ZERO-SHOT TRANSFER LEARNING FOR GRAY-BOX HYPER-PARAMETER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot hyper-parameter optimization refers to the process of selecting hyper-parameter configurations that are expected to perform well for a given dataset upfront, without access to any observations of the losses of the target response. Existing zero-shot approaches are posed as initialization strategies for Bayesian Optimization and they often rely on engineered meta-features to measure dataset similarity, operating under the assumption that the responses of similar datasets behaves similarly with respect to the same hyper-parameters. Solutions for zero-shot HPO are embarrassingly parallelizable and thus can reduce vastly the required wallclock time of learning a single model. We propose a very simple HPO model called **Gr**ay-box Zero(**0**)-**S**hot **I**nitialization (GROSI) as a conditional parametric surrogate that learns a universal response model by exploiting the relationship between the hyper-parameters and the dataset meta-features directly. In contrast to existing HPO solutions, we achieve transfer of knowledge without engineered meta-features, but rather through a shared model that is trained simultaneously across all datasets. We design and optimize a novel loss function that allows us to regress from the dataset/hyper-parameter pair unto the response. Experiments on 120 datasets demonstrate the strong performance of GROSI, compared to conventional initialization strategies. We also show that by fine-tuning GROSI to the target dataset, we can outperform state-of-the-art sequential HPO algorithms.

1 INTRODUCTION

Within the research community, the concentration of efforts towards solving the problem of hyper-parameter optimization (HPO) has been mainly through sequential model-based optimization (SMBO), i.e. iteratively fitting a probabilistic response model, typically a Gaussian process (Rasmussen (2003)), to a history of observations of losses of the target response, and suggesting the next hyper-parameters via a policy, acquisition function, that balances exploration and exploitation by leveraging the uncertainty in the posterior distribution (Jones et al. (1998); Wistuba et al. (2018); Snoek et al. (2012)). However, even when solutions are defined in conjunction with transfer learning techniques (Bardenet et al. (2013); Wistuba et al. (2016); Feurer et al. (2015)), the performance of SMBO is heavily affected by the choice of the initial hyper-parameters. Furthermore, SMBO is sequential by design and additional acceleration by parallelization is not possible.

In this paper, we present the problem of zero-shot hyper-parameter optimization as a meta-learning objective that exploits dataset information as part of the surrogate model. Instead of treating HPO as a black-box function, operating blindly on the response of the hyper-parameters alone, we treat it as a gray-box function (Whitley et al. (2016)), by capturing the relationship among the dataset meta-features and hyper-parameters to approximate the response model.

In this paper, we propose a novel formulation of HPO as a conditional gray-box function optimization problem, Section 4, that allows us to regress from the dataset/hyper-parameter pair directly onto the response. Driven by the assumption that similar datasets should have similar response approximations, we introduce an additional data-driven similarity regularization objective to penalize the difference between the predicted response of similar datasets. In Section 5, we perform an extensive battery of experiments that highlight the capacity of our universal model to serve as a solution for: (1) zero-shot HPO as a stand-alone task, (2) zero-shot as an initialization strategy for Bayesian Optimization (BO), (3) transferable sequential model-based optimization. A summary of our contributions is:

- a formulation of the zero-shot hyper-parameter optimization problem in which our response model predicts upfront the full set of hyper-parameter configurations to try, without access to observations of losses of the target response;
- a novel multi-task optimization objective that models the inherent similarity between datasets and their respective responses;
- three new meta-datasets with different search spaces and cardinalities to facilitate the experiments and serve as a benchmark for future work;
- an empirical demonstration of the performance of our approach through a battery of experiments that address the aforementioned research aspects, and a comparison against state-of-the-art HPO solutions for transfer-learning.

2 Related Work

The straightforward zero-shot approaches for HPO consist of random search (Bergstra & Bengio (2012)), or simply selecting hyper-parameters that perform well on general tasks (Brazdil et al. (2003)). Some recent work has also shown that simply selecting random hyper-parameters from a restricted search space significantly outperforms existing solutions, and improves the performance of conventional SMBO approaches (Perrone et al. (2019)). The restricted search space is created by eliminating regions that are further away from the best hyper-parameters of the training tasks.

Another prominent direction for zero-shot HPO depends heavily on engineered meta-features, i.e. dataset characteristics (Vanschoren (2018)), to measure the similarity of datasets. Following the assumption that the responses of similar datasets behave similarly to the hyper-parameters, it has been shown that even the simplest of meta-features (Bardenet et al. (2013)) improve the performance of single task BO algorithms (Feurer et al. (2014; 2015)). The target response is initialized with the top-performing hyper-parameters of the dataset nearest neighbor in the meta-feature space. The shortcomings of using engineered meta-features are that they are hard to define (Leite & Brazdil (2005)), and are often selected through trial-and-error or expert domain knowledge. As a remedy, replacing engineered meta-features with learned meta-features (Jomaa et al. (2019)) compensates for such limitations, by producing expressive meta-features agnostic to any meta-task, such as HPO.

Zero-shot HPO is also posed as an optimization problem that aims to minimize the meta-loss over a collection of datasets (Wistuba et al. (2015a)) by replacing the discrete minimum function with a differentiable softmin function as an approximation. The initial configurations boost the single task BO without any meta-features. In (Wistuba et al. (2015b)), hyper-parameter combinations are assigned a static ranking based on the cumulative average normalized error, and dataset similarity is estimated based on the relative ranking of these combinations. Winkelmolen et al. (2020) introduce a Bayesian Optimization solution for zero-shot HPO by iteratively fitting a surrogate model over the observed responses of different tasks, and selecting the next hyper-parameters and datasets that minimize the aggregated observed loss.

