-2153. doi: 10.1088/2632-2153/ad35a4. URL <http://arxiv.org/abs/2112.13901>. arXiv:2112.13901 [cs].
- 640  
641  
642  
643  
644  
645  
646  
647

- 648 S. Izquierdo, J. Guerrero-Viu, S. Hauns, G. Miotto, S. Schrodi, A. Biedenkapp, T. Elsken,  
649 D. Deng, M. Lindauer, and F. Hutter. Bag of baselines for multi-objective Joint Neural  
650 Architecture Search and Hyperparameter Optimization. In M. Meila and T. Zhang (eds.),  
651 *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, volume  
652 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.
- 653 K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and Hyperparameter  
654 Optimization. In A. Gretton and C. Robert (eds.), *Proceedings of the Seventeenth Inter-  
655 national Conference on Artificial Intelligence and Statistics (AISTATS'16)*, volume 51.  
656 *Proceedings of Machine Learning Research*, 2016.
- 657 Y. Jin and J. Kacprzyk (eds.). *Multi-Objective Machine Learning*, volume 16 of *Studies in  
658 Computational Intelligence*. Springer, Berlin, Heidelberg, 2006. ISBN 978-3-540-30676-4  
659 978-3-540-33019-6. doi: 10.1007/3-540-33019-4. URL [http://link.springer.com/10.  
660 1007/3-540-33019-4](http://link.springer.com/10.1007/3-540-33019-4).
- 661 D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black box  
662 functions. *Journal of Global Optimization*, 13:455–492, 1998.
- 663 D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal  
664 of Global Optimization*, 21:345–383, 2001.
- 665 J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunya-  
666 suvunakool, R. Bates, A. Zidek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl,  
667 A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen,  
668 D. A. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer,  
669 S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Has-  
670 sabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589,  
671 2021.
- 672 K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian Opti-  
673 misation with Continuous Approximations. In D. Precup and Y. Teh (eds.), *Proceedings  
674 of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, pp.  
675 1799–1808. *Proceedings of Machine Learning Research*, 2017.
- 676 D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The Third  
677 International Conference on Learning Representations (ICLR'15)*. ICLR, 2015. Published  
678 online: [iclr.cc](http://iclr.cc).
- 679 J. D. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for  
680 expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary  
681 Computation*, 10(1):50–66, 2006.
- 682 A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report,  
683 University of Toronto, 2009.
- 684 H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin (eds.). *Proceedings of the  
685 34th International Conference on Advances in Neural Information Processing Systems  
686 (NeurIPS'20)*, 2020. Curran Associates.
- 687 B. Lefaudeux, F. Massa, D. Liskovich, W. Xiong, V. Caggiano, S. Naren, M. Xu, J. Hu,  
688 M. Tintore, S. Zhang, P. Labatut, D. Haziza, L. Wehrstedt, J. Reizenstein, and G. Sizov.  
689 xFormers: A modular and hackable transformer modelling library. [https://github.com/  
690 facebookresearch/xformers](https://github.com/facebookresearch/xformers), 2022.
- 691 L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-  
692 based configuration evaluation for Hyperparameter Optimization. In *The Fifth Interna-  
693 tional Conference on Learning Representations (ICLR'17)*. ICLR, 2017. Published online:  
694 [iclr.cc](http://iclr.cc).
- 695 L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, and  
696 A. Talwalkar. A system for massively parallel hyperparameter tuning. In I. Dhillon,  
697 D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems 2*,  
698 volume 2, 2020.

- 702 N. Mallik, C. Hvarfner, E. Bergman, D. Stoll, M. Janowski, M. Lindauer, L. Nardi, and  
703 F. Hutter. PriorBand: Practical hyperparameter optimization in the age of deep learning.  
704 In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.),  
705 *Proceedings of the 37th International Conference on Advances in Neural Information*  
706 *Processing Systems (NeurIPS'23)*. Curran Associates, 2023.
- 707 Y. Nesterov. A method of solving a convex programming problem with convergence rate  
708  $O(1/\sqrt{k})$ . *Soviet Mathematics Doklady*, 27:372–376, 1983.
- 709 Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi. Multiobjective tree-structured parzen  
710 estimator for computationally expensive optimization problems. In J. Ceberio (ed.),  
711 *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20)*, pp.  
712 533–541. ACM Press, 2020.
- 713 Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi. Multiobjective Tree-  
714 Structured Parzen Estimator. *Journal of Artificial Intelligence Research*, 73:1209–1250,  
715 April 2022. ISSN 1076-9757. doi: 10.1613/jair.1.13188. URL [https://www.jair.org/  
716 index.php/jair/article/view/13188](https://www.jair.org/index.php/jair/article/view/13188).
- 717 B. Paria, K. Kandasamy, and B. Póczos. A Flexible Framework for Multi-Objective Bayesian  
718 Optimization using Random Scalarizations, June 2019. URL [http://arxiv.org/abs/  
719 1805.12168](http://arxiv.org/abs/1805.12168). arXiv:1805.12168 [cs].
- 720 F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, and B. Bischl. YAHPO Gym – An  
721 Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization,  
722 July 2022. URL <http://arxiv.org/abs/2109.03670>. arXiv:2109.03670 [cs].
- 723 W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective Optimization on a  
724 Limited Budget of Evaluations Using Model-Assisted  $\mathcal{S}$ -Metric Selection. In G. Rudolph,  
725 T. Jansen, N. Beume, S. Lucas, and C. Poloni (eds.), *Parallel Problem Solving from Nature*  
726 *– PPSN X*, pp. 784–794, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-87700-4. doi:  
727 10.1007/978-3-540-87700-4\_78.
- 728 A. Ramachandran, S. Gupta, S. Rana, C. Li, and S. Venkatesh. Incorporating expert prior  
729 in Bayesian optimisation via space warping. *Knowledge-Based Systems*, 195, 2020.
- 730 S. Riezler and M. Haggmann. *Validity, Reliability, and Significance: Empirical Methods*  
731 *for NLP and Data Science*. Synthesis Lectures on Human Language Technologies.  
732 Springer International Publishing, Cham, 2022. ISBN 978-3-031-01055-2 978-3-031-  
733 02183-1. doi: 10.1007/978-3-031-02183-1. URL [https://link.springer.com/10.1007/  
734 978-3-031-02183-1](https://link.springer.com/10.1007/978-3-031-02183-1).
- 735 A. Roy, M. Saffar, A. Vaswani, and D. Grangier. Efficient content-based sparse attention with  
736 routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–  
737 68, 2021. doi: 10.1162/tacl.a.00353. URL [https://aclanthology.org/2021.tacl-1.4/  
738](https://aclanthology.org/2021.tacl-1.4/).
- 739 O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy,  
740 A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet large scale visual recognition  
741 challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- 742 D. Salinas, V. Perrone, O. Cruchant, and C. Archambeau. A multi-objective perspective on  
743 jointly tuning hardware and hyperparameters, June 2021. URL [http://arxiv.org/abs/  
744 2106.05680](http://arxiv.org/abs/2106.05680). ICLR Workshop on Neural Architecture Search.
- 745 R. Schmucker, M. Donini, V. Perrone, M. Zafar, and C. Archambeau. Multi-objective  
746 multi-fidelity hyperparameter optimization with application to fairness. In R. Calandra,  
747 J. Clune, E. Grant, J. Schwarz, J. Vanschoren, F. Visin, and J. Wang (eds.), *NeurIPS*  
748 *2020 Workshop on Meta-Learning*, 2020.
- 749 R. Schmucker, M. Donini, M. Zafar, D. Salinas, and C. Archambeau. Multi-objective  
750 asynchronous successive halving. *arXiv:2106.12639 [stat.ML]*, 2021.

