MULTI-OBJECTIVE HYPERPARAMETER OPTIMIZATION WITH EXPERT PRIORS

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 propose the novel problem setting of multi-objective HPO with expert priors and introduce PriMO, the first HPO algorithm that can integrate user beliefs over multiple objectives. We show PriMO achieves state-of-the-art performance across 8 DL benchmarks in the multi-objective and single-objective setting. Further, we demonstrate that PriMO remains state-of-the-art at large scales by optimizing a vision-language model over three objectives.

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 PriMO 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	PriMO
Utilize cheap approximations	✗	✗	✓	✗	✓
Integrate multi-objective expert priors	✗	✗	✗	✗	✓
Strong anytime performance	✗	✗	✓	✗	✓
Strong final performance	✗	(✓)	(✓)	✓	✓

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

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 multi-objective HPO is to find Pareto optimal candidates with strong dominated Hypervolume. 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 study expert priors for multiple objectives (Section 2).
- 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 requirements of multi-objective HPO for DL (Table 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, and show its robustness to different prior conditions (Section 5.2). We also conduct several ablation studies (Section 5.4 and Appendix J) and a hyperparameter sensitivity analysis (Appendix J).
- We validate PriMO's applicability to large models by optimizing the vision-language model OpenCLIP (Radford et al., 2021) over three objectives. Our case study shows that PriMO maintains its superiority even when navigating competing objectives in a compute-intensive setting (Section 5.3).

2 Multi-objective HPO with expert priors and cheap approximations

To capture all the above desiderata, we propose the novel problem formulation of optimizing a vector-valued objective function f, while exploiting cheap approximations of its individual objectives and expert priors. For background on HPO for DL, the utilization of cheap proxies, and the multi-objective case see Appendix B, and for a comparison to related problem formulations see Section 3.

Pareto optimality Typically, there does not exist a single best solution for multi-objective problems that minimizes all the objectives simultaneously. Rather, the goal is to find a strong set of solutions.

Given two candidates $\lambda_1, \lambda_2 \in \Lambda$, we say that λ_2 dominates λ_1 *if and only if* $f(\lambda_2) < f(\lambda_1)$. A candidate λ that is not dominated by any other candidate λ' is called *Pareto Optimal*, and the set of Pareto Optimal candidates is known as the *Pareto Set* $\mathcal{P}$, defined as

$$\mathcal{P} := \{\lambda \in \Lambda \mid \nexists \lambda' \in \Lambda \text{ with } f(\lambda') < f(\lambda)\} \quad . \tag{1}$$

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

$$\pi_{f_i}(\lambda) = \mathbb{P}\left(f_i(\lambda) = \min_{\lambda' \in \Lambda} f_i(\lambda')\right) , \tag{2}$$

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 6 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 λ are evaluated at the fidelity level z, where $f_i(\lambda) = \hat{f}_i(\lambda, z_{max})$. Therefore, our goal is to solve for $\mathcal{P}$ guided by $\Pi_f(\lambda)$ using inexpensive evaluations of f, while addressing the challenge that the priors may be misleading.

3 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). πBO (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 5.4).

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 $\gamma = exp\left(-n_{\text{BO}}^2/n_d\right)$, where n_{BO} and n_d refer to the number of BO samples and the number of dimensions in the search space, respectively. Unlike πBO, where $\gamma = \frac{10}{n}$, with n referring to the n$^{\text{th}}$ iteration, we reduce the overdependence on the prior by setting γ 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 random weights, sampled at every optimization

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

Algorithm	Sampling time [s]	Model evaluation time [s]
PriMO	10 $\pm$1	8307 $\pm$105
BO+RW	10 $\pm$1	8741 $\pm$215

iteration, which is not only simple but also scalable with the number of objectives n:

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

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

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

We use the `qLogNoisyExpectedImprovement` acquisition function (Ament et al., 2023), which we find to be the best overall compared to other prominent acquisition functions (see Appendix J). Furthermore, we set $\epsilon = 0.25$ in our experiments, selecting the prior-weighted acquisition with a probability of 0.75, and perform a sensitivity study on ϵ in Appendix K.

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. See Algorithm 4 and 5 for more details on MOASHA.

4.3 The single-objective setting

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 ϵ-greedy prior-augmented Bayesian Optimization phase remains unchanged.

4.4 Discussion

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 ϵ-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). We further note that the runtime of PriMO's initial design scales linearly with the number of objectives and quadratically with increasing number of candidate configurations in the higher rungs. However, that is still negligible compared to the evaluation cost of a DL model.

Algorithm 1 Initial design strategy

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

Algorithm 2 BO step with multi-obj. priors

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

Algorithm 3 PriMO

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

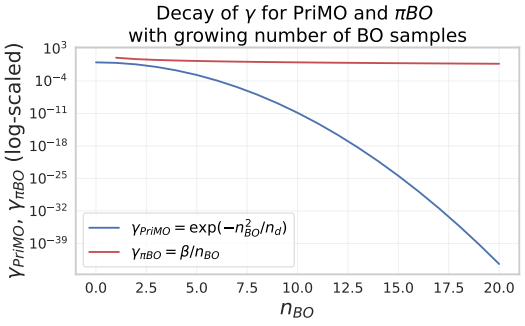


Figure 1: Decay of the prior exponent γ with the increase in the number of BO samples for PriMO and the original πBO implementation. Here, n_d refers to the search spaces dimensions, which we fix to 4 - the size of the search space of the `PD1` benchmarks. We note that the πBO authors set $\beta = 10$.

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.

γ decay comparison In Figure 1, we plot the decay of γ, the exponent of the prior in the weighted acquisition function, in case of PriMO and the original πBO implementation, with increasing number of Bayesian Optimization samples. We show that the γ, and hence the strength of the prior, decays much faster in PriMO in comparison to πBO, which helps to recover from misleading prior information much quicker.

5 EXPERIMENTS

To empirically demonstrate that PriMO fulfills the desiderata outlined in the Section 1, we address the following research questions.

- **RQ1**: Does PriMO achieve state-of-the-art performance for multi-objective optimization?
- **RQ2**: Does PriMO maintain state-of-the-art performance in the single-objective setting?
- **RQ3**: Can PriMO effectively leverage multi-objective expert priors, as well as recover from misleading priors, to maintain its robustness?
- **RQ4**: Does PriMO maintain its superior performance when scaling to a compute-intensive application with three objectives?
- **RQ5**: How does the contribution of the proposed ϵ-BO compare to that of the initial design?
- **RQ6**: Are the choices of scalarization and acquisition function for PriMO justified?

After providing details on our experimental setup and baselines (Section 5.1), we first show PriMO's state-of-the-art performance in the multi-objective and single-objective setting across our 8 Deep Learning benchmarks in Section 5.2 (answering **RQ1**, **RQ2** and the first part of **RQ3**). We then discuss its robustness across all prior conditions (answering the second part of **RQ3**), and demonstrate how PriMO scales to real world applications and problems with more than 2 objectives in Section 5.3 (**RQ4**). Finally, we provide ablation studies in Section 5.4 (answering **RQ5** and **RQ6**). In Appendix L we provide detailed case-level analysis based on Pareto fronts, and hypervolume plots under all 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 I, provide additional experiments and analysis in Appendix L, and conduct a statistical significance analysis in Appendix M.

Benchmarks For our multi-objective experiments, 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. For the single-objective setting, we stick to the benchmarks used by Mallik et al. (2023). Note that for our main benchmark suites - PD1 and Yahpo-Gym , we have 4 and 7 dimensional search spaces respectively, with 2 objectives for each, which are the most common settings for practical DL.