Aside from zero-shot HPO, transfer learning is employed by learning better response models (Wistuba et al. (2016)) based on the similarity of the response. Feurer et al. (2018) propose an ensemble model for BO by building the target response model as a weighted sum of the predictions of base models as well as the target model. In addition to the transferable response models, Volpp et al. (2019) design a transferable acquisition function as a policy for hyper-parameter optimization defined in a reinforcement learning framework. As a replacement to the standard Gaussian process, Perrone et al. (2018) train a multi-task adaptive Bayesian linear regression model with a shared feature extractor that provides context information for each independent task.

In contrast to the literature, we formulate the problem of zero-shot HPO as a gray-box function optimization problem, by designing a universal response model defined over the combined domain of datasets and hyper-parameters. We rely on the embeddings to estimate the similarities across datasets and design a novel multi-task optimization objective to regress directly on the response. This allows us to delineate from the complexity paired with Bayesian uncertainty, as well as the trouble of engineering similarity measures.

3 HYPER-PARAMETER OPTIMIZATION

Consider a dataset $D = \{\left(x^{(\text{Train})}, y^{(\text{Train})}\right), \left(x^{(\text{Val})}, y^{(\text{Val})}\right), \left(x^{(\text{Test})}, y^{(\text{Test})}\right)\}$ for a supervised learning task, with training, validation and test splits of predictors $x \in \mathcal{X}$ and targets $y \in \mathcal{Y}$. We aim at training a parametric approximation of the target using $\hat{y} := f(\theta, \lambda) : \mathcal{X} \rightarrow \mathcal{Y}$, where $\theta \in \Theta$ denotes the parameters and $\lambda \in \Lambda$ its hyper-parameters, by minimizing a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ as:

$$\lambda^* = \underset{\lambda \in \Lambda}{\arg\min} \ \mathcal{L}\left(y^{(\text{Val})}, f\left(x^{(\text{Val})}; \theta^*, \lambda\right)\right) \quad \text{s.t.} \quad \theta^* = \underset{\theta \in \Theta}{\arg\min} \ \mathcal{L}\left(y^{(\text{Train})}, f\left(x^{(\text{Train})}; \theta, \lambda\right)\right) \tag{1}$$

For simplicity we hereafter denote the validation error as the response $\ell(\lambda) := \mathcal{L}\left(y^{(\text{Val})}, f\left(x^{(\text{Val})}; \theta^*, \lambda\right)\right)$. Unfortunately, a direct optimization of the response $\ell(\lambda)$ in terms of λ is not trivial, because θ^* is the result of the minimization problem and its gradients with respect to λ are not easy to compute. Instead, in order to learn the optimal hyper-parameters λ we train a probabilistic surrogate $\hat{\ell}(\lambda; \beta) : \Lambda \times \mathcal{B} \rightarrow R$ parameterized by $\beta \in \mathcal{B}$, with $\mathcal{B}$ as the space of response model parameters, that minimizes the log-likelihood of approximating the response $\ell(\lambda)$ over a set of K evaluations $S := \{(\lambda_1, \ell(\lambda_1)), \ldots, (\lambda_K, \ell(\lambda_K))\}$. We denote P as the probability of estimating the response given a surrogate model. Given the surrogate, the next hyper-parameter to be evaluated $\lambda^{(\text{next})}$ is computed by maximizing an acquisition function $\mathcal{A}$ (e.g. EI (Močkus (1975)) as:

$$\lambda^{(\text{next})} := \underset{\lambda \in \Lambda}{\arg\max} \ \mathcal{A}(\hat{\ell}(\lambda; \beta^*)) \quad \text{s.t.} \quad \beta^* := \underset{\beta \in \mathcal{B}}{\arg\min} \ \sum_{k=1}^{K} \ln P\left(\ell(\lambda_k), \ \hat{\ell}(\lambda_k; \beta)\right) \tag{2}$$

4 META-LEARNING OF CONDITIONAL GRAY-BOX SURROGATES

Let us define a collection of T datasets as $\left\{D^{(1)}, \ldots, D^{(T)}\right\}$ and let $\ell^{(t)}(\lambda)$ measure the response of the hyper-parameter λ on the t-th dataset $D^{(t)}$. Furthermore, assume we have previously evaluated $K^{(t)}$ many hyper-parameters $\lambda_k^{(t)}, \forall k \in \{1, \ldots, K^{(t)}\}$ on that particular dataset. We condition the surrogate $\hat{\ell}$ to capture the characteristics of the t-th dataset, by inputting the meta-features representation of the dataset as $\phi^{(t)}$. Therefore, a dataset-aware surrogate can be trained using meta-learning over a cumulative objective function $\mathcal{O}(\beta)$ as:

$$\mathcal{O}(\beta) \ := \ \sum_{t=1}^{T} \sum_{k=1}^{K^{(t)}} \left(\ell^{(t)}\left(\lambda_k^{(t)}\right) - \hat{\ell}\left(\lambda_k^{(t)}, \phi^{(t)}; \ \beta\right)\right)^2 \tag{3}$$

Our surrogate differs from prior practices, because we do not consider the response to be entirely black-box. Instead, since we know the features and the target values of a dataset even before evaluating any hyper-parameter, we model a **gray-box** surrogate by exploiting the dataset characteristics ϕ when approximating the response ℓ. As a result, if the surrogate faces a new dataset that is similar to one of the T datasets from the collection it was optimized (i.e. similar meta-features ϕ extracted directly from the dataset), it will estimate a similar response. Yet, if we know apriori that two datasets are similar by means of the distance of their meta-features, we can explicitly regularize the surrogate to produce similar response estimations for such similar datasets, as shown in Equation 4.

$$\mathcal{R}(\beta) := \sum_{t=1}^{T-1} \sum_{q=t+1}^{T} \sum_{k=1}^{K^{(t)}} \left(|\phi^{(t)} - \phi^{(q)}|\right) \ \left(\hat{\ell}\left(\lambda_k^{(t)}, \phi^{(t)}; \ \beta\right) - \hat{\ell}\left(\lambda_k^{(t)}, \phi^{(q)}; \ \beta\right)\right)^2 \tag{4}$$

Overall we train the surrogate model to estimate the collection of response evaluations and explicitly capture the dataset similarity by solving the following problem, where $\alpha \in \mathbb{R}$ controls the amount of similarity regularization:

$$\beta^* := \underset{\beta \in \mathcal{B}}{\arg\min} \ \mathcal{O}(\beta) + \alpha \, \mathcal{R}(\beta) \tag{5}$$

The meta-features $\phi^{(t)} := \phi(D^{(t)})$ are computed using the Dataset2Vec end-to-end representation of Jomaa et al. (2019), which is further detailed in Appendix B. On the other hand, Equation 5 is minimized via standard meta-learning optimization routines, as expanded in Appendix D.1. Following the approach of the meta-feature extraction paper, we jointly keep updating meta-features as we train Equation 5 by means of the dataset similarity loss (Jomaa et al. (2019)), Appendix B.1.