- 756 L. Schneider, B. Bischl, and J. Thomas. Multi-objective optimization of performance  
757 and interpretability of tabular supervised machine learning models. In *Proceedings of*  
758 *the Genetic and Evolutionary Computation Conference, GECCO '23*, pp. 538–547, New  
759 York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi:  
760 10.1145/3583131.3590380. URL <https://doi.org/10.1145/3583131.3590380>.
- 761 A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Prior-guided  
762 Bayesian optimization. In Larochelle et al. (2020).
- 763 A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Bayesian  
764 optimization with a prior for the optimum. In N. Oliver, F. Pérez-Cruz, S. Kramer,  
765 J. Read, and J. A. Lozano (eds.), *Machine Learning and Knowledge Discovery in Databases.*  
766 *Research Track*, volume 12975 of *Lecture Notes in Artificial Intelligence*, pp. 265–296.  
767 Springer-Verlag, 2021.
- 768 N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated  
769 Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Septem-  
770 ber 1994. ISSN 1063-6560. doi: 10.1162/evco.1994.2.3.221. URL [https://](https://doi.org/10.1162/evco.1994.2.3.221)  
771 [doi.org/10.1162/evco.1994.2.3.221](https://doi.org/10.1162/evco.1994.2.3.221). eprint: [https://direct.mit.edu/evco/article-](https://direct.mit.edu/evco/article-pdf/2/3/221/1492770/evco.1994.2.3.221.pdf)  
772 [pdf/2/3/221/1492770/evco.1994.2.3.221.pdf](https://direct.mit.edu/evco/article-pdf/2/3/221/1492770/evco.1994.2.3.221.pdf).
- 773 D. Stoll, N. Mallik, E. Bergman, S. Schrodi, S. Basu, A. M. Geburek, T. Abou Chakra,  
774 S. Garibov, G. Gaur, N. Alipour, M. Janowski, C. Hvarfner, T. Carstensen, E. Mekić,  
775 D. Rogalla, R. Binxin, and F. Hutter. Neural Pipeline Search (NePS), April 2025. URL  
776 <https://github.com/automl/neps>.
- 777 Rhea Sanjay Sukthanker, Arber Zela, Benedikt Staffler, Aaron Klein, Lennart Purucker, Jörg  
778 K. H. Franke, and Frank Hutter. Hw-gpt-bench: Hardware-aware architecture benchmark  
779 for language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet,  
780 J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*,  
781 volume 37, pp. 60776–60834. Curran Associates, Inc., 2024. doi: 10.52202/079017-1944.  
782 URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/file/](https://proceedings.neurips.cc/paper_files/paper/2024/file/6ffdbf064df51857eb802a904aaaba63-Paper-Datasets_and_Benchmarks_Track.pdf)  
783 [6ffdbf064df51857eb802a904aaaba63-Paper-Datasets\\_and\\_Benchmarks\\_Track.pdf](https://proceedings.neurips.cc/paper_files/paper/2024/file/6ffdbf064df51857eb802a904aaaba63-Paper-Datasets_and_Benchmarks_Track.pdf).
- 784 J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5 2:  
785 99–114, 1949. URL <https://api.semanticscholar.org/CorpusID:806596>.
- 786 J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine  
787 learning. *SIGKDD Explorations*, 15(2):49–60, 2014.
- 788 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and  
789 I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio,  
790 H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Proceedings of the*  
791 *31st International Conference on Advances in Neural Information Processing Systems*  
792 *(NeurIPS'17)*. Curran Associates, Inc., 2017.
- 793 Z. Wang, G. E. Dahl, K. Swersky, C. Lee, Z. Nado, J. Gilmer, J. Snoek, and Z. Ghahramani.  
794 Pre-trained Gaussian Processes for Bayesian Optimization. *Journal of Machine Learning*  
795 *Research*, 25(212):1–83, 2024. URL <http://jmlr.org/papers/v25/23-0269.html>.
- 796 K. Yang, M. Emmerich, A. Deutz, and T. Bäck. Multi-Objective Bayesian Global Optimiza-  
797 tion using expected hypervolume improvement gradient. *Swarm and Evolutionary Com-*  
798 *putation*, 44:945–956, February 2019. ISSN 2210-6502. doi: 10.1016/j.swevo.2018.10.007.  
799 URL <https://www.sciencedirect.com/science/article/pii/S2210650217307861>.
- 800 M. Yoon, Y. Yun, and H. Nakayama. *Sequential Approximate Multiobjective Optimization*  
801 *Using Computational Intelligence*. Vector Optimization. Springer, Berlin, Heidelberg,  
802 2009. ISBN 978-3-540-88909-0 978-3-540-88910-6. doi: 10.1007/978-3-540-88910-6. URL  
803 <https://link.springer.com/10.1007/978-3-540-88910-6>.
- 804 S. Zagoruyko and N. Komodakis. Wide residual networks. In Edwin R. Hancock Richard  
805 C. Wilson and William A. P. Smith (eds.), *Proceedings of the 27th British Machine*  
806 *Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, 2016. ISBN 1-901725-59-6. doi:  
807 10.5244/C.30.87. URL <https://dx.doi.org/10.5244/C.30.87>.

810 Michael R Zhang, Nishkrit Desai, Juhan Bae, and Jonathan Lorraine. Using Large Language  
811 Models for Hyperparameter Optimization. *NeurIPS Workshop on Foundation Models for*  
812 *Decision Making*, 2023.

813

814 Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decom-  
815 position. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi:  
816 10.1109/TEVC.2007.892759.

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

820

821 E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms — A  
822 comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel  
823 (eds.), *Parallel Problem Solving from Nature — PPSN V*, pp. 292–301, Berlin, Heidelberg,  
824 1998. Springer. ISBN 978-3-540-49672-4. doi: 10.1007/BFb0056872.

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

## 864 A REPRODUCIBILITY CHECKLIST

865 To ensure reproducibility of our work, we adhere to the checklist in use by AutoML-Conf.

- 866
- 867
- 868 1. For all authors...
- 869 (a) Do the main claims made in the abstract and introduction accurately reflect
- 870 the paper’s contributions and scope? [Yes] Yes, in our list of contributions in
- 871 the introduction we reference the corresponding sections for each contribution
- 872 we claim..
- 873 (b) Did you describe the limitations of your work? [Yes] See Section 7.
- 874 (c) Did you discuss any potential negative societal impacts of your work? [N/A]
- 875 (d) Did you read the ethics review guidelines and ensure that your paper conforms
- 876 to them? (see <https://iclr.cc/public/CodeOfEthics>) [Yes]
- 877
- 878 2. If you ran experiments...
- 879 (a) Did you use the same evaluation protocol for all methods being compared (e.g.,
- 880 same benchmarks, data (sub)sets, available resources, etc.)? [Yes] We describe
- 881 our evaluation protocol in Section 5.1.
- 882 (b) Did you specify all the necessary details of your evaluation (e.g., data splits,
- 883 pre-processing, search spaces, hyperparameter tuning details and results, etc.)?
- 884 [Yes] See Appendix E, F, G, and H.
- 885 (c) Did you repeat your experiments (e.g., across multiple random seeds or splits)
- 886 to account for the impact of randomness in your methods or data? [Yes] See
- 887 Section 5.1.
- 888 (d) Did you report the uncertainty of your results (e.g., the standard error across
- 889 random seeds or splits)? [Yes] See Section 5 and Appendix I.
- 890 (e) Did you report the statistical significance of your results? [Yes] See Appendix J.
- 891 (f) Did you use enough repetitions, datasets, and/or benchmarks to support your
- 892 claims? [Yes]
- 893 (g) Did you compare performance over time and describe how you selected the
- 894 maximum runtime? [Yes] We compare across budgets and provide a description
- 895 in Section 5.1.
- 896 (h) Did you include the total amount of compute and the type of resources used
- 897 (e.g., type of GPUS, internal cluster, or cloud provider)? [Yes] See Appendix L.
- 898 (i) Did you run ablation studies to assess the impact of different components of
- 899 your approach? [Yes] See Section 5.6.
- 900
- 901 3. With respect to the code used to obtain your results...
- 902 (a) Did you include the code, data, and instructions needed to reproduce the
- 903 main experimental results, including all dependencies (e.g., `requirements.txt`
- 904 with explicit versions), random seeds, an instructive README with installation
- 905 instructions, and execution commands (either in the supplemental material or
- 906 as a URL)? [Yes] See Appendix K.
- 907 (b) Did you include a minimal example to replicate results on a small subset of the
- 908 experiments or on toy data? [Yes] In the experiment repository provided in
- 909 Appendix K.
- 910 (c) Did you ensure sufficient code quality and documentation so that someone else
- 911 can execute and understand your code? [Yes]
- 912 (d) Did you include the raw results of running your experiments with the given
- 913 code, data, and instructions? [Yes] See Appendix K.
- 914 (e) Did you include the code, additional data, and instructions needed to generate
- 915 the figures and tables in your paper based on the raw results? [Yes] See
- 916 Appendix K.
- 917
4. If you used existing assets (e.g., code, data, models)...
- (a) Did you cite the creators of used assets? [Yes]