Baselines We give an overview of all our baselines here and provide additional details in Appendix F. We compare PriMO against a host of prominent MO baselines representing different classes of optimization algorithms for MO, which include model-based MOMF (MOMF-BO) (Irshad et al., 2024), scalarized Bayesian Optimization approaches like BO with random weights (BO+RW) and ParEGO (Knowles, 2006), BO with Expected Hypervolume Improvement acquisition function (qLogNEHVI) (Ament et al., 2023), 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). While no multi-objective in the literature leverage expert priors, we augment single-objective approaches to provide strong prior-based baselines and refer to Appendix L.1 for a comparison. For our

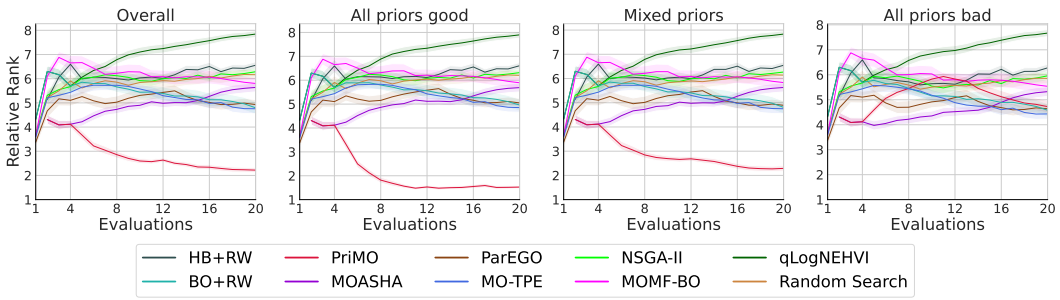


Figure 2: Mean relative ranks ± 1 standard error of PriMO and prominent multi-objective algorithms across benchmarks and seeds under various prior conditions.

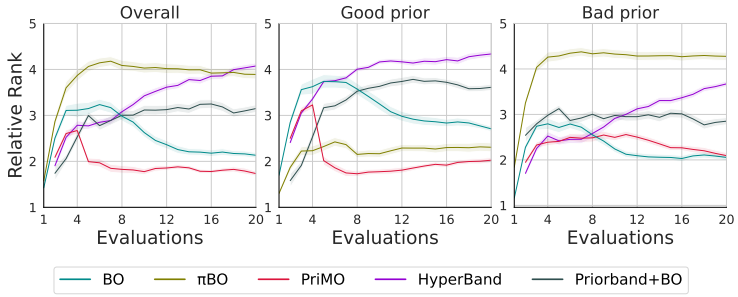


Figure 3: Mean relative ranks ± 1 standard error in the single-objective setting across benchmarks and seeds under various prior conditions.

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 (Mallik et al., 2023) and πBO (Hvarfner et al., 2022).

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 adapt the protocols in the literature on single-objective optimization (Hvarfner et al., 2022; Mallik et al., 2023), for details see Appendix E.

5.2 PriMO's state-of-the-art performance and robustness to prior conditions

State-of-the art in the multi-objective setting Figure 2 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 between PriMO and the second best optimizer, MO-TPE, is even more pronounced. PriMO shows a strong start under good priors, and is the best performing algorithm 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**). Designed with the practical DL use case in mind, where most practitioners operate on modest budgets, PriMO reduces its dependence on priors after approximately 10 BO samples, governed by our chosen γ setting (see Section 4).

State-of-the art in the single-objective setting Figure 3 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 πBO 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.

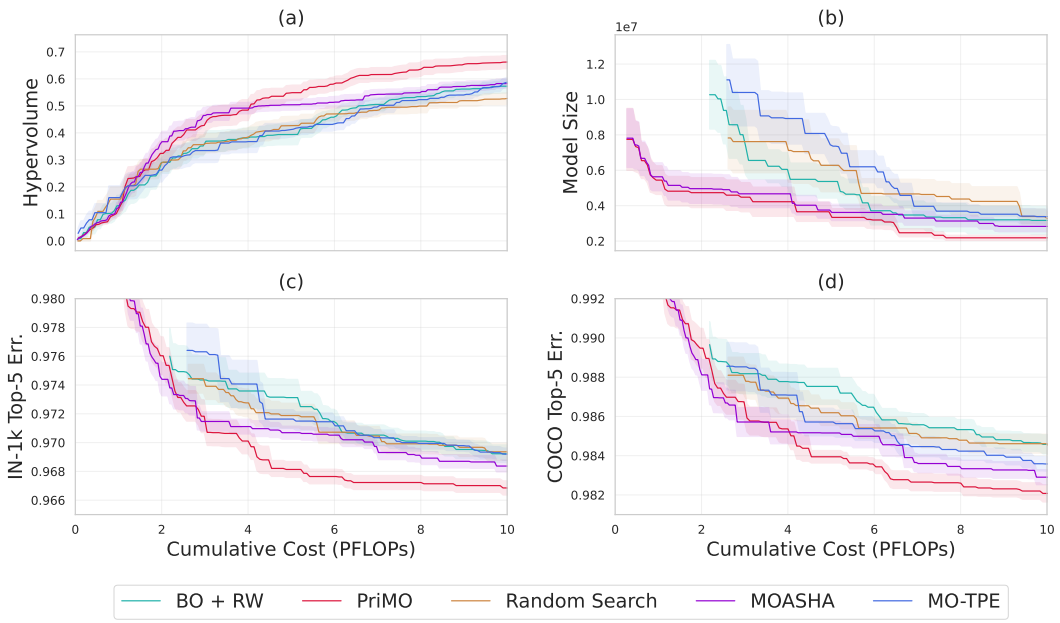


Figure 4: Case study on the OpenCLIP pipeline across 3 objectives over 20 seeds. [Panel (a)] Mean dominated hypervolume vs. cumulative compute cost (in PFLOPs). [Panels (b), (c), and (d)] The trajectories of the best-found (incumbent) values for model parameter size, ImageNet1k error, and MS-COCO error. PriMO outperforms all other baselines, achieving both the highest hypervolume and the best individual values for each objective within the 10 PFLOP budget.

PriMO is robust to bad priors Even when the priors for all objectives are bad (Figure 2, right) PriMO shows a remarkable recovery after a poor initial performance and by the end of the optimization budget, nearly matches MO-TPE, ParEGO and BO+RW (with no statistically significant difference in performance - Appendix M), resulting in a competitive final performance (**RQ3**). 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.

5.3 Large-scale case study

Beyond surrogate benchmarks, we evaluate PriMO on a real-world pipeline demanding a substantial compute budget: pretraining OpenCLIP across 3 objectives comprising model size and the error rates (1 - Acc@5) on two downstream tasks (image classification on ImageNet1k and a retrieval task on MS-COCO). In this case study, we do not study different prior strengths in a controlled setting, but formulate expert priors based on prior experience with the OpenCLIP pipeline. Due to the computational costs involved, we compare PriMO only against the strongest baselines of the previous experiments (MOASHA, BO+RW, and MO-TPE), and RS. As shown in Figure 4, given a cumulative budget of 10 PFLOPs and aggregating over 20 seeds, PriMO achieves a significantly higher dominated hypervolume and superior convergence across the objective space (**RQ4**). Further details on the case study can be found in Appendix H.

5.4 The significant contribution of the ϵ-BO over the initial design strategy

We consider different design ablations of PriMO in Figure 5 to answer **RQ5** and, although overall, find that all components of PriMO are important, the contribution of the ϵ-BO is rather significant. We divide PriMO into its constituent components, namely - the initial design, MO-Priors and the ϵ-BO (including the random weights), and label each design ablation with respect to the component(s) that were removed. We note that PriMO without Priors and PriMO without ϵ-BO correspond to vanilla BO and πBO+RW respectively, both augmented with our initial design strategy.