NETWORK ARCHITECTURE

The architecture of the models used for each meta-dataset are selected based on their performance on the validation datasets. The hyper-parameters of our model are also analyzed in Appendix F.1. The model is implemented in Tensoflow (Abadi et al. (2016)) and the complete architecture details are provided in Section D. In Appendix E we provide the pseudo-code for optimizing our model.

5 EXPERIMENTS

Our experiments are designed to answer three research questions[1]:

- **Q1**: Can we learn a universal response model that provides useful hyper-parameter initializations from unseen datasets without access to pervious observations of hyper-parameters for the dataset itself?
- **Q2**: Do the proposed suggestions serve as a good initialization strategy for existing SMBO algorithms?
- **Q3**: Aside from zero-shot HPO, does the performance of our method improve by refitting the response model to the observations of the hyper-parameters for the target dataset and how well does our approach compare to state-of-the-art methods in transfer learning for HPO?[2]

5.1 META-DATASET

We create three meta-datasets by using 120 datasets chosen from the UCI repository (Asuncion & Newman (2007)). We then create the meta-instances by training a feedforwad neural network and report the validation accuracy. Each dataset is provided with a predefined split 60% train, 15% validation and 25% test instances. We train each configuration for 50 epochs with a learning rate of 0.001. The hyper-parameter search space is described in Table 1.

Table 1: Hyper-parameter search space for the meta-datasets. The name of each the meta-datasets is inspired by the most prominent hyper-parameter, highlighted in red.

Hyper-parameter	Layout Md	Regularization Md	Optimization Md
Activation	ReLU, SeLU	ReLU, SeLU, LeakyReLU	ReLU, SeLU, LeakyReLU
Neurons	$4, 8, 16, 32$	$4, 8, 16, 32$	$4, 8, 16$
Layers	$1, 3, 5, 7$	$1, 3, 5, 7$	$3, 5, 7$
Layout	$\square, \triangleleft, \triangleright, \diamond, \triangle$	$\square$	$\triangleleft, \triangleright, \diamond, \triangle$
Dropout	$0, 0.5$	0, 0.2, 0.5	0
Normalization	False	False, True	False
Optimizer	ADAM	ADAM	ADAM, RMSProp, GD

The **layout** hyper-parameter (Jomaa et al. (2019)) corresponds to the overall shape of the neural network, and provides information regarding the number of neurons in each layer. For example, all the layers in the neural network with a $\square$ layout share the same number of **neurons**. We introduce an additional layout, $\triangle$, where the number of neurons in each layer is successively halved until it reaches the corresponding number of **neurons** in the centeral layer, then doubles successively.We also use dropout (Srivastava et al. (2014)) and batch normalization (Ioffe & Szegedy (2015)) as regularization strategies, and stochastic gradient descent (GD), ADAM (Kingma & Ba (2015)) and RMSProp (Tieleman & Hinton (2012)) as optimizers. SeLU (Klambauer et al. (2017)) represents

[1]For a better understading of the different problem settings, see Appendix A

[2]The associated code and meta-dataset described will be available upon acceptance.

the self-normalizing activation unit. The search space consists of all possible combinations of the hyper-parameters. After removing redundant configurations the resulting meta-datasets have 256, 288 and 324 unique configurations respectively. For the purposes of our algorithm, we need access to the datasets used to generate the meta-features[3]. Further details are available in Appendix C.

5.2 BASELINES

We introduce two sets of baselines to evaluate against the different aspects of our approach:

ZERO-SHOT HYPER-PARAMETER OPTIMIZATION

- **Random** search (Bergstra & Bengio (2012)) is the simplest approach where the hyper-parameters are selected randomly.
- **NN-⟨METAFEATURE⟩** (Feurer et al. (2015)) refers to the process of selecting the top performing hyper-parameters of the nearest neighboring dataset based on their meta-features. We use two sets of well-established engineered meta-features, which we refer to as **MF1** (Feurer et al. (2015)) and **MF2** (Wistuba et al. (2016)), as well as learned meta-features (Jomaa et al. (2019)), which we denote by **D2V**. The similarity is measured by the Euclidean distance.
- **Ellipsoid** (Perrone et al. (2019)) is also a random search approach, however the hyper-parameters are sampled from a hyper-ellipsoid search space that is restricted to encompass as many optimal hyper-parameters from the training dataset as possible.

SEQUENTIAL-MODEL BASED OPTIMIZATION FOR TRANSFER LEARNING

- **GP** (Rasmussen (2003)) is standard Gaussian process response model with a Matern 3/2 and automatic relevance determination. This approach is trained independently on each dataset.
- **TST-R** (Wistuba et al. (2016)) is a two-stage approach where the parameters of the target response model are adjusted via a kernel-weighted average based on the similarity of the hyperaparameter response between the target dataset and the training datasets.
- **RGPE** (Feurer et al. (2018)) is an ensemble model that estimates the target response model as a weighted combination of the training datasets' response models and the target itself. The weights are assigned based on a ranking loss of the respective model.
- **ABLR** (Perrone et al. (2018)) is a multi-task ensemble of adaptive Bayesian linear regression models with all the tasks sharing a common feature extractor.
- **TAF-R** (Wistuba et al. (2018)) learns a transferable acquisition function, unlike the aforementioned algorithms that focus on a transferable response model, that selects the next hyper-parameter based on a weighted combination of the expected improvement of the target task, and predicted improvement on the source tasks.
- **MetaBO** (Volpp et al. (2019)) is another transferable acquisition function, optimized as a policy in a reinforcement learning framework. This approach, however, demands a pre-computed target response model as part of the state representation.