- 918 (b) Did you discuss whether and how consent was obtained from people whose data  
 919 you're using/curating if the license requires it? [Yes] See Appendix M.  
 920 (c) Did you discuss whether the data you are using/curating contains personally  
 921 identifiable information or offensive content? [N/A]  
 922 5. If you created/released new assets (e.g., code, data, models)...
- 923 (a) Did you mention the license of the new assets (e.g., as part of your code  
 924 submission)? [Yes] See Appendix M.  
 925 (b) Did you include the new assets either in the supplemental material or as a URL  
 926 (to, e.g., GitHub or Hugging Face)? [Yes] See Appendix K.  
 927 6. If you used crowdsourcing or conducted research with human subjects...
- 928 (a) Did you include the full text of instructions given to participants and screenshots,  
 929 if applicable? [N/A]  
 930 (b) Did you describe any potential participant risks, with links to institutional  
 931 review board (IRB) approvals, if applicable? [N/A]  
 932 (c) Did you include the estimated hourly wage paid to participants and the total  
 933 amount spent on participant compensation? [N/A]  
 934 7. If you included theoretical results...
- 935 (a) Did you state the full set of assumptions of all theoretical results? [N/A]  
 936 (b) Did you include complete proofs of all theoretical results? [N/A]  
 937  
 938  
 939

## 940 B BACKGROUND AND ADDITIONAL RELATED WORK

### 941 B.1 HYPERPARAMETER OPTIMIZATION FOR DEEP LEARNING

942 **Multi-fidelity optimization** The high computational cost of DL model evaluations has  
 943 motivated research in multi-fidelity optimization. Multi-fidelity (MF) (Kandasamy et al.,  
 944 2017) optimizers use *cheap proxies* to approximate promising candidates and speed up the  
 945 search. Bandit-based methods (Jamieson & Talwalkar, 2016; Li et al., 2017) are the most  
 946 popular in the Automated Machine Learning community for multi-fidelity optimization.  
 947 These have been further extended by replacing their Random Search (RS) component with  
 948 evolutionary (Awad et al., 2021) and model-based (Falkner et al., 2018) search, and increasing  
 949 efficiency for large-scale parallelization (Li et al., 2020).  
 950

951 Instead of optimizing the expensive objective function  $f$  as a blackbox, multi-fidelity opti-  
 952 mization leverages evaluations of  $f$  at lower fidelities. For example, when training a Neural  
 953 network with a particular hyperparameter configuration for 100 epochs, a lower-fidelity proxy  
 954 would be the validation score obtained by training the model with the same hyperparameter  
 955 configuration for 15 epochs. More formally, for a hyperparameter configuration  $\lambda \in \Lambda$  at a  
 956 fidelity level  $z \in Z$  where  $Z := \{z_{min}, \dots, z_{max}\}$ ,  $|Z| = m$  is the fidelity space, a cheap proxy  
 957 function of  $f$  is defined as  $\hat{f}(\lambda, z)$ . Therefore, when  $z = z_{max}$  (the maximum fidelity), the  
 958 proxy function  $\hat{f}$  converges to the true objective function  $f$ . Hence,  $f = \hat{f}(\lambda, z_{max})$ .  
 959

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

966 **Prior-based optimization for a single objective** Prior-based single objective optimiza-  
 967 tion can be defined as solving

$$968 \arg \min_{\lambda \in \Lambda} f(\lambda), \quad \text{guided by } \pi(\lambda), \quad (5)$$

969 where prior  $\pi(\lambda)$  is a probability distribution over the location of the optimum of the objective  
 970 function  $f$ .  
 971

PrBO (Souza et al., 2020) 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., 2011) (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. (2011).  $\pi$ BO (Hvarfner et al., 2022) directly augments the acquisition function  $\alpha$  with the unnormalized user-specified prior distribution  $\pi(\lambda)$  which decays over time, controlled by a parameter  $\beta$ :  $\alpha_\pi^n(\lambda) = \alpha(\lambda) \cdot \pi(\lambda)^{\frac{\beta}{n}}$ , where  $n$  refers to the  $n^{\text{th}}$  iteration. Unlike PrBO,  $\pi$ BO generalizes to acquisition functions other than EI and offers convergence guarantees. However, as we saw see in Section 5,  $\pi$ BO’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 4.

Mallik et al. (2023) 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  $p_\mathcal{U}$ ,  $p_\pi$  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  $p_\pi$ . When  $\hat{\lambda}(\cdot)$  becomes active,  $p_\pi$  is split into  $p_\pi$  and  $p_{\hat{\lambda}}$  according to weighted scores  $\mathcal{S}_\pi$  and  $\mathcal{S}_{\hat{\lambda}}$ , 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.

## B.2 MULTI-OBJECTIVE OPTIMIZATION

For many real-world problems we are often interested in optimizing not one, but multiple, potentially competing objectives. MO (Srinivas & Deb, 1994; Deb et al., 2002; Knowles, 2006; Zhang & Li, 2007) 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:

$$\arg \min_{\lambda \in \Lambda} f(\lambda) = \arg \min_{\lambda \in \Lambda} (f_1(\lambda), f_2(\lambda), \dots, f_n(\lambda)) \quad . \quad (6)$$

**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  $\Lambda$ .

Given two candidates  $\lambda_1, \lambda_2 \in \Lambda$ , we say that  $\lambda_2$  dominates  $\lambda_1$  *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  $\lambda_1$  weakly dominates  $\lambda_2$ .

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

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

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

$$\mathcal{P} := \{\lambda \in \Lambda \mid \nexists \lambda' \in \Lambda \text{ with } f(\lambda') < f(\lambda)\} \quad . \quad (7)$$

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:

$$\mathcal{F} = \{f(\lambda) \in \mathbb{R}^n \mid \lambda \in \Lambda, \nexists \lambda' \in \Lambda \text{ with } f(\lambda') < f(\lambda)\} \quad . \quad (8)$$

**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* (Zitzler & Thiele, 1998) 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:

$$\mathcal{H}_r(\mathcal{A}) = \mu(\{x \in \mathbb{R}^n | \exists a \in \mathcal{A} : a \leq x \cap x \leq r\}) \quad , \quad (9)$$

where  $\mu$  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  $\gamma$ , an existing Pareto set  $\mathcal{P}$  and a reference point  $r$ , the *HVI* is formally defined as:

$$HVI(\mathcal{P}, r, \gamma) = \mathcal{H}_r(\mathcal{P} \cup \gamma) - \mathcal{H}_r(\mathcal{P}) \quad . \quad (10)$$

### B.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) (Izquierdo et al., 2021). 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 (Jin & Kacprzyk, 2006) have been adapted mainly from the general MO literature.

**Scalarization-based Bayesian Optimization** Scalarization-based multi-objective Bayesian Optimization (MO-BO) approaches (Knowles, 2006; Golovin & Zhang, 2020; Paria et al., 2019; Yoon et al., 2009) 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 (Knowles, 2006) or the distribution from which the weights are sampled (Yoon et al., 2009; Paria et al., 2019). Knowles (2006) introduced *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 (2005) proposed the Expected Hypervolume Improvement (EHVI) acquisition function wherein a surrogate model is fitted for each objective separately, and then the Expected Improvement (EI) (Jones et al., 1998) of the HV contribution is calculated. Several improvements to calculate EHVI have been proposed, such as in Yang et al. (2019) and Daulton et al. (2020). EHVI is also used in Ozaki et al. (2020) to extend the TPE (Bergstra et al., 2011) to MO TPE. Ponweiser et al. (2008) 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 (Jones, 2001) acquisition function. Izquierdo et al. (2021) 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 (Belakaria et al., 2019) (MESMO) and predictive entropy search (Hernández-Lobato et al., 2016) (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. (2002) proposed the popular Non-dominated Sorting Genetic Algorithm (NSGA-II) which

uses non-dominated sorting (Srinivas & Deb, 1994) to rank candidates from multiple non-dominated fronts and conducts survival selection (tie-breaking) using crowding-distance sort (Deb et al., 2002). S-Metric Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) (Beume et al., 2007) also employs non-dominated sorting from (Srinivas & Deb, 1994) and (Deb et al., 2002) 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 `PrIM0`.