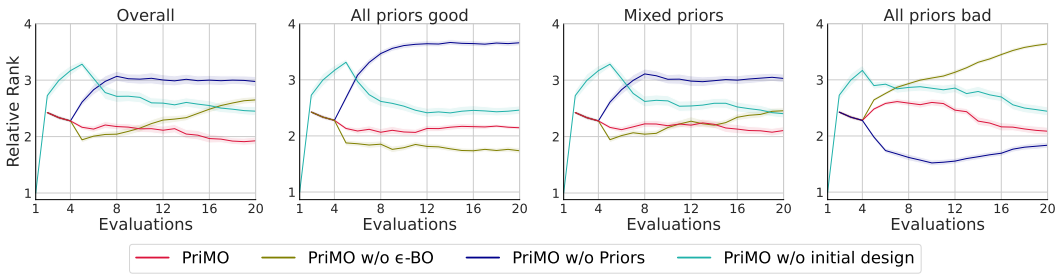


Figure 5: Mean relative ranks $\pm$ 1 standard error of various ablations of PriMO across benchmarks and seeds under various prior conditions.

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 ϵ-BO.

PriMO without Priors and PriMO without ϵ-BO are two of the worst performing ablations overall, highlighting the importance of a prior-based BO design, coupled with the ϵ-greedy optimization strategy in ϵ-BO. These findings, taken together, support our final design choice for PriMO (**RQ5**).

In Appendix J, we perform another ablation study on the scalarization method and acquisition function used in PriMO, to answer **RQ6**.

6 Limitations and Discussion

Despite its impressive empirical performance and our extensive ablation study and sensitivity analysis to PriMO's hyperparameters, we do not provide theoretical guarantees for PriMO. We do experiment with Hypervolume scalarization (Golovin & Zhang, 2020) which has provable guarantees to converge to non-convex Pareto fronts, however, we see that PriMO with the default Linear scalarization outperforms PriMO with Hypervolume Scalarization (Appendix J) and opt for strong empirical performance over theoretical guarantees.

In line with previous work (Souza et al., 2020; Hvarfner et al., 2022; Mallik et al., 2023), we 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 to formulate priors based on an approximate Pareto front, PriMO already achieves state-of-the-art performance using factorized priors.

We chose our benchmarks to represent practically relevant DL settings; naturally, most of our benchmarks have low-dimensional hyperparameter spaces and two objectives that correspond to some performance metric and a compute cost metric. While benchmarks with more than three objectives are known to have highly correlating objectives (Li et al., 2021; Bansal et al., 2022; Dou et al., 2023; Sukthanker et al., 2024), we analyze PriMO's runtime for more than two objectives and investigate its empirical performance for three objectives in our case study.

7 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.

AI USE STATEMENT

In this work, we have only used generative AI tools to aid with the visualization of the results, and not for any other tasks including the code or writing of the manuscript. The new algorithm proposed in this paper is an original contribution of the authors and we take responsibility for the full final content of this work.

REPRODUCIBILITY STATEMENT

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

REFERENCES

K. Abe, Y. Wang, and S. Watanabe. Tree-structured parzen estimator can solve black-box combinatorial optimization more efficiently, 2025. URL https://arxiv.org/abs/2507.08053.

S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, December 2023.

Peter Auer. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002. ISSN ISSN 1533-7928. URL https://www.jmlr.org/papers/v3/auer02a.html.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256, May 2002. ISSN 1573-0565. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

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

A. Bansal, D. Stoll, M. Janowski, A. Zela, and F. Hutter. JAHS-bench-201: A foundation for research on joint architecture and hyperparameter search. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, 2022.

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

S. Belakaria, A. Deshwal, and J. R. Doppa. Information-theoretic multi-objective bayesian optimization with continuous approximations, 2020a. URL https://arxiv.org/abs/2009.05700.

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

E. Bergman, N. Mallik, C. Hvarfner, and S. Basu. mf-prior-bench, March 2025. URL https://github.com/automl/mf-prior-bench.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (eds.), *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'11)*, pp. 2546–2554. Curran Associates, 2011.

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

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

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In Larochelle et al. (2020), pp. 1877–1901.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. URL https://arxiv.org/abs/1312.3005.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2829. IEEE, June 2023. doi: 10.1109/cvpr52729.2023.00276. URL http://dx.doi.org/10.1109/CVPR52729.2023.00276.

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

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

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

Shuguang Dou, XINYANG JIANG, Cai Rong Zhao, and Dongsheng Li. EA-HAS-bench: Energy-aware hyperparameter and architecture search benchmark. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=n-bvaLSCC78.

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

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

L. Franceschi, M. Donini, V. Perrone, A. Klein, C. Archambeau, M. Seeger, M. Pontil, and P. Frasconi. Hyperparameter Optimization in Machine Learning, April 2025. URL http://arxiv.org/abs/2410.22854. arXiv:2410.22854 [stat].

A. M. Geburek, N. Mallik, D. Stoll, X. Bouthillier, and F. Hutter. LMEMs for post-hoc analysis of HPO benchmarking. In *AutoML Conference 2024 (Workshop Track)*, 2024. URL https://openreview.net/forum?id=AAASG6BNvl.

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

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

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

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

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

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

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

Y. Jin and J. Kacprzyk (eds.). *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, Berlin, Heidelberg, 2006. ISBN 978-3-540-30676-4 978-3-540-33019-6. doi: 10.1007/3-540-33019-4. URL http://link.springer.com/10.1007/3-540-33019-4.

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

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

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

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

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations (ICLR'15)*. ICLR, 2015. Published online: iclr.cc.

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

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

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

B. Lefaudeux, F. Massa, D. Liskovich, W. Xiong, V. Caggiano, S. Naren, M. Xu, J. Hu, M. Tintore, S. Zhang, P. Labatut, D. Haziza, L. Wehrstedt, J. Reizenstein, and G. Sizov. xFormers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. {HW}-{nas}-bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_0kaDkv3dVf.

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

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

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

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

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

Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi. Multiobjective Tree-Structured Parzen Estimator. *Journal of Artificial Intelligence Research*, 73:1209–1250, April 2022. ISSN 1076-9757. doi: 10.1613/jair.1.13188. URL https://www.jair.org/index.php/jair/article/view/13188.

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

F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, and B. Bischl. YAHPO Gym – An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization, July 2022. URL http://arxiv.org/abs/2109.03670. arXiv:2109.03670 [cs].

W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted $\mathcal{S}$-Metric Selection. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni (eds.), *Parallel Problem Solving from Nature – PPSN X*, pp. 784–794, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-87700-4. doi: 10.1007/978-3-540-87700-4_78.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv:2103.00020v1 [cs.CV]*, 2021.

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

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

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

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

D. Salinas, V. Perrone, O. Cruchant, and C. Archambeau. A multi-objective perspective on jointly tuning hardware and hyperparameters, June 2021. URL http://arxiv.org/abs/2106.05680. ICLR Workshop on Neural Architecture Search.

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

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

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

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs, 2021. URL https://arxiv.org/abs/2111.02114.

A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Prior-guided Bayesian optimization. In Larochelle et al. (2020).

A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. Bayesian optimization with a prior for the optimum. In N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano (eds.), *Machine Learning and Knowledge Discovery in Databases. Research Track*, volume 12975 of *Lecture Notes in Artificial Intelligence*, pp. 265–296. Springer-Verlag, 2021.

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

D. Stoll, N. Mallik, E. Bergman, S. Schrodi, S. Basu, A. M. Geburek, T. Abou Chakra, S. Garibov, G. Gaur, N. Alipour, M. Janowski, C. Hvarfner, T. Carstensen, E. Mekic, D. Rogalla, R. Binxin, and F. Hutter. Neural Pipeline Search (NePS), April 2025. URL https://github.com/automl/neps.

Rhea Sanjay Sukthanker, Arber Zela, Benedikt Staffler, Aaron Klein, Lennart Purucker, Jörg K. H. Franke, and Frank Hutter. Hw-gpt-bench: Hardware-aware architecture benchmark for language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 60776–60834. Curran Associates, Inc., 2024. doi: 10.52202/079017-1944.