In our approach, we learn a universal response model based on the underlying assumption that the response is not only dependent on the hyper-parameters, as is assumed in black-box optimization techniques, but also on the dataset itself, presenting the problem as a gray-box function optimization.

5.3 RESULTS AND DISCUSSION

Q1: ZERO-SHOT HPO AS A STAND-ALONE PROBLEM

In Table 2 we report the final normalized regret achieved by the different zero-shot approaches for the first 20 hyper-parameters (Feurer et al. (2015)). Our method consistently provides better

[3]Unfortunately, we could not evaluate our approach on some of the published meta-datasets (Schilling et al. (2016)) due to the unavailability of the associated datasets (original predictors and target values) used for generation of the meta-instances

hyper-parameters for such a small budget[4]. The use of engineered meta-features to represent datasets for HPO solutions is not reliable, as the results achieved by NN-$\langle$MF1$\rangle$ and NN-$\langle$MF2$\rangle$ are no better than random. On the other hand, using the meta-features extracted from the dataset directly, NN-$\langle$D2V$\rangle$serves as a better approximation. Furthermore, random sampling from the restricted hyper-ellipsoid also outperforms the use of initialization strategies based on meta-features. We obtain the zero-shot hyper-parameters via Algorithm 2.

Table 2: Final results on several zero-shot HPO benchmarks. The numbers reported are the average normalized regret after 20 trials for 120 tasks on each meta-dataset. We report the best results in **bold** and underline the second best.

Method	Layout Md	Regularization Md	Optimization Md
Random	6.900	8.909	7.185
NN-$\langle$MF1$\rangle$	7.430	7.542	7.164
NN-$\langle$MF2$\rangle$	7.621	6.788	8.055
NN-$\langle$D2V$\rangle$	6.526	6.820	6.695
Ellipsoid	5.713	6.592	6.665
GROSI(Ours)	**5.438**	**5.859**	**6.242**

As a plausibility argument for the usefulness of our zero-shot strategy, we depict in Figure 1 the top 10 suggested hyper-parameters by our approach, as well as two initialization strategies on the actual response surface. Our picks can be seen colocated near the different optima in the search space whereas hyper-parameters of other strategies are dispersed.

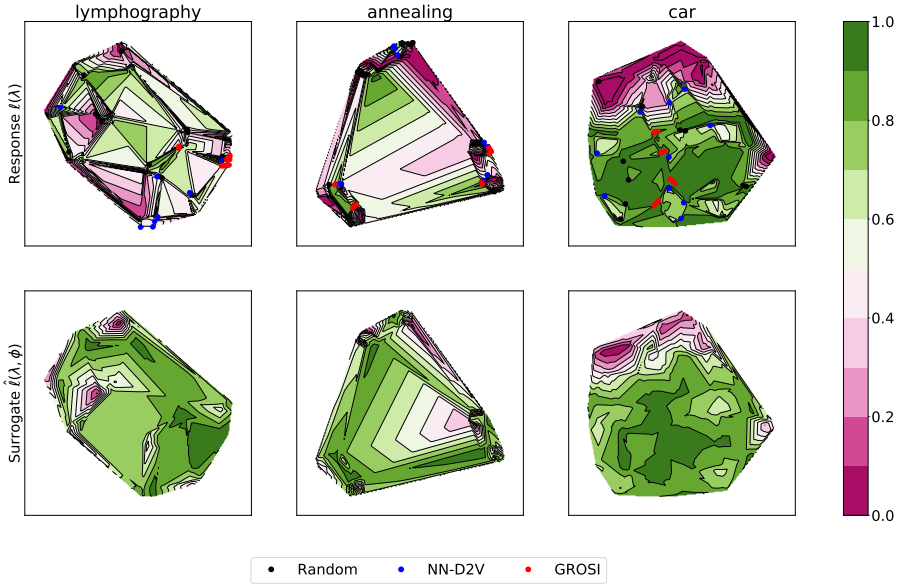

Figure 1: Top 10 hyper-parameters suggested by our approach for test datasets from the three different meta-datasets. We reduce the dimensionality of each search space into a 2D representation via TSNE (Liu et al. (2016)). The first row represents the actual response surface. We notice that the true response and the predicted response are similar, and the location of the predicted minima, in green, overlaps with the minima of the actual response.

[4]Another criteria for comparison would be the duration required to evaluate the response of the best hyper-parameter. On average, our approach requires less time to evaluate the best suggested hyper-parameter, Appendix F.3

Q2: ZERO-SHOT HPO AS AN INITIALIZATION STRATEGY FOR SINGLE-TASK SEQUENTIAL HPO METHODS

We use the aforementioned initialization strategies to warm-start single task GP with a Matern 3/2 kernel and automatic relevance determination as the response model. The quality of our suggested hyper-parameters are reflected in the improved performance of the response model compared to metafeature-based initialization and random search, Table 9. Figure 2 shows that the intermediary hyper-parameters also outperform the baselines. The pseudo-code for sequential model-based optimization is provided by Algorithm 3.