**Multi-objective multi-fidelity optimization** Izquierdo et al. (2021) 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. (2020) adapted HyperBand (HB) to MO using a randomly scalarized objective value (HB+RW) to select and promote promising configurations. Salinas et al. (2021) and Schmucker et al. (2021) further build on Schmucker et al. (2020) 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` ( $\epsilon$ -net) strategy for exploration.

MF-OSEMO (Belakaria et al., 2020b) and iMOCA (Belakaria et al., 2020a) extend the information-theoretic method MESMO to discrete and continuous fidelities, respectively. Irshad et al. (2024) 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 `PrIM0`’s effectiveness in both synthetic MO problems, as well as DL benchmarks.

## C ALGORITHM DETAILS

All the parts of `PrIM0` were implemented based on the NePS (Stoll et al., 2025) package. For `PrIM0`’s initial design strategy we used MOASHA already implemented in NePS. MOASHA in NePS uses the  $\epsilon$ -net MO promotion strategy from the Syne Tune repository, which is the original implementation by its authors (Schmucker et al., 2021). It is also important to note here that MOASHA is the most viable choice for the initial design compared to bandit-based optimizers with synchronous promotions like HB or SH. This is because the latter would require much longer budgets to promote configurations to the highest fidelity rung, which is impractical for the initial design size of BO. We set  $\eta = 3$  and the initial design size to 5. Furthermore, we set  $\epsilon = 0.25$  in our experiments, selecting the prior-weighted acquisition with a probability of 0.75.

In Figure 9, we plot the decay of  $\gamma$ , the exponent of the prior in the weighted acquisition function, in case of `PrIM0` and the original  $\pi$ BO implementation, with increasing number of Bayesian Optimization samples. We show that the  $\gamma$ , and hence the strength of the prior, decays much faster in `PrIM0` in comparison to  $\pi$ BO, which helps to recover from misleading prior information much quicker.

We use the BoTorch implementation of `SingleTaskGP` as a surrogate model for purely continuous or purely discrete spaces, and `MixedSingleTaskGP` for mixed search spaces. 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 (Ament et al.,

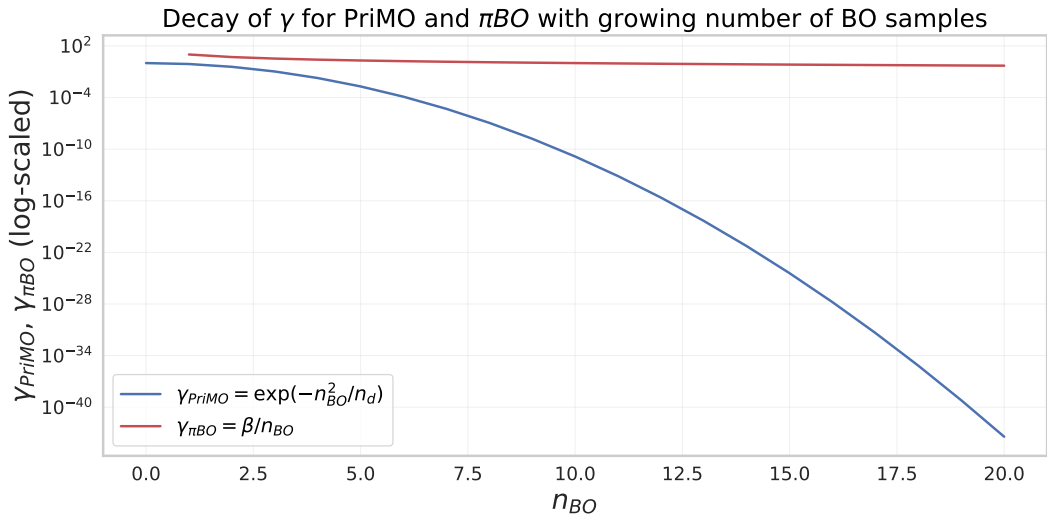


Figure 9: Decay of the prior exponent  $\gamma$  with the increase in the number of Bayesian Optimization samples for PriMO and the original  $\pi$ BO implementation. Here,  $n_d$  refers to the search spaces dimensions, which we fix to 4 - the size of the search space of the PD1 benchmarks. We note that the  $\pi$ BO authors (Hvarfner et al., 2022) set  $\beta = 10$  in the paper.

2023). The NePS package already contains code for the `WeightedAcquisition` function for  $\pi$ BO, which we borrow for PriMO.

## D SOURCES OF PRIORS

In practice, Deep Learning experts often have substantial intuition about ranges of hyperparameter settings that work best for certain models and downstream tasks (Souza et al., 2021; Hvarfner et al., 2022; Mallik et al., 2023). For example, an ideal learning rate for the Adam optimizer (Kingma & Ba, 2015) is often around  $1 \times 10^{-3}$  (Hvarfner et al., 2022). Prior knowledge often stems from the literature, experience in tuning similar models, repeated experimentation or transfer learning across tasks. Consequently, DL researchers often start their experiments with hyperparameter settings that are reported in the original paper for that architecture. In recent literature, large language models have also been used to generate priors for Hyperparameter Optimization (Zhang et al., 2023). Priors for resource-related objectives, such as training time, FLOPS, memory consumption, or energy usage, are generally easier to define, as these objectives are directly observable and can hence be easily measured.

## E CONSTRUCTION OF PRIORS

For the construction of priors, we closely follow the procedure described by Mallik et al. (2023). Our priors are hyperparameter settings, perturbed by a Gaussian noise with a  $\sigma$  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 config-

urations 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.

## F BASELINES

The implementation and hyperparameter setting of all baselines used in this paper are individually detailed below

### F.1 SINGLE-OBJECTIVE BASELINES

**Bayesian Optimization (BO)** Bayesian Optimization is a popular HPO algorithm that builds a probabilistic model to estimate the optimum of a blackbox objective function. We select the Gaussian Processes-based BO implementation from the NePS (Stoll et al., 2025) package, which uses the **q-Log-Noisy Expected Improvement** acquisition function from BoTorch. This has been shown to perform significantly better than ordinary Expected Improvement implementations. The initial design size of the BO is set to be the same as the dimensionality of the corresponding benchmark’s search space (Ament et al., 2023).

**HyperBand (HB)** HyperBand (Li et al., 2017) is a common multi-armed bandit-based HPO algorithm that iterates over multiple Successive Halving brackets, and is a common baseline MF benchmarking studies. The NePS package provides an implementation of HB which allows for continuations, and we set  $\eta = 3$  for all our experiments.

**$\pi$ BO**  $\pi$ BO is a single-objective, blackbox optimization algorithm which augments the acquisition function with user-specified priors. We use the  $\pi$ BO implementation from the NePS package. The original  $\pi$ BO paper (Hvarfner et al., 2022) 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  $\beta$  is set to 10. In the NePS package, however,  $\gamma$  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.

### F.2 MULTI-OBJECTIVE BASELINES FROM THE LITERATURE

**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. Keeping all the settings as described in BO above, we extend the BO implementation in the NePS package by scalarizing the multivariate objective function  $f$  with randomly chosen weights for every seed, at the beginning of the optimization process.

**ParEGO** Just like BO+RW, ParEGO (Knowles, 2006) 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 size of the BO as the package default (search space dimensions).