W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

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

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

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

Z. Wang, G. E. Dahl, K. Swersky, C. Lee, Z. Nado, J. Gilmer, J. Snoek, and Z. Ghahramani. Pre-trained Gaussian Processes for Bayesian Optimization. *Journal of Machine Learning Research*, 25 (212):1–83, 2024. URL http://jmlr.org/papers/v25/23-0269.html.

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

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

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

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

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

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

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

A Reproducibility Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Yes, in our list of contributions in the introduction we reference the corresponding sections for each contribution we claim..
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? (see https://iclr.cc/public/CodeOfEthics) [Yes]
2. If you ran experiments...
   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources, etc.)? [Yes] We describe our evaluation protocol in Section 5.1.
   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, preprocessing, search spaces, hyperparameter tuning details and results, etc.)? [Yes] See Appendix E, F, G, and I.
   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] See Section 5.1.
   (d) Did you report the uncertainty of your results (e.g., the standard error across random seeds or splits)? [Yes] See Section 5 and Appendix L.
   (e) Did you report the statistical significance of your results? [Yes] See Appendix M.
   (f) Did you use enough repetitions, datasets, and/or benchmarks to support your claims? [Yes]
   (g) Did you compare performance over time and describe how you selected the maximum runtime? [Yes] We compare across budgets and provide a description in Section 5.1.
   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix O.
   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] See Section 5.4.
3. With respect to the code used to obtain your results...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all dependencies (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation instructions, and execution commands (either in the supplemental material or as a URL)? [Yes] See Appendix N.
   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] In the experiment repository provided in Appendix N.
   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]
   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] See Appendix N.
   (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] See Appendix N.
4. If you used existing assets (e.g., code, data, models)...
   (a) Did you cite the creators of used assets? [Yes]
   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] See Appendix P.
   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you created/released new assets (e.g., code, data, models)...
   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] See Appendix P.

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

6. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to institutional review board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
7. If you included theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

B Background and Additional Related Work

B.1 Hyperparameter optimization for Deep Learning

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

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

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

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

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

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

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

on the Priors has major downsides. PriMO addresses this issue using a novel MO-priors-based augmentation of the BO component that we introduced in Section 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_π and $p_{\hat{\lambda}}$ respectively. Initially, $\hat{\lambda}(\cdot)$ is inactive. Given the constraint $p_\mathcal{U} + p_\pi = 1$, $\mathcal{E}_\pi$ selects from $\mathcal{U}(\cdot)$ and $\pi(\cdot)$ according to $p_\mathcal{U}$ and p_π. When $\hat{\lambda}(\cdot)$ becomes active, p_π is split into p_π and $p_{\hat{\lambda}}$ according to weighted scores $\mathcal{S}_\pi$ and $\mathcal{S}_{\hat{\boldsymbol{\lambda}}}$, calculated by first computing the likelihood of the top performing configurations under $\pi(\cdot)$ and $\hat{\lambda}(\cdot)$, which capture how much *trust* should be placed on each.

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

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 $f : \lambda \to \mathbb{R}^n$, $\lambda \in \mathbb{R}$. Without loss of generality, we assume minimization of all objectives.

More on Pareto Optimality As mentioned earlier in Section 2, given two candidates $\lambda_1, \lambda_2 \in \Lambda$, we say that λ_2 dominates λ_1 *if and only if* $f(\lambda_2) < f(\lambda_1)$. Formally, we write $\lambda_2 \prec \lambda_1$. Additionally, for $f(\lambda_2) \leq f(\lambda_1)$, we write $\lambda_2 \preceq \lambda_1$ and say that λ_1 weakly dominates λ_2.

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

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

Pareto Front 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:

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

Hypervolume indicator The true Pareto front of a real-world MO problem is generally unknown. Thus, the goal of MO Optimization algorithms is to return a set of non-dominated candidates from which we can obtain an *approximated Pareto front*. To assess the quality of this approximation, the *S-Metric* or *Hypervolume (HV) Indicator* (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:

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

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

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

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. Furthermore, benchmarks with more than three objectives are known to have highly correlating objectives (Li et al., 2021; Bansal et al., 2022; Dou et al., 2023; Sukthanker et al., 2024). 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 PriMO.

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` (ϵ-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 PriMO's effectiveness in both synthetic MO problems, as well as DL benchmarks.

C Algorithm details

All the parts of PriMO were implemented based on the NePS (Stoll et al., 2025) package. For PriMO's initial design strategy we used MOASHA already implemented in NePS. MOASHA in NePS uses the ϵ-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.

We use the BoTorch implementation of `SingleTaskGP` as s 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., 2023). The NePS package already contains code for the `WeightedAcquisition` function for πBO, 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 1×10^{-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 generated 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 Prior construction

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 σ depending on the prior quality. In all our experiments we use two kinds of priors for every objective - *good* and *bad* priors. The good priors represent areas of the hyperparameter space where we expect the corresponding objective to have a value close to its *optimum*. The bad priors represent *inaccurate* configurations which yield poor values for the objective function. The hyperparameter configurations for these priors are generated using the methods listed below:

- Class "**good**" priors: To generate *good* priors, we begin by uniformly sampling 100,000 hyperparameter configurations at random using a fixed global seed for all prior generation runs. We then evaluate these configurations on the corresponding benchmark at the highest

available fidelity, z_{max}. Afterwards, we rank the configurations based on the objective values derived from their evaluations. Since we always aim to minimize each objective, for objectives intended to be maximized, we take their negative values to find the minimum. The configuration that yields the best objective value is perturbed by a Gaussian noise with $\sigma = 0.01$. This slight perturbation reflects a realistic scenario where prior knowledge is good or near-optimal, but never precisely so.

- Class "**bad**" priors: Similar to the good prior case, for the *bad* priors, we sort the configurations based on the corresponding objective value. From this, we select the configuration with the worst seen value and do not perturb it any further. This forms our *bad* prior configuration.

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

πBO πBO is a single-objective, blackbox optimization algorithm which augments the acquisition function with user-specified priors. We use the πBO implementation from the NePS package. The original πBO 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 β is set to 10. In the NePS package, however, γ is completely different and is set to e^{-n_{BO}/n_d}, where n_{BO} refers to the number of BO samples and n_d indicates the dimensions of the search space.

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 at every iteration of optimization.

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 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.

Model-based Multi-objective Multi-fidelity We incorporate a Model-based Multi-objective Multi-fidelity (MOMF-BO) from (Irshad et al., 2024) which uses a *trust-based* cost objective which is directly proportional to the fidelity, optimizing the multi-objective function and the fidelity of the data source jointly. However, it is quite computationally expensive and running a single seed of this MOMF-BO over the translate-wmt-xformer benchmark often took several hours.

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 $\eta = 3$.

Algorithm 4 Multi-objective selector

function `non_dom_sorting`(P)
  $i = 1$
  while $P \neq \emptyset$ **do**
    $F_i \leftarrow \{\vec{\theta} \in P : \vec{\theta}$ non-dominated$\}$
    $(P,\ i) \leftarrow (P \setminus F_i,\ i+1)$
  end while
  return $F_1, \ldots, F_m$
end function

function `selector_eps_net`(P, n)
  $F_1, \ldots, F_m \leftarrow$ `non_dom_sorting`(P)
  $\mathcal{C} \leftarrow [F_1.\text{pop}()]$
  for $F = F_1, \ldots, F_m$ **do**
    while $F \neq \emptyset$ **do**
      $\vec{\theta} \leftarrow \min_{\vec{\theta} \in F} \max_{\vec{\theta}' \in \mathcal{C}} \|\vec{y}_{\vec{\theta}} - \vec{y}_{\vec{\theta}'}\|_2$
      $\mathcal{C}$.append$(\vec{\theta})$
      $F \leftarrow F \setminus \{\vec{\theta}\}$
    end while
  end for
  return $\mathcal{C}_1, \ldots, \mathcal{C}_n$
end function

function `selector_nsga_ii`(P, n)
  $F_1, \ldots, F_m \leftarrow$ `non_dom_sorting`(P)
  $\mathcal{C} \leftarrow [\,]$
  for $F = F_1, \ldots, F_m$ **do**
    $\mathcal{C} \leftarrow \mathcal{C} +$ `crowding_dist_sorting`(F)
  end for
  return $\mathcal{C}_1, \ldots, \mathcal{C}_n$
end function