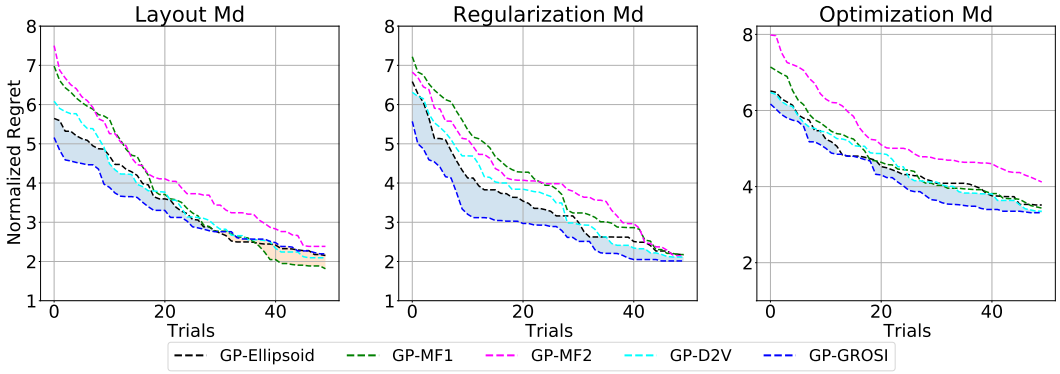

Figure 2: Average normalized regret for single-task sequential HPO methods, a Gaussian process, with different initialization strategies.

Q3: SEQUENTIAL GRAY-BOX FUNCTION OPTIMIZATION

The proposed universal response model provides useful hyper-parameters upfront without access to any observations of losses of the target responses. However, by iteratively refitting the model to the history of observations, the response prediction is improved, as depicted in Figure 3 for an additional 50 trials. We evaluate two policies for selecting the next hyper-parameter after refitting, (1) greedily selecting the hyper-parameter with highest predicted response, GROSI(+1), and (2) selecting the next hyper-parameter randomly from the top 5 hyper-parameters with the highest predicted response, GROSI(+5). In contrast to the baselines that select hyper-parameters through an acquisiton function which capitalizes on the uncertainty of the posterior samples, we incorporate uncertainty by selecting the next hyper-parameter from the top-5 hyper-parameters uniformly at random, and thus introduce a small trade-off between exploration and exploitation.

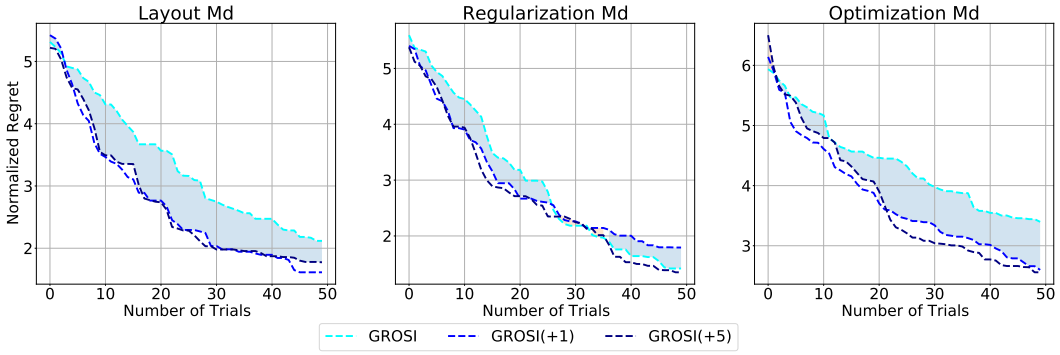

Figure 3: Average normalized regret for GROSI, our zero-shot approach, and the refitted response models GROSI(+1) and GROSI(+5).

Furthermore, our method outperforms the state-of-the-art in transfer learning approaches for HPO. Across all three meta-datasets, we acheive the best final performance, Table 3, as well as better inter-

mediary results[5]. The baselines are warm-started with 20 randomly selected hyper-parameters (Feurer et al. (2015)). It is worth mentioning that the transferable acquisition function trained by MetaBO demonstrates poor performance, which can be attributed to the large search space that prevents the transferable policy from learning to suggest useful hyper-parameters. The improvement over the baselines is shaded with light blue.

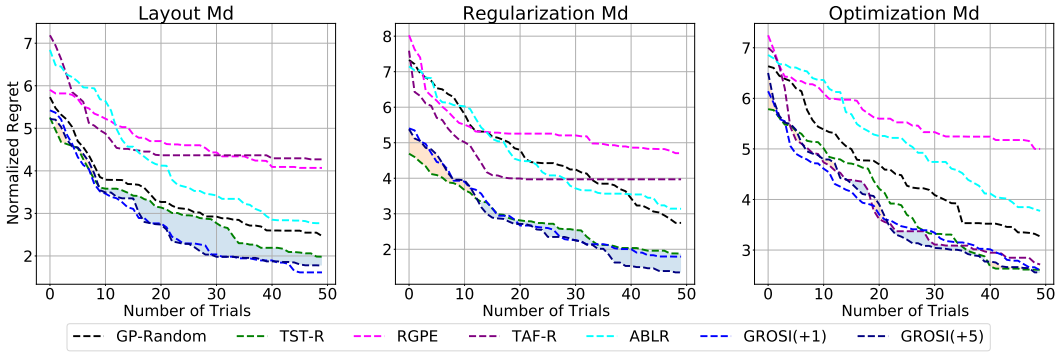

Figure 4: Average normalized regret for state-of-the-art transfer learning HPO methods.

Table 3: Final normalized regret for state-of-the-art transfer learning HPO methods after 50 trials.