**Bayesian Optimization with Expected Hypervolume Improvement acquisition function (LogNEHVI)** **The Expected Hypervolume Improvement acquisition function**

was introduced by (Emmerich, 2005) which fits a surrogate model separately for each objective of the multi-objective problem. We employ the BoTorch implementation of Log Noisy Expected Hypervolume Improvement (Ament et al., 2023) from the NePS package. The initial design size for LogNEHVI is set to the size of the search space.

**Multi-objective Tree-structured Parzen Estimator (MO-TPE)** MO-TPE (Ozaki et al., 2020; 2022) is an extension of the popular Tree-structure Parzen Estimator algorithm using EHVI as an acquisition function. We use the Optuna implementation of MO-TPE with recent computational and performance improvements (Abe et al., 2025). Like the other BO algorithms, we set the initial random sampling of Optuna’s MO-TPE to the number of dimensions of the search space.

**NSGA-II** NSGA-II (Deb et al., 2002) 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)** Following Schmucker et al. (2020), we modify HB from NePS with random weights the same way as BO+RW above. For all our experiments, we set the  $\eta = 3$ .

**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 (Schmucker et al., 2021), and that is what we employ for the experiments in this paper. We used MOASHA from the NePS package, and just like HB+RW above, we set  $\eta = 3$ .

### F.3 BASELINES WE CONSTRUCTED TO LEVERAGE MULTI-OBJECTIVE EXPERT PRIORS

**$\pi$ BO with random weights ( $\pi$ BO+RW)**  $\pi$ BO+RW is a MO direct extension of  $\pi$ BO from NePS with random weights, just like BO+RW above. For use with MO priors, we modify  $\pi$ BO to randomly chose and sample from one of the MO priors at each iteration. 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. The remaining details of the base  $\pi$ BO algorithm is the same as detailed above.

**Priorband+BO** Priorband integrates cheap proxies unlike  $\pi$ BO to achieve good anytime performance. It employs an ESP strategy for sampling proportionately from the *priors*, the *incumbent* and at *random*. Priorband+BO is a model-based extension of Priorband using Gaussian Processes with the EI acquisition function. NePS provides an implementation of Priorband+BO which we use for our single-objective experiments. Just like in BO above, Priorband+BO uses the q-Log-Noisy Expected Improvement acquisition function from BoTorch. Further details about this model-based extension is available in the Priorband paper (Mallik et al., 2023).

**MO-Priorband** 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 scalarization-based incumbent modification works better for MO-Priorband than a Pareto front incumbent such as  $\epsilon$ -net. Additionally, we set MO-Priorband’s  $\eta = 3$ , just as in Priorband.

**Random Search + Prior (RS + Prior)** We first randomly select one of the multi-objective priors to sample from. We then equip Random Search to draw random samples from the selected prior distribution (instead of the entire search space) to construct the RS + Prior baseline.

**MOASHA + Prior** We equip MOASHA to sample from one of the multi-objective prior distributions in the same way as Random Search above. For MOASHA + 50% prior sampling, we choose uniformly at random, between prior-based sampling and random sampling from the entire search space.

## G BENCHMARKS

Our main experiments in Section 5 include the 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.

### G.1 PD1 (HYPERBO)

PD1 from HyperBO (Wang et al., 2024) 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, 1983).

From these 24, we select 4 benchmarks from mf-prior-bench (Bergman et al., 2025) 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 *validation error* objective and `train_cost` as the *training 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 & Komodakis, 2016) model on the CIFAR-100 (Krizhevsky, 2009) 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., 2016) on the ImageNet (Russakovsky et al., 2015) 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., 2021) on the *One Billion Word* statistical language modeling benchmark (Chelba et al., 2014). 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., 2022) transformer model on the WMT15 German-English text translation dataset (Bojar et al., 2015). 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	<code>valid_error_rate</code> (max)	<code>train_cost</code> (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

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
<code>lr_decay_factor</code>	float		[0.010093, 0.989012]	continuous	
<code>lr_initial</code>	float	✓	[0.000010, 9.779176]	continuous	
<code>lr_power</code>	float		[0.100708, 1.999376]	continuous	
<code>opt_momentum</code>	float	✓	[0.000059, 0.998993]	continuous	
<code>epoch</code>	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
<code>lr_decay_factor</code>	float		[0.010294, 0.989753]	continuous	
<code>lr_initial</code>	float	✓	[ $1e-5$ , 9.774312]	continuous	
<code>lr_power</code>	float		[0.100225, 1.999326]	continuous	
<code>opt_momentum</code>	float	✓	[ $5.9e-5$ , 0.998993]	continuous	
<code>epoch</code>	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
<code>lr_decay_factor</code>	float		[0.010543, 0.9885653]	continuous	
<code>lr_initial</code>	float	✓	[ $1e-5$ , 9.986256]	continuous	
<code>lr_power</code>	float		[0.100811, 1.999659]	continuous	
<code>opt_momentum</code>	float	✓	[ $5.9e-5$ , 0.9989986]	continuous	
<code>epoch</code>	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
<code>lr_decay_factor</code>	float		[0.0100221257, 0.988565263]	continuous	
<code>lr_initial</code>	float	✓	[ $1.00276e-5$ , 9.8422475735]	continuous	
<code>lr_power</code>	float		[0.1004250993, 1.9985927056]	continuous	
<code>opt_momentum</code>	float	✓	[ $5.86114e-5$ , 0.9989999746]	continuous	
<code>epoch</code>	integer		[1, 19]	discrete	fidelity

## G.2 LCBENCH SURROGATE BENCHMARKS (YAHPO-GYM)

Yahpo-Gym (Pfisterer et al., 2022) 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 fractions. Yahpo-Gym also contains surrogates for the LCBench (Zimmer et al., 2021) set of benchmarks that consists of surrogates trained on the learning curves of DL models, on several OpenML (Vanschoren et al., 2014) 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` as the *validation error* and the *training cost* objectives respectively, 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	<code>val_cross_entropy</code> (max)	<code>time</code> (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
<code>batch_size</code>	integer	✓	[16, 512]	discrete	
<code>learning_rate</code>	float	✓	[1e-4, 0.1]	continuous	
<code>momentum</code>	float		[0.1, 0.99]	continuous	
<code>weight_decay</code>	float		[1e-5, 0.1]	continuous	
<code>num_layers</code>	integer		[1, 5]	discrete	
<code>max_units</code>	integer	✓	[64, 1024]	discrete	
<code>max_dropout</code>	float		[0.0, 1.0]	continuous	
<code>epoch</code>	integer		[1, 52]	discrete	fidelity

### G.3 HARDWARE-AWARE ARCHITECTURE BENCHMARK FOR LANGUAGE MODELS - HW-GPT-BENCH

**HW-GPT-Bench** (Sukthanker et al., 2024) is a hardware-aware language modeling benchmark for multi-objective NAS. It is based on the GPT-2 architecture with up to  $\sim 1.55$  billion parameters. HW-GPT-Bench consists of trained surrogates to predict perplexity and other hardware metrics across multiple devices and architecture scales.

To build the benchmark, the authors train a supernet covering many possible sub-architectures and inherit pretrained weights from the largest GPT-2 model. They use Autogluon as a surrogate model and train it on 100000 randomly sampled architectures from each of the various search space scales to predict the performance and hardware metrics. For the experiments in this paper, we select the **s**, **m** and **l** architecture scales as distinct tasks, and the **perplexity** and **FLOPS** metrics as objectives. For the detailed search spaces, see Table 11, Table 12 and Table 13. The reference points for Hypervolume calculation are listed in Table 14.

## H DETAILS ON EVALUATION PROTOCOL

**Computing hypervolume** We compute the HV with respect to a static reference point set for each benchmark (Appendix G).

**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

1458 Table 11: Hyperparameter search space for HW-GPT-Bench-s benchmark including fidelity  
 1459 `n_layer_choices`. We note here that HW-GPT-Bench itself does not define a fidelity.  
 1460 We select `n_layer_choices` as a practical fidelity for a NAS benchmark, which scales the  
 1461 `mlp_ratio_choices` and `n_head_choices` parameters accordingly.