Algorithm 5 Multi-objective ASHA

Input: R, r_0, s, η (default $\eta = 3$)
function `mo_asha`
  repeat
    for each free worker **do**
      $(\vec{\theta}, k) \leftarrow$ `get_job`()
      `run_evaluation`$(\vec{\theta}, r_0 \eta^{s+k})$
    end for
    for completed job $(\vec{\theta}, k)$ with objective vector $\vec{y}_{\vec{\theta}}$ **do**
      update $\vec{\theta}$ in rung k with $\vec{y}_{\vec{\theta}}$
    end for
  until budget exhausted
end function

function `get_job`
  for $k = \lfloor \log_\eta(R/r_0) \rfloor - s - 1, \ldots, 1, 0$ **do**
    candidates $\leftarrow$ `mo_selector`(rung k, |rung k| $/\eta$)
    promotable $\leftarrow$ $\{\vec{\theta} \in$ candidates : $\vec{\theta}$ not promoted$\}$
    if |promotable| > 0 **then**
      return (promotable[0], $k+1$)
    end if
  end for
  draw random configuration $\vec{\theta}$
  return $(\vec{\theta}, 0)$
end function

Multi-objective Asynchronous Successive Halving (MOASHA) MOASHA (Schmucker et al., 2021) (Algorithm 4, 5) 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, 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

We modify single-objective approaches from the literature (RS + Prior, MOASHA + Prior, πBO (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 πBO with random scalarizations and modify Priorband's ensemble sampling policy using scalarized incumbents for MO to build MO-Priorband.

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

Priorband+BO Priorband integrates cheap proxies unlike πBO to achieve good anytime performance. It employs an ESP strategy for sampling proportionately from the *priors*, the *incumbent* and at *random*. 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 ϵ-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. For an overview of the search spaces of all our benchmarks, see Table 3.

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 bench-

Table 3: Overview of benchmark search spaces.

Benchmark	# Total HPs	# Continuous HPs	# Discrete HPs
cifar-100-wide_resnet-2048	5	4	1
imagenet-resnet-512	5	4	1
lm1b-transformer-2048	5	4	1
translatewmt-xformer-64	5	4	1
yahpo-lcbench 126026	7	4	3
yahpo-lcbench 146212	7	4	3
yahpo-lcbench 168330	7	4	3
yahpo-lcbench 168868	7	4	3

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

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

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

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

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

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

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

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

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

Hyperparameter	Type	Log-scaled	Range	Space Type	Notes
`lr_decay_factor`	float		$[0.010022, 0.988565]$	continuous	
`lr_initial`	float	✓	$[1.00276e{-}5, 9.842248]$	continuous	
`lr_power`	float		$[0.100425, 1.998593]$	continuous	
`opt_momentum`	float	✓	$[5.86114e{-}5, 0.998999]$	continuous	
`epoch`	integer		$[1, 19]$	discrete	fidelity

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

H Details on the large-scale case study

The OpenCLIP pipeline trains CLIP (Radford et al., 2021) models using an open-source implementation (Cherti et al., 2023) on a 1M-sample subset of the LAION-400M dataset (Schuhmann et al.,

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

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

2021). We determined that training on a 1M-sample subset is sufficient to observe meaningful performance differences on downstream tasks while remaining computationally tractable for optimization benchmarking. For our fidelity-based evaluation, we use training epochs (from 1 to 10) as our cheap proxy.

The hyperparameter search space comprises 5 parameters, including the fidelity dimension, which are detailed in Table 11. During the optimization process, several architectural and optimizer parameters are kept constant to isolate the effects of the chosen search space: `text_heads` $= 8$, weight decay (`wd`) $= 1.0 \times 10^{-6}$, Adam optimizer betas (`beta1` $= 0.9$, `beta2` $= 0.999$), and `eps` $= 1.0 \times 10^{-6}$.

We evaluate PriMO against the strongest baselines, comprising MOASHA, BO+RW, MO-TPE, and RS, under a budget of 10 PFLOPs across 20 seeds. In Figure 4, we plot the mean performance, with shaded regions representing the standard error.

Table 11: Hyperparameter search space for the OpenCLIP benchmark. The `epochs` parameter serves as the fidelity for the optimization process.

Hyperparameter	Type	Log-scaled	Range/Set	Space Type	Notes
`lr`	float	✓	$[0.0001, 0.01]$	continuous	
`warmup_fraction`	float		$[0.0, 0.75]$	continuous	
`vision_width`	integer		$\{32, 64, 128, 192, 256\}$	discrete	
`text_width`	integer		$\{32, 64, 128, 192, 256\}$	discrete	
`epochs`	integer		$[1, 10]$	discrete	fidelity

I 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 function evaluation has been performed, *i.e.*, when the benchmark is evaluated at its highest fidelity z_{max}.

We note that for fairness, we generated the plots for MOMF-BO using a fidelity-aware trust threshold of 70% of the respective benchmark's z_{max}, *i.e.*, only evaluations at sufficiently high fidelity were counted as informative progress, while lower-fidelity probes were treated as exploratory. This avoids penalizing MOMF-BO for its sampling strategy, which queries many fidelities before concentrating on the most trusted evaluations.

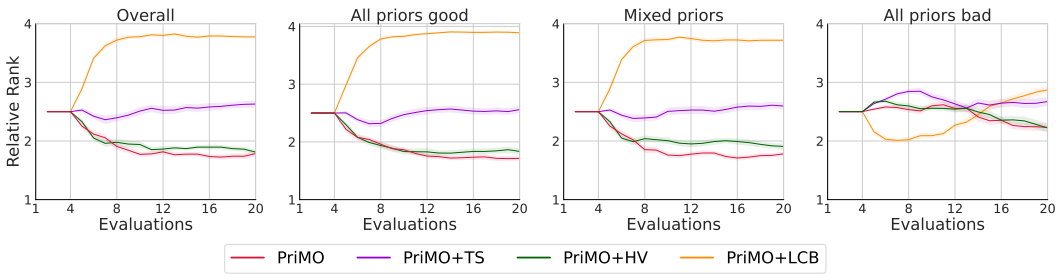


Figure 6: Mean relative ranks ± 1 standard error of various ablations of the ϵ-BO of PriMO across benchmarks and seeds under various prior conditions.

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

J Ablation study of the BO components of PriMO

In this section, we perform an ablation study of the scalarization method and acquisition function of PriMO's ϵ-BO. In particular, we create 3 different versions of PriMO using Hypervolume scalarization (HV) (Golovin & Zhang, 2020), and the Lower Confidence Bound (LCB) (Auer, 2002; Auer et al., 2002) and Thompson Sampling (TS) (Thompson, 1933) acquisition functions, to justify our design choice and answer **RQ6**. As noted in Appendix C, PriMO uses random linear scalarization and the Expected Improvement acquisition function. We note here that we only ablate one component at a time, *i.e.*, ablations using a different acquisition function still use linear scalarization, while the ablation using the Hypervolume scalarization uses the Expected Improvement acquisition function.

Figure 6 shows that after the end of the initial design strategy, the acquisition function has a significant impact on the relative ranking of the ablations, while the effect of the scalarization method is marginal. Overall, the PriMO ablation with the Lower Confidence Bound acquisition function is the worst across all prior conditions, followed by Thompson Sampling. Hypervolume scalarization performs well and is marginally worse than the original design of PriMO under all prior conditions. However, PriMO remains the best design overall and under all prior conditions, thereby justifying our choices for the BO components of PriMO (**RQ6**).