Method	Layout Md	Regularization Md	Optimization Md
GP	2.469	2.743	3.268
TST-R	1.982	1.879	2.611
RGPE	4.069	4.706	5.000
ABLR	2.770	3.140	3.775
MetaBO	7.422	7.145	6.468
TAF-R	4.270	3.969	2.714
GROSI(+1)	**1.614**	1.793	2.598
GROSI(+5)	1.774	**1.349**	**2.558**

6 Conclusion

Over the last decade, the hyper-parameter optimization problem has been treated as a black-box function optimization problem, that relies on uncertainty-driven exploration provided by Bayesian optimization techniques. In this paper, we formulate HPO as a gray-box function optimization problem that incorporates an important domain of the response function, the dataset itself. We design a novel universal response model for zero-shot HPO that provides good initial hyper-parameters for unseen datasets in the abscence of associated observations of hyper-parameters. We propose and optimize a novel multi-task objective in order to estimate the response while learning expressive dataset meta-features. We also reinforce the assumption that similar dataset behave similarly with respect to hyper-parameters by introducing a novel similarity-driven regularization technique. The empirical evaluation demonstrates that our zero-shot hyper-parameters are (1) better than meta-feature-based initialization and random search, (2) provide a better support to sequential-model based optimization approaches, (3) and the response model can be fine-tuned iteratively to outperform state-of-the-art in transfer learning algorithms for HPO.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[5]For better visualization, we remove MetaBO from Figure 4, but its results are reported in Table 3

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International conference on machine learning*, pp. 199–207, 2013.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

Pavel B Brazdil, Carlos Soares, and Joaquim Pinto Da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Using meta-learning to initialize Bayesian optimization of hyperparameters. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection-Volume 1201*, pp. 3–10. Citeseer, 2014.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for Bayesian optimization using ranking-weighted Gaussian process ensembles. In *AutoML Workshop at ICML*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Hadi S. Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. Dataset2vec: Learning dataset meta-features. *arXiv preprint arXiv:1905.11063*, 2019.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Věra Krková. Kolmogorov's theorem and multilayer neural networks. *Neural networks*, 5(3):501–506, 1992.

Rui Leite and Pavel Brazdil. Predicting relative performance of classifiers from samples. In *Proceedings of the 22nd international conference on Machine learning*, pp. 497–503, 2005.

Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE transactions on visualization and computer graphics*, 23(3):1249–1268, 2016.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pp. 400–404. Springer, 1975.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, pp. 6845–6855, 2018.

Valerio Perrone, Huibin Shen, Matthias W Seeger, Cédric Archambeau, and Rodolphe Jenatton. Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, pp. 12771–12781, 2019.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Scalable hyperparameter optimization with products of Gaussian process experts. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 33–48. Springer, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Michael Volpp, Lukas Fröhlich, Andreas Doerr, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for Bayesian optimization. *arXiv preprint arXiv:1904.02642*, 2019.

L Darrell Whitley, Francisco Chicano, and Brian W Goldman. Gray box optimization for mk landscapes (nk landscapes and max-ksat). *Evolutionary computation*, 24(3):491–519, 2016.

Fela Winkelmolen, Nikita Ivkin, H Furkan Bozkurt, and Zohar Karnin. Practical and sample efficient zero-shot hpo. *arXiv preprint arXiv:2007.13382*, 2020.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015a.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Sequential model-free hyperparameter tuning. In *2015 IEEE international conference on data mining*, pp. 1033–1038. IEEE, 2015b.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 199–214. Springer, 2016.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

A DETAILED PROBLEM SETTING

By a **learning task** we denote a pair (p, ℓ) of an unknown distribution p of pairs $(x, y) \in \mathbb{R}^{M+L}$, with $M, L \in \mathbb{N}$, and a loss $\ell : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$. A function $\hat{y} : \mathbb{R}^M \to \mathbb{R}^L$ is called a **model for task** p and

$$\ell(\hat{y}; p) := \mathbb{E}_{(x,y)\sim p}(\ell(y, \hat{y}(x)))$$

its (expected) **loss**.

Let a be a **learning algorithm** that yields for every sample D of pairs (x, y) from a task (p, ℓ) and hyper-parameters $\lambda \in \mathbb{R}^P$ a model $\hat{y}$ for the task. We call

$$\ell(\lambda) := \ell(a(D, \lambda); p)$$

the loss (or the response) of hyper-parameters λ. We say validation loss for the loss estimated on fresh validation data.

Sequential single-task hyper-parameter optimization problem. Given an initial number K of pairs (λ_k, l_k) of hyper-parameters and their (validation) losses and a budget $B \in \mathbb{N}$ of trials, find sequentially B many hyper-parameters $\lambda_{K+1}, \ldots, \lambda_{K+B}$, such that their smallest loss

$$\min_{k \in 1:K+B} \ell(\lambda_k)$$

is minimal among all such sequences. To compute the next guess λ_{k+1}, the hyper-parameters $\lambda_1, \ldots, \lambda_k$ tried so far and their (validation) losses $l_k := \ell(\lambda_k)$ can be used.

Zero-shot cross-task hyper-parameter optimization problem. Let p^{task} be an unknown distribution of supervised learning tasks. Given a sample of triples $((p, \ell), \lambda, l)$ of learning tasks (p, ℓ), hyper-parameters λ and their losses l, find for a fresh task (p, ℓ) and a budget $B \in \mathbb{N}$ — without any observations of losses of hyper-parameters on this task — a set $\{\lambda_1, \ldots, \lambda_B\}$ of hyper-parameters, such that their smallest loss $\min_{k \in 1:B} \ell(\lambda_k)$ is minimal among all such sets.

Sequential cross-task hyper-parameter optimization problem. Given both, (i) a sample of triples $((p, \ell), \lambda, l)$ of learning tasks (p, ℓ), hyper-parameters λ and their losses l, and (ii) a fresh task (p, ℓ) and a budget $B \in \mathbb{N}$, find sequentially B many hyper-parameters $\lambda_1, \ldots, \lambda_B$, such that their smallest loss $\min_{k \in 1:B} \ell(\lambda_k)$ is minimal among all such sequences. To compute the next guess λ_{k+1}, the hyper-parameters $\lambda_1, \ldots, \lambda_k$ tried so far and their (validation) losses $l_k := \ell(\lambda_k)$ as well as all data on other tasks can be used.

B META-FEATURE LEARNING

Learning meta-features across datasets of varying schema has proved successful recently, where the meta-feature extractor is formulated as a function over sets. The idea is an extended derivation of the Kolmogorov-Arnold representation theorem (Krková (1992)), which states that a multi-variate function ϕ can be defined as an aggregation of univariate functions over single variables:

$$\phi(x_1, \ldots, x_M) \approx \sum_{j=0}^{2M} h_m\Big(\sum_{m=1}^{M} g_{m,j}(x_m)\Big) \tag{6}$$

It is important to note that ϕ is permutation invariant, i.e. unaffected by any permutation on the input, which allows us to obtain the same output for the same multi-variate data regardless of the order of input. As a simple variant of this formulation (Zaheer et al. (2017)), we can replace the set of functions h_m, with single function h, and $g_{m,j}$ with a function g.