Hyperparameter	Type	Range	Space Type	Notes
<code>embed_dims</code>	categorical	[192, 384, 768]	discrete	
<code>mlp_ratio_choices</code>	categorical	[2, 3, 4]	discrete	
<code>n_head_choices</code>	categorical	[4, 8, 12]	discrete	
<code>bias_choices</code>	categorical	[ <i>True, False</i> ]	discrete	
<code>n_layer_choices</code>	integer	[10, 11, 12]	discrete	fidelity

1470  
 1471 Table 12: Hyperparameter search space for HW-GPT-Bench-m benchmark including fidelity  
 1472 `n_layer_choices`.

Hyperparameter	Type	Range	Space Type	Notes
<code>embed_dims</code>	categorical	[256, 512, 1024]	discrete	
<code>mlp_ratio_choices</code>	categorical	[2, 3, 4]	discrete	
<code>n_head_choices</code>	categorical	[8, 12, 16]	discrete	
<code>bias_choices</code>	categorical	[ <i>True, False</i> ]	discrete	
<code>n_layer_choices</code>	integer	[22, 23, 24]	discrete	fidelity

1481  
 1482 Table 13: Hyperparameter search space for HW-GPT-Bench-l benchmark including fidelity  
 1483 `n_layer_choices`

Hyperparameter	Type	Range	Space Type	Notes
<code>embed_dims</code>	categorical	[320, 640, 1280]	discrete	
<code>mlp_ratio_choices</code>	categorical	[2, 3, 4]	discrete	
<code>n_head_choices</code>	categorical	[8, 16, 20]	discrete	
<code>bias_choices</code>	categorical	[ <i>True, False</i> ]	discrete	
<code>n_layer_choices</code>	integer	[34, 35, 36]	discrete	fidelity

1493  
 1494 Table 14: Reference values for `val_cross_entropy` and `time` objectives for HVI calculation  
 1495 across the 3 selected HW-GPT-Bench tasks.

Task ID	perplexity (max)	FLOPS (max)
s	100	$9 \times 10^{12}$
m	100	$9 \times 10^{12}$
l	100	$9 \times 10^{12}$

1503  
 1504  
 1505 function evaluation has been performed, *i.e.*, when the benchmark is evaluated at its highest  
 1506 fidelity  $z_{max}$ .

1507  
 1508  
 1509 **Single-objective evaluations** We report the relative rankings of all optimizers over all  
 1510 benchmarks based on the normalized regret per benchmark. Similar to the multi-objective  
 1511 case, we run each optimizer-benchmark pair for 20 equivalent full function evaluations, across  
 25 random seeds.

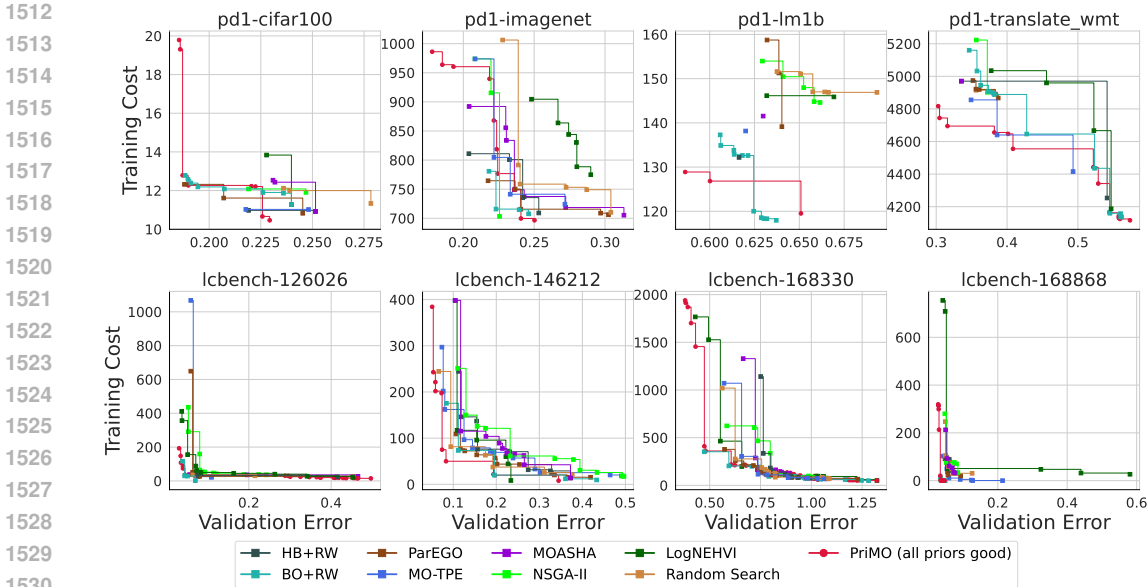


Figure 10: Shown here are the Pareto fronts obtained by PriMO, compared to other non-prior MO baselines under all good prior conditions.

## I ADDITIONAL EXPERIMENTS AND ANALYSIS

In this section, we present detailed Hypervolume and Pareto plots across all 8 benchmarks under good and bad priors, comparing PriMO against both, non-prior and prior-based MO-adapted baselines.

### I.1 PERFORMANCE UNDER GOOD PRIORS

**Pareto Front analysis and comparison against non-prior baselines.** For every baseline, we report the Pareto front aggregated across all seeds per benchmark in line with existing literature (Izquierdo et al., 2021; Schmucker et al., 2020; 2021), with the primary (validation error) objective on the x-axis and the training cost objective along the y-axis. The Pareto front plot in Figure 10 shows that, on average, PriMO and BO+RW locate the most non-dominated points compared to the other optimizers, however, PriMO clearly has the better Pareto Front coverage of the two across most benchmarks.

**Case-level comparison across all benchmarks** In Table 15, we perform a case-level analysis across all benchmarks and compare the performance of PriMO and BO+RW based on the Pareto fronts aggregated across all seeds. We notice that, compared to standard BO+RW, PriMO achieves lower validation error across all benchmarks and lower training cost in most. We further notice that on 5 out of 8 benchmarks, PriMO clearly has both the lower error and model cost, further underlining PriMO’s better coverage of the Pareto front.

**Detailed Hypervolume comparison against prior-based baselines.** In Figure 11, we present more detailed Hypervolume plots across all our 8 benchmarks, comparing PriMO against some prior-based baselines, adapted by us to the MO case. The plots demonstrate that, with the exception of the cifar-100 benchmark, PriMO is one of the two best optimizers across all benchmarks. This highlights PriMO’s ability to effectively utilize good priors despite the  $\epsilon$ -greedy non-prior based component of its BO.  $\pi$ BO+RW is marginally better than PriMO in some benchmarks, due to its much longer dependence on the priors. On the other hand, MO-Priorband seems to be quite ineffective in the utilization of good priors and is the worst performing HPO algorithm across most benchmarks.

Table 15: Case level comparison of performance of PriMO against BO+RW on 8 Deep Learning benchmarks based on aggregated Pareto fronts.

Benchmark	PriMO Best Error	BO+RW Best Error	PriMO Min Cost	BO+RW Min Cost	Superior Algorithm
cifar-100	<b>0.185</b>	0.188	<b>10.46</b>	11.27	PriMO
imagenet	<b>0.178</b>	0.218	<b>696.0</b>	707.0	PriMO
lm1b-transformer	<b>0.586</b>	0.605	119.5	<b>117.9</b>	-
translate-wmt-xformer	<b>0.303</b>	0.347	<b>4120</b>	4140	PriMO
LCBench-126026	<b>0.0307</b>	0.0390	14.97	<b>1.43</b>	-
LCBench-146212	<b>0.0509</b>	0.0841	<b>7.94</b>	9.49	PriMO
LCBench-168330	<b>0.379</b>	0.473	50.77	<b>48.96</b>	-
LCBench-168868	<b>0.0284</b>	0.0353	<b>0.00154</b>	<b>0.00154</b>	PriMO