K ϵ sensitivity study

In Figure 7, we perform a sensitivity study on the ϵ in PriMO's BO, across multiple seeds and all benchmarks under good prior conditions. Note that ϵ denotes the probability with which PriMO select vanilla BO *i.e.* without the prior-augmentation in the acquisition function. A high value of ϵ would indicate convergence towards vanilla Bayesian Optimization, while an extremely low value would mean convergence towards πBO+RW. An ϵ=0.5 would essentially mean a 'coin-toss' between prior-based and standard Bayesian Optimization. We choose ϵ=0.25 (in red) which appears to strike the perfect balance between exploiting prior knowledge and exploration without a prior, as well as, aiding in robustness in case of a misleading prior.

L 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.

L.1 Performance under good priors

Hypervolume Plots across all benchmarks under good prior conditions against non-prior baselines Figure 8 shows that PriMO exhibits strong starts across all benchmarks, with respect to

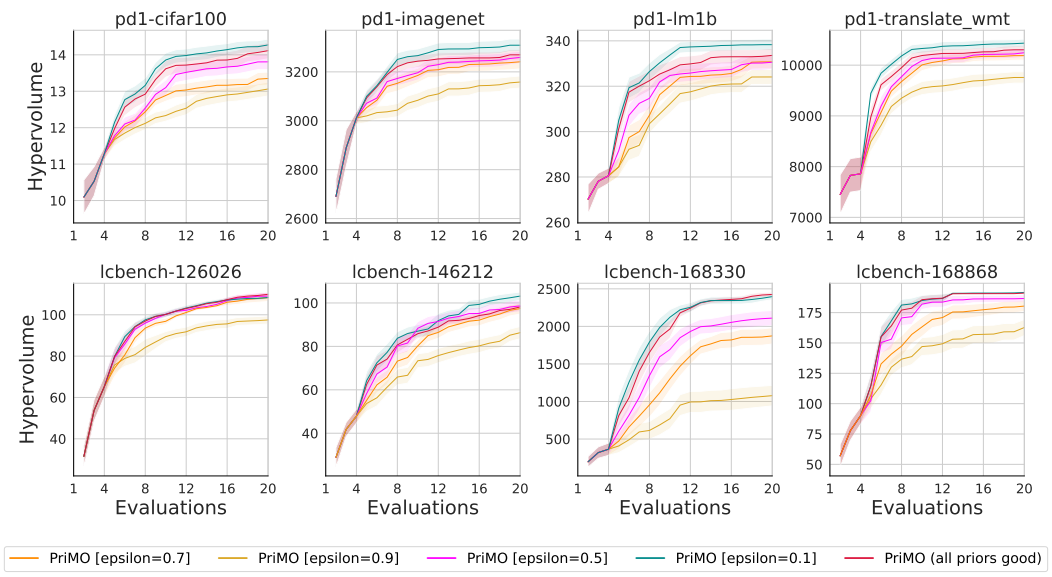


Figure 7: Sensitivity study of ϵ in PriMO's ϵ-BO using mean dominated Hypervolume $\pm$ 1 standard error across seeds for each benchmark. PriMO is under all good priors setting here.

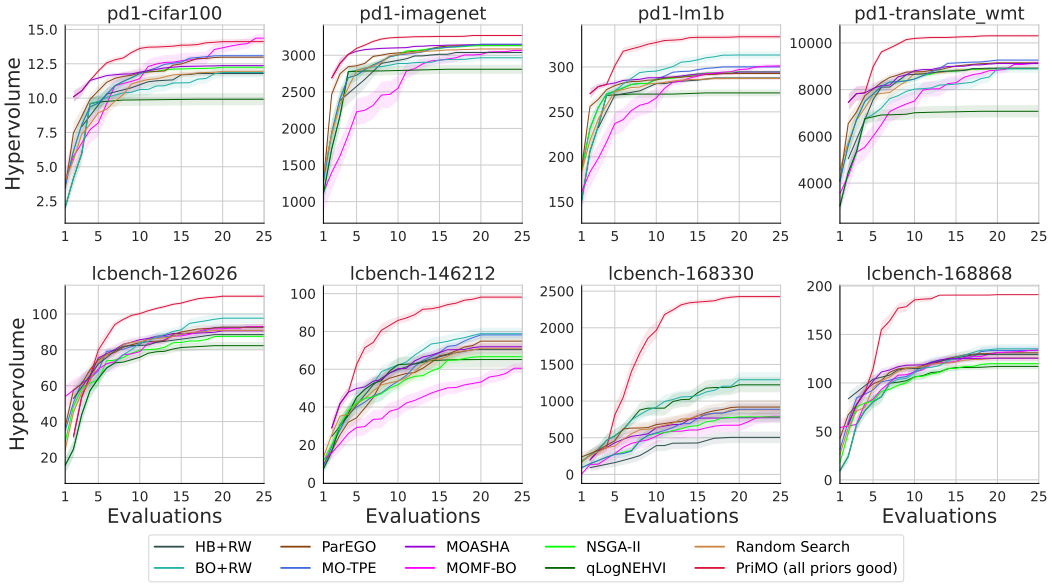


Figure 8: Mean dominated Hypervolume $\pm$ 1 standard error of PriMO and prominent multi-objective algorithms across seeds for each benchmark. PriMO is under all good priors setting here.

the mean dominated Hypervolume, and is the best performing HPO algorithm early on. As explained in Section 5.2, we attribute this head start to the novel initial design strategy which samples good seed points. The ϵ-BOfurther leverages these strong initial points to achieve state-of-the-art performance across all benchmarks, with the exception of cifar-100, where MOMF-BO marginally outperforms PriMO at the end of the budget.

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 9 shows

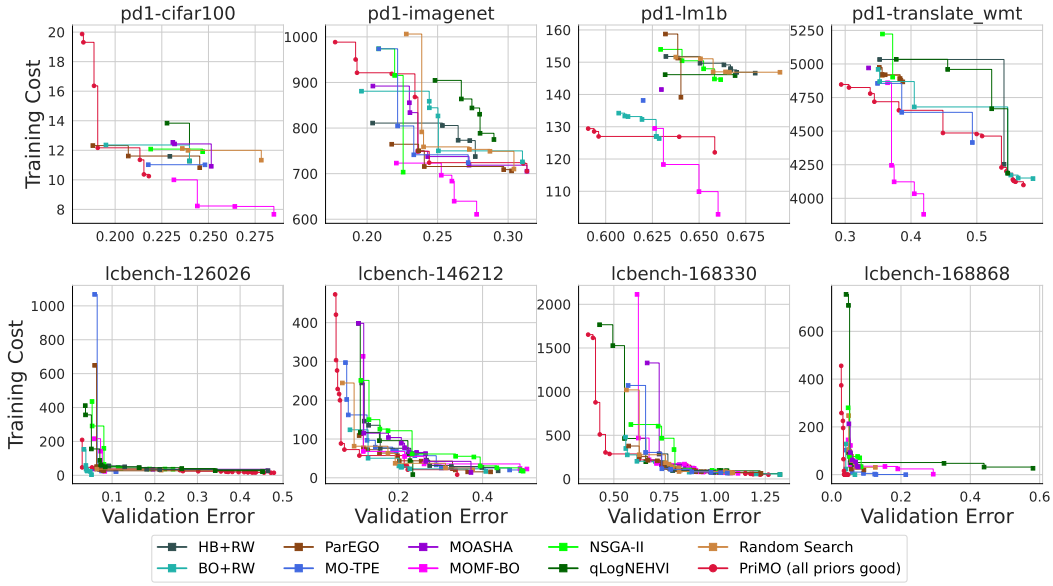


Figure 9: Shown here are the Pareto fronts obtained by PriMO, compared to other non-prior MO baselines under all good prior conditions.

that, on average, PriMO locates the most non-dominated points compared to the baselines on the benchmarks.