Let us define a collection of T datasets as $\{D^{(1)}, \ldots, D^{(T)}\}$. Each supervised (tabular) dataset $D^{(t)} = \big(x^{(t)}, y^{(t)}\big)$ consists of instances $x^{(t)} \in \mathcal{X} \in \mathbb{R}^{N \times M}$ and targets $y^{(t)} \in \mathcal{Y} \in \mathbb{R}^{N \times C}$ such that N, M and C represent the number of instances, predictors and targets respectively. The dataset can be further represented as a set of smaller components, *set of sets*, $D^{(t)} = \{x^{(t)}_{i,m}, y^{(t)}_{i,c}\}$, with $m \in \{1, \ldots, M\}$, $i \in \{1, \ldots, N\}$, and $c \in \{1, \ldots, C\}$ representing the predictors, instances, and targets respectively. A tabular dataset composed of columns (predictors,targets) and rows (instances)

is reduced to *single* predictor-target pair instead of an instance-target pair. Based on this definition, a meta-feature extractor parameterized as a neural network Jomaa et al. (2019), is formulated in Equation 7. For simplicity of notation, we drop the superscript (t) unless needed.

$$\phi(D) = h\left(\frac{1}{MC}\sum_{m=1}^{M}\sum_{c=1}^{C} g\left(\frac{1}{N}\sum_{i=1}^{N} f(x_{i,m}, y_{i,c})\right)\right) \tag{7}$$

with $f : \mathbb{R}^2 \to \mathbb{R}^{K_f}$, $g : \mathbb{R}^{K_f} \to \mathbb{R}^{K_g}$ and $h : \mathbb{R}^{K_g} \to \mathbb{R}^{K}$ represented by neural networks with K_f, K_g and K output units, respectively. The objective of this formulation is to capture the correlation between each variable (predictor) and its assigned target. In contrast to existing parametric approaches (Edwards & Storkey (2016)) which are bound to fixed-datset schema that makes them ill-suited for real world applications, this meta-feature extractor representation is agnostic to dataset schema, mainly due to its permutation invariance property.

In this paper, we incorporate the meta-feature extractor as part of the response model, effectively learning a conditional response on the dataset meta-features directly such that the approximation is defined as $\hat{\ell}(\lambda^{(t)}, \phi^{(t)}; \beta)$, with $\phi^{(t)} = \phi(D^{(t)})$.

B.1 The Auxiliary Dataset Identification Task

The meta-features extracted by optimizing Equation 5 alone lack any semantic meaning, and thus are not really considered meta-features, as much as they are simple latent represenations, as they might result in dissimilar datasets being colocated in the meta-feature space, i.e. not versatile enough for proper distinction between the datasets. The dataset identification task, introduced previously as dataset similarity learningJomaa et al. (2019), ensures that the meta-features of *similar* datasets are colocated in the meta-feature space, providing more expressive and distinct meta-features for every dataset.

Let p_D a joint distribution over *dataset* pairs such that $(D^{(t)}, D^{(q)}, s) \in T \times T \times \{0, 1\}$ with s being a binary dataset similarity indicator. We define a classification model $\hat{s} : T \times T \to \{0, 1\}$, Equation 8:

$$\hat{s}(D^{(t)}, D^{(q)}) = e^{-\gamma Z\left(\phi^{(t)}, \phi^{(q)}\right)} \tag{8}$$

where $Z : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ represents any distance metric, and γ a tuneable hyper-parameter. Without loss of generality, we use the Euclidean distance to measure the similarity between the extracted meta-features, i.e. $Z\left(\phi^{(t)}, \phi^{(q)}\right) = \left(|\phi^{(t)} - \phi^{(q)}|\right)$, and set $\gamma = 1$. The classification model is trained by optimizing the negative log likelihood, Equation 9:

$$\mathcal{P}(\beta) := -\sum_{(t,q)\sim p_{D^+}} \log\left(\hat{s}(D^{(t)}, D^{(q)})\right) - \sum_{(t,q)\sim p_{D^-}} \log\left(1 - \hat{s}(D^{(t)}, D^{(q)})\right) \tag{9}$$

with p_{D^+} as the distribution of similar datasets, $p_{D^+} = \{(D^{(t)}, D^{(q)}, s) \sim p_D \mid s = 1\}$, and p_{D^-} as the distribution of dissimilar datasets, $p_{D^-} = \{(D^{(t)}, D^{(q)}, s) \sim p_D \mid s = 0\}$.

C Meta-datasets

C.1 Layout Hyper-parameter

Below are some examples of the number of neurons per layer for networks with different **layout** hyper-parameters given 4 neurons and 5 layers:

- Layout $\square$: [4,4,4,4,4]
- Layout $\triangleleft$: [4,8,16,32,64]
- Layout $\triangleright$: [64,32,16,8,4]
- Layout $\diamond$: [4,8,16,8,4]
- Layout $\triangle$: [16,8,4,8,16]

The search space consists of all possible combinations of the hyper-parameters. After removing redundant configurations, e.g. $\triangle$ layout with 1 layer is similar to a $\square$ layout with 1 layer, the resulting meta-datasets have 256, 288 and 324 unique configurations respectively.

C.2 HYPER-PARAMETER ENCODING

Below is description of the encodings applied to our hyper-parameters. We also like to note that the scalar values are normalized between $(0, 1)$.

Table 4: Encoding of the different hyper-parameters used in the meta-dataset.