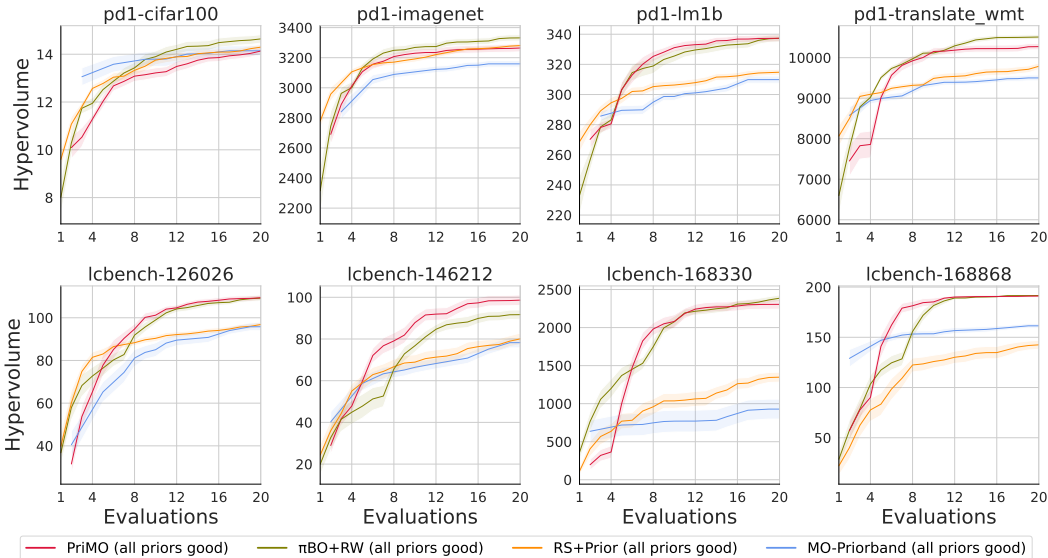


Figure 11: Comparing the average dominated HVI over 20 evaluations and across 25 seeds, between PriMO and prior-based baselines MO-Priorband,  $\pi$ BO+RW and RS + Prior, under all good prior conditions.

## I.2 ROBUSTNESS OF PriMO UNDER BAD PRIOR CONDITIONS

Figure 12 shows PriMO’s remarkable ability to recover from bad priors with respect to the dominated Hypervolume, across all benchmarks. At the end of the optimization budget, PriMO achieves a competitive final performance, very close to BO+RW. Compared to prior-based baselines in Figure 13, PriMO is clearly shown to be the best performing optimizer across all benchmarks, with MO-Priorband - a close second.  $\pi$ BO+RW is unable to recover from bad priors and is the algorithm with the worst final performance on most benchmarks.

We attribute PriMO’s strong recovery under misleading priors to our design of the *decaying MO-prior-weighted acquisition*, influenced by two key parameters –  $\beta$  and  $\epsilon$ . Unlike  $\pi$ BO+RW, which relies on the prior for much longer due to its slow decay schedule, PriMO is explicitly designed to reduce prior influence more aggressively. This design choice allows PriMO to recover more quickly when the priors are misleading, whereas  $\pi$ BO+RW’s prolonged dependence on bad priors significantly hinders its performance, resulting in noticeably worse final performance compared to both PriMO and MO-Priorband. The aggressive  $\beta$  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  $\epsilon$  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

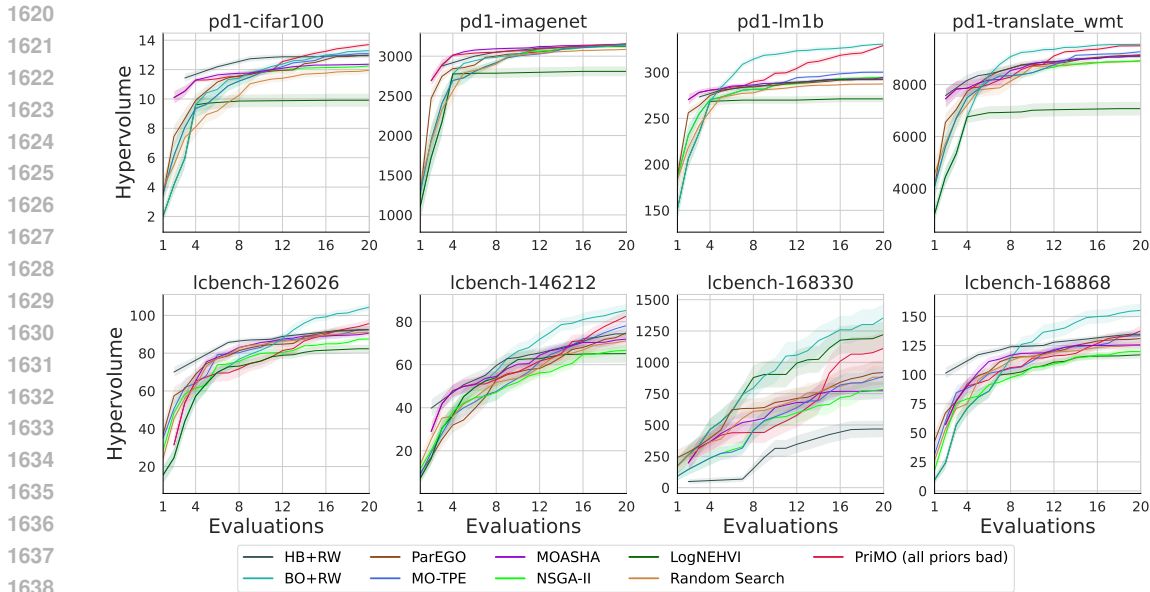


Figure 12: Mean dominated HV across 8 Deep Learning benchmarks, showcasing remarkable recovery of PriMO from bad priors, compared against some prominent non-prior baselines.

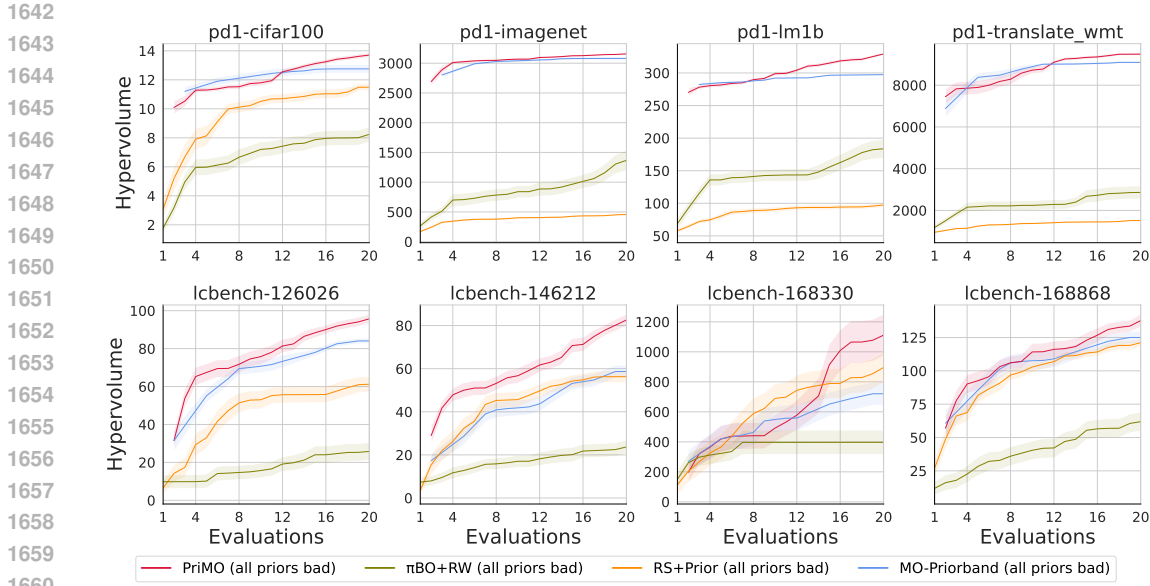


Figure 13: Dominated HV plot comparing PriMO against prior-based baselines that were adapted to MO, under bad priors.

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.