Case-level comparison across all benchmarks In Table 12, 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.

Table 12: 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	**0.185**	0.188	**10.46**	11.27	**PriMO**
imagenet	**0.178**	0.218	**696.0**	707.0	**PriMO**
lm1b-transformer	**0.586**	0.605	119.5	**117.9**	-
translate-wmt-xformer	**0.303**	0.347	**4120**	4140	**PriMO**
LCBench-126026	**0.0307**	0.0390	14.97	**1.43**	-
LCBench-146212	**0.0509**	0.0841	**7.94**	9.49	**PriMO**
LCBench-168330	**0.379**	0.473	50.77	**48.96**	-
LCBench-168868	**0.0284**	0.0353	**0.00154**	**0.00154**	**PriMO**

Performance against constructed prior-based MO baselines under good priors In this section, we present the relative rankings and detailed Hypervolume plots across all our 8 benchmarks, comparing PriMO against some prior-based baselines, adapted by us to the MO case. Figure 10 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 πBO+RW are usually 2 of the best optimizers on average. This is expected in terms of the all priors good priors setting since PriMO's BO phase is based on πBO's design.

The plots in Figure 11 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

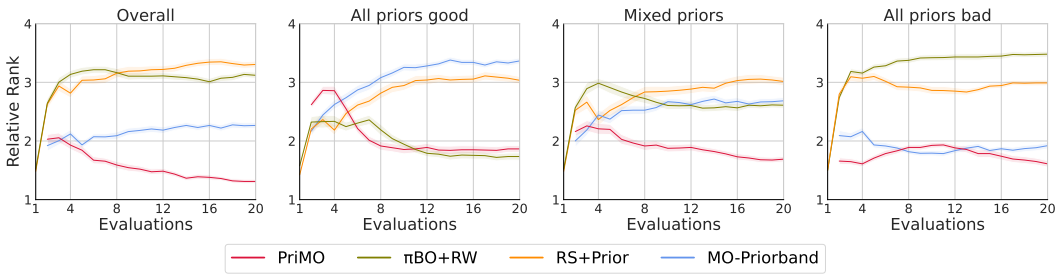


Figure 10: Mean relative ranks $\pm$ 1 standard error of baselines we constructed to use multi-objective priors and PriMO across benchmarks and seeds under various prior conditions.

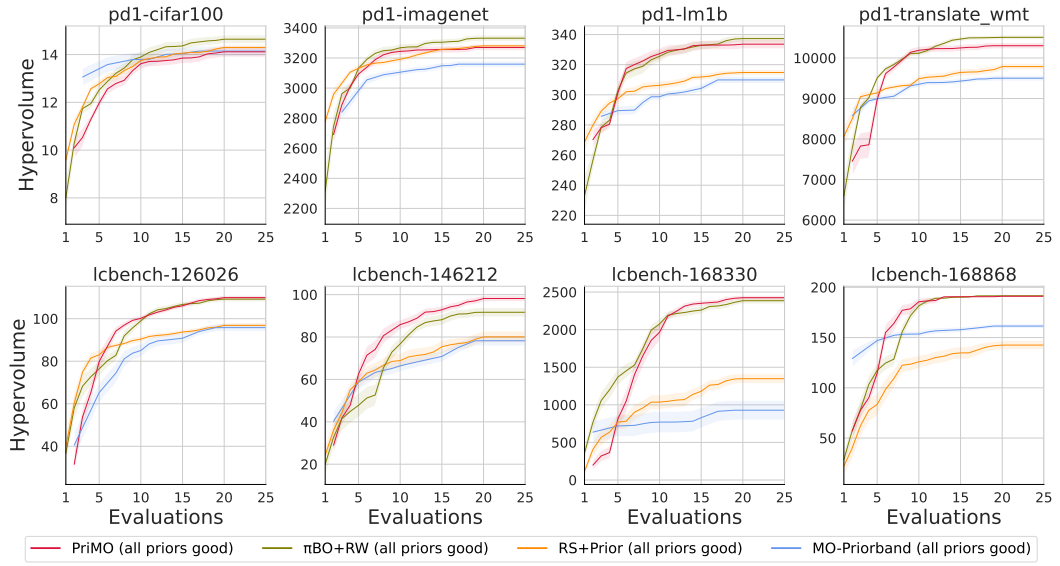


Figure 11: Comparing the average dominated HVI over 20 evaluations and across 25 seeds, between PriMO and prior-based baselines MO-Priorband, πBO+RW and RS + Prior, under all good prior conditions.

utilize good priors despite the ϵ-greedy non-prior based component of its BO. πBO+RWis 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.

L.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 on most benchmarks.

Compared to MO-adapted prior-based baselines in Figure 10 (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, πBO+RW, despite being able to leverage good priors well, is quite prone to misleading priors and, overall, significantly (Appendix M) less robust compared to PriMO. This performance is further detailed in the Hypervolume plots in Figure 13. PriMO is clearly shown to be the best performing optimizer across all benchmarks, with MO-Priorband - a close second. πBO+RW is unable to recover from bad priors and is the algorithm with the worst final performance on most benchmarks.

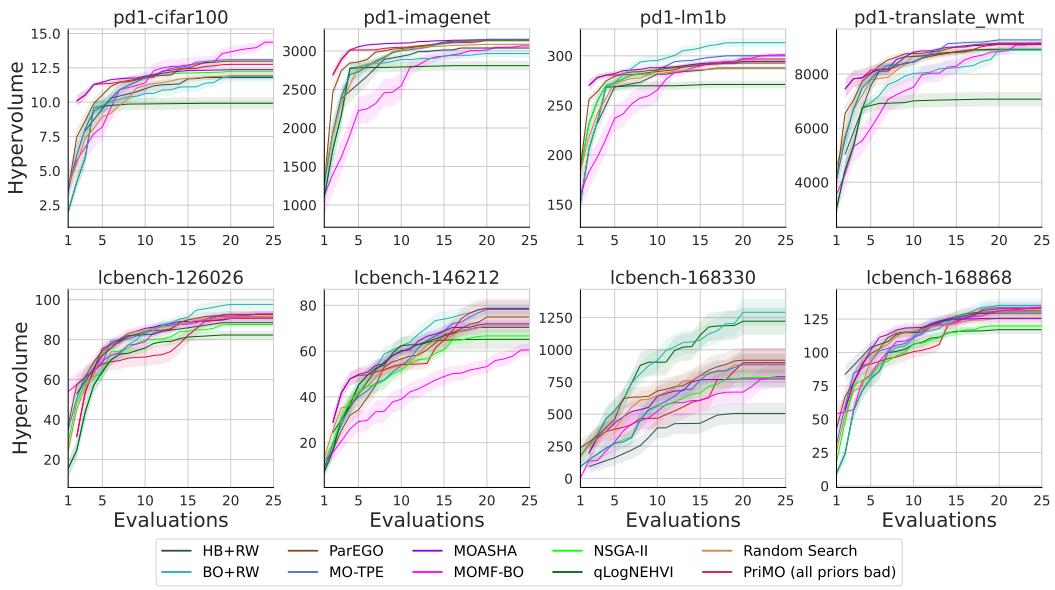


Figure 12: Mean dominated HV across 8 Deep Learning benchmarks, showcasing remarkable recovery of PriMO from bad priors, compared against some prominent non-prior baselines.

We attribute PriMO's strong recovery under misleading priors to our design of the *decaying MO-prior-weighted acquisition*, influenced by two key parameters - β and ϵ. Unlike πBO+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 πBO+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 β setting ensures the prior's influence diminishes rapidly — an important property for practical DL scenarios where HPO is not expected to be run for long. Additionally, the parameter ϵ in the acquisition function controls how much the prior contributes while it is still active, thus encouraging exploration of the search space. Together, these two effects ensure that PriMO does not become overly dependent on the prior and, under inaccurate priors, can still effectively explore and discover better hyperparameter configurations than its counterparts.