### I.3 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 study this under overall prior conditions with respect to the dominated HV in Figure 14. 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

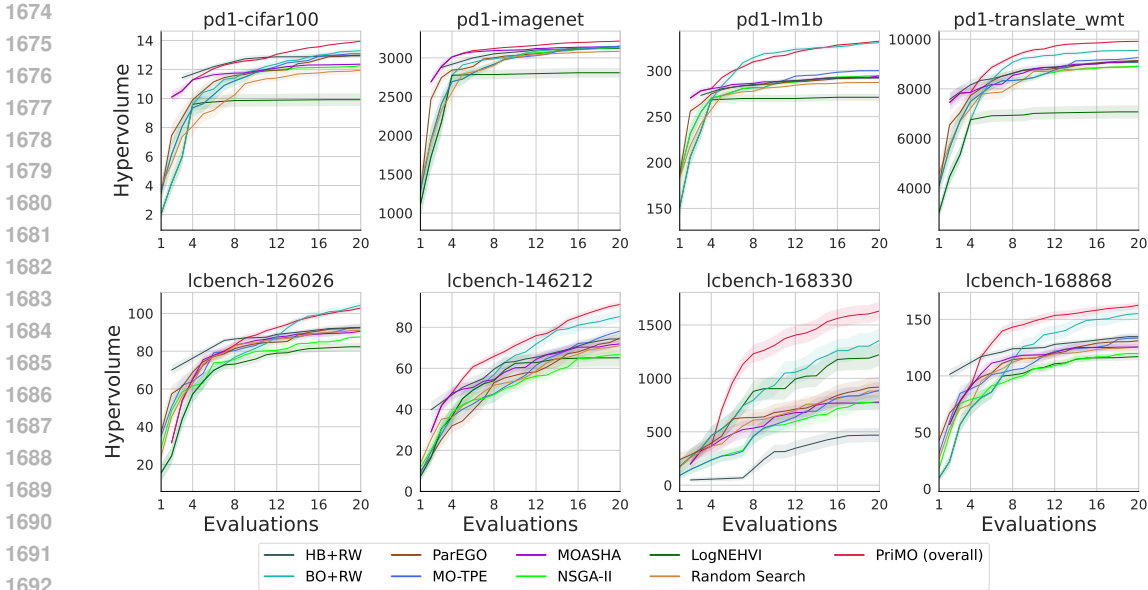


Figure 14: Comparing PriMO against some prominent non-prior MO baselines under the overall prior setting, with respect to the mean dominated HV across all 8 benchmarks.

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 BO-based baselines on most benchmarks, 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.

## 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 Riezler & Hagmann (2022) who proposed LMEM-based significance testing for Natural Language Processing tasks. Further, Geburek et al. (2024) 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:

$$\text{normalized\_hv\_regret} \sim \text{algorithm} \quad , \quad (11)$$

and,

$$\text{normalized\_hv\_regret} \sim (0 + \text{algorithm} \mid \text{seed}) \quad , \quad (12)$$

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

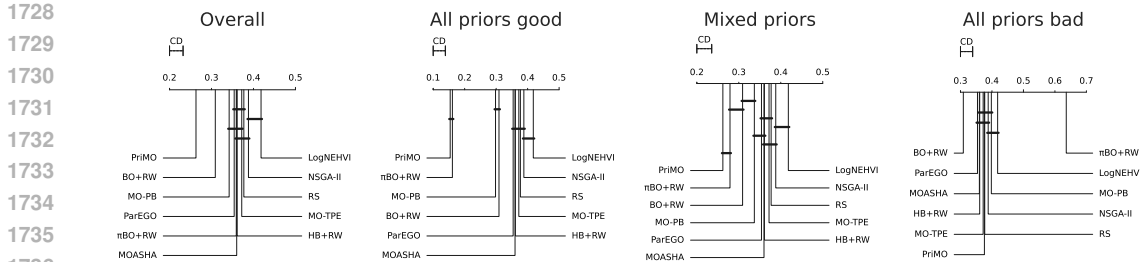


Figure 15: Critical Difference diagrams at **10 evaluations** comparing **early performance** of **PriMO** against the baselines – BO+RW,  $\pi$ BO+RW, MO-Priorband (MO-PB), MOASHA, HB+RW, MO-TPE, LogNEHVI, Random Search (RS), ParEGO and NSGA-II, under various prior conditions.

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

$$\text{normalized\_hv\_regret} \sim 1 \quad , \quad (13)$$

and,

$$\text{normalized\_hv\_regret} \sim \text{algorithm} \quad , \quad (14)$$

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.

## J.2 CRITICAL DIFFERENCE DIAGRAMS

We perform pairwise Tukey HSD (Tukey, 1949) 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 15 shows CD plots for 10 function evaluations, *i.e.*, halfway through our entire allocated budget. Figure 15 (all priors good) shows that PriMO is able to efficiently leverage good priors very early during the optimization, and significantly better than all baselines except  $\pi$ BO+RW. Under all bad priors, PriMO’s performance is not significantly worse than that of most of the best performing non-prior baselines, except BO+RW, but is significantly better than  $\pi$ BO+RW. However, averaging all prior conditions in Figure 15 (overall), 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 16, 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  $\pi$ BO+RW under all good priors, while both are significantly better than other algorithms. Figure 16 (all priors bad) verifies the final performance of PriMO under bad priors, highlighting a strong recovery, where, with the notable exception of BO+RW, PriMO is shown to be significantly better than all baselines. While not significantly better than BO+RW under mixed priors, PriMO is nevertheless the best ranked optimizer. overall, PriMO is clearly shown to be the algorithm with the highest rank, indicating the strongest final performance.

Thus, Figures 15 and 16 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 [this](#) repository. This repository also contains the code used to

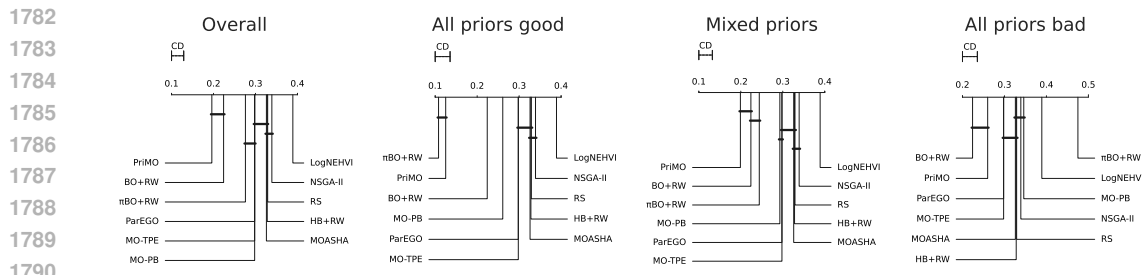


Figure 16: CD diagrams at **20 evaluations** comparing **final performance** of PriMO against other non-prior and prior-guided (adapted to MO) baselines, under various prior conditions.

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, and the raw results from the all optimization runs of PriMO used in this paper are also included.

## 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  $\sim 0.025$  CPU hours, or  $\sim 0.75$  core hours on average. While MF optimizers such as MOASHA and HB+RW completed in just a few seconds ( $\sim 0.15$  core hours), model-based baselines such as BO+RW and  $\pi$ BO+RW required significantly longer on average – typically over 5 minutes ( $\sim 2.5$  core hours).

For the experiments in Section 5, we ran 19 optimizers in total – 12 non-prior and 9 prior-based, including PriMO and all its design ablations, and the optimizers in the single-objective setting. Each prior-based multi-objective optimizer was evaluated under 4 different prior combinations, whereas 2 priors were used for the prior-based single-objective optimizers. Each run lasted 20 evaluations and we evaluated each optimizer on 8 benchmarks across 25 seeds. In total, this amounted to  $\sim 208$  CPU hours, or  $\sim 6240$  core hours to generate the results presented in Section 5.

## M LICENSES

- NePS package: **Apache License, Version 2.0**
- hpoglua: **BSD 3-Clause License**
- mf-prior-bench: **Apache License, Version 2.0**
- Yahoo-Gym : **Apache License, Version 2.0**
- HyperBO PD1: **Apache License, Version 2.0**
- Nevergrad: **MIT License**
- SMAC: **BSD 3-Clause License**
- Syne Tune (code for  $\epsilon$ -net): **Apache License, Version 2.0**
- lmem-significance: **MIT License**