L.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 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.

M 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

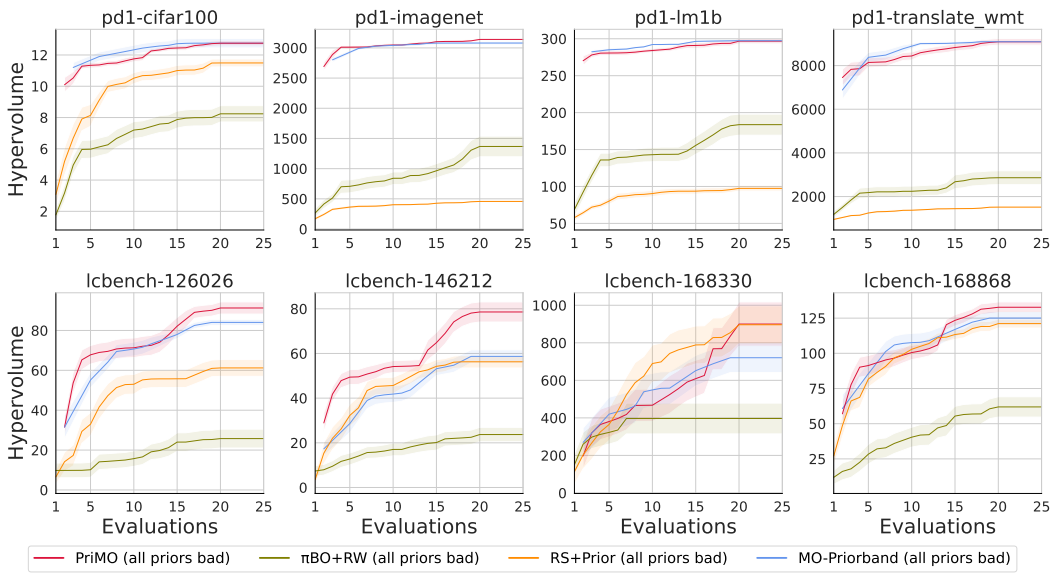


Figure 13: Dominated HV plot comparing PriMO against prior-based baselines that were adapted to MO, under bad priors.

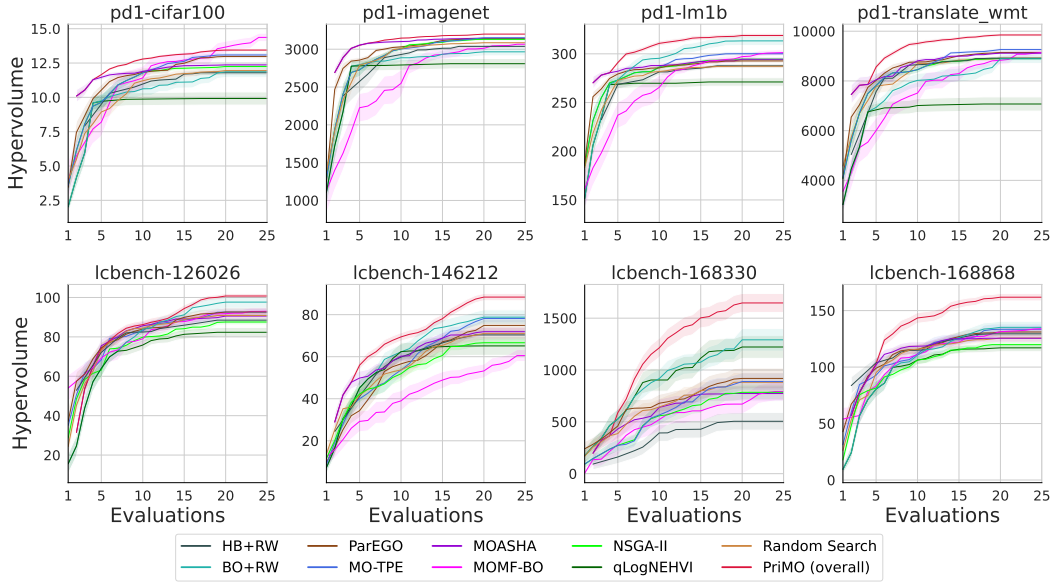


Figure 14: Comparing PriMO against some prominent non-prior MO baselines under the overall prior setting, with respect to the mean dominated HV across all 8 benchmarks.

Processing tasks. Further, Geburek et al. (2024) argued for the usage of LMEM-based significance analysis for HPO benchmarking.

M.1 Data preparation and sanity checks

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

Seed independency check We fitted two LMEMs:

$$\texttt{normalized_hv_regret} \sim \texttt{algorithm} \quad , \tag{9}$$

and,

$$\texttt{normalized_hv_regret} \sim \texttt{(0 + algorithm | seed)} \quad , \tag{10}$$

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

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

$$\texttt{normalized_hv_regret} \sim \texttt{1} \quad , \tag{11}$$

and,

$$\texttt{normalized_hv_regret} \sim \texttt{algorithm} \quad , \tag{12}$$

which confirmed that our benchmarks are informative, as the second model (Equation 12) 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.

M.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 πBO+RW. Under all bad priors, PriMO's performance is not significantly worse than that of most the of best performing non-prior baselines, except BO+RW, but is significantly better than πBO+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 πBO+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.

N 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 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.

O 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

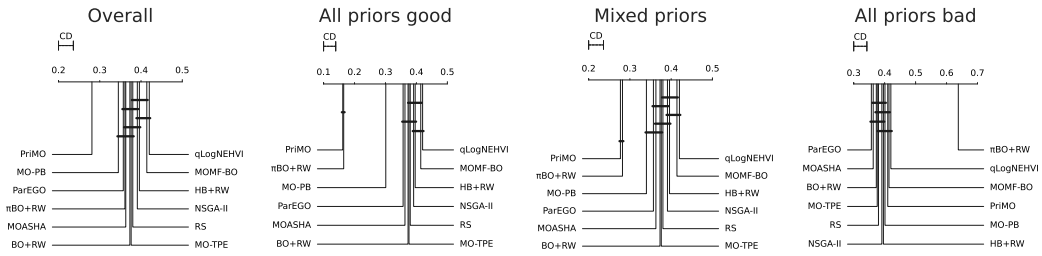


Figure 15: Critical Difference diagrams at **10 evaluations** comparing **early performance** of PriMO against the baselines – BO+RW, πBO+RW, MO-Priorband (MO-PB), MOASHA, HB+RW, MO-TPE, LogNEHVI, Random Search (RS), ParEGO and NSGA-II, under various prior conditions.

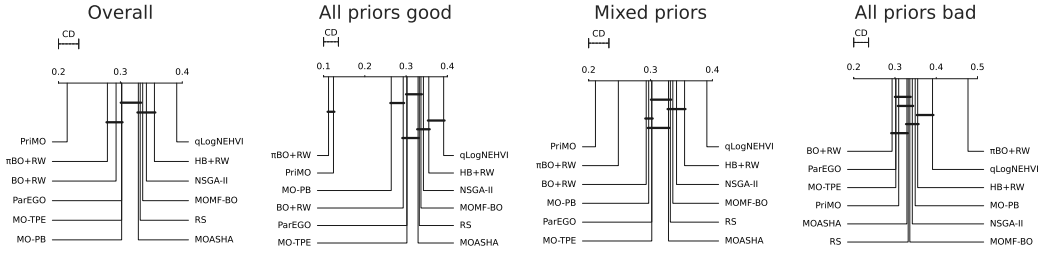


Figure 16: CD diagrams at **20 evaluations** comparing **final performance** of PriMO against other non-prior and prior-guided (adapted to MO) baselines, under various prior conditions.

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$0.15 core hours), model-based baselines such as BO+RW and πBO+RW required significantly longer on average - typically over 5 minutes ($\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.

P LICENSES

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