Hyper-parameter	Encoding
Activation	One-hot encoding
Neurons	Scalar
Layers	Scalar
Layout	One-hot encoding
Dropout	Scalar
Normalization	Scalar
Optimizer	One-hot encoding

D NETWORK ARCHITECTURE

Our model architecture is divided into two modules, $\hat{l} := \phi \circ \psi$, the meta-feature extractor ϕ, and the regression head ψ. The meta-feature extractor $\phi : \mathbb{R}^2 \to \mathbb{R}^{K_h}$ is composed of three functions, Equation 7, namely f, g and h.

The regression head is also composed of two functions, i.e.$\psi : \psi_1 \circ \psi_2$. We define by $\psi_1 : \mathbb{R}^{K_h} \times \Lambda \to \mathbb{R}^{K_{\psi_1}}$ as the function that takes as input the meta-feature/hyper-parameter pair, and by $\psi_2 : \mathbb{R}^{K_{\psi_1}} \to \mathbb{R}$ the function that apprximates the response.

Finally, let **Dense(n)** define one fully connected layer with n neurons, and **ResidualBlock(n,m)** be $m\times$ Dense(n) with residual connections (Zagoruyko & Komodakis (2016)).

We describe the network architectures for each meta-dataset in Table 5. The network architecture of the universal response model for each meta-dataset has been selected based on the best validation performance.

Table 5: The network architecture optimized for every meta-dataset.

Functions	Layout Md
f	Dense(32);6×ResidualBlock(3,32);Dense(32)
g	Layout: $\triangleright$ with 3 layers and 16 Neurons
h	Dense(16);3×ResidualBlock(3,16);Dense(16)
ψ_1	Layout: $\triangleright$ with 4 layers and 4 Neurons
ψ_2	Layout: $\triangleright$ with 4 layers and 4 Neurons
Functions	Regularization Md
f	Dense(16);6×ResidualBlock(3,16);Dense(16)
g	Layout: $\triangleright$ with 3 layers and 16 Neurons
h	Dense(32);3×ResidualBlock(3,32);Dense(32)
ψ_1	Layout: $\triangleright$ with 4 layers and 4 Neurons
ψ_2	Layout: $\triangleright$ with 4 layers and 16 Neurons
Functions	Optimization Md
f	Dense(16);6×ResidualBlock(3,16);Dense(16)
g	Layout: $\triangleright$ with 3 layers and 16 Neurons
h	Dense(16);3×ResidualBlock(3,16);Dense(16)
ψ_1	Layout: $\triangleright$ with 4 layers and 4 Neurons
ψ_2	Layout: $\triangleright$ with 4 layers and 16 Neurons

D.1 TRAINING PROTOCOL

The universal response model is trained by optimizing the multi-task objective.

$$\beta^* := \arg\min_{\beta \in \mathcal{B}} \quad \mathcal{O}\left(\beta\right) + \alpha\, \mathcal{R}\left(\beta\right) + \delta\, \mathcal{P}\left(\beta\right) \tag{10}$$

We use a batch size of 8 tasks sampled randomly with each iteration. We define $p_{\mathcal{T}}$ as the task distribution that represents pairs of datasets and hyper-parameters, i.e. $\mathcal{T}^{(t,k)} = (D^{(t)}, \lambda_k^{(t)}) \in T \times \Lambda$, and $p_{\mathcal{D}}$ be the distribution of the datasets as defined in Section B.1. In Algorithm 1 we describe the pseudo-code for optimizing our response model. We use stochastic gradient descent to optimize the internal model, and Adam optimizer (Kingma & Ba (2015)) to optimize outer loop. We set the number of inner iterations to $v = 5$, and use a learning rate of 0.001 for both optimizers. The code is implemented in Tensoflow (Abadi et al. (2016))

E ALGORITHMS

Algorithm 1 Adaptation step(D)

Require:p_D: distribution over datasets,p_T: distribution over tasks
Require:lr_{inner},lr_{outer}: learning rates
Randomly initialize $\beta \in \mathcal{B}$, the parameters of our response model $\hat{l}$
while not done **do**
    Set $\beta' \leftarrow \beta$
    Sample $(D^{(t)}, \lambda_k^{(t)}) = \mathcal{T}^{(t,k)} \sim p_{\mathcal{T}}$
    for v steps
        sample $D^{(q)} \sim p_{D \setminus (D^{(t)}}$
        Evaluate gradients $\mathcal{G} \leftarrow \nabla_\beta \left(\mathcal{O} + \alpha\, \mathcal{R} + \delta\, \mathcal{P}\right)$
        Compute adapted parameters with stochastic gradient descent: $\beta' \leftarrow \beta' - lr_{\text{inner}} \mathcal{G}$
    Update $\beta \leftarrow \beta - lr_{\text{outer}} \left(\beta - \beta'\right)$
return β

Algorithm 2 Zero-shot HPO

Require: target dataset $D^{(t)}$; response model $\hat{l}$; desired zero-shot hyper-parameters K
$\mathcal{H} \leftarrow \arg\min_{\lambda \in \Lambda}^{K} \hat{l}\left(D^{(t)}, \lambda\right)$
return $\mathcal{H}$

Algorithm 3 Sequential Model-based Optimization Warm-start

Require: target dataset $D^{(t)}$; response model $\hat{l}$; desired zero-shot hyper-parameters K, number of trials I, acquisition function a
Get initial hyper-parameters $\mathcal{H}_0 \leftarrow$Zero-shot HPO
$\lambda^{\text{min}} \leftarrow \arg\min_{\lambda \in \mathcal{H}_0} \left(l(D^{(t)}, \lambda)\right)$
for $i = 1 \ \ldots \ I$
    fit $\hat{l}_i$ to $\mathcal{H}_{i-1}$
    $\lambda \leftarrow \arg\max_{\lambda \in \Lambda} a\left(\hat{l}(D^{(t)}, \lambda)\right)$
    $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \bigcup \{\lambda\}$
    if $l\left(D^{(t)}, \lambda\right) < l\left(D^{(t)}, \lambda^{\text{min}}\right)$
        $\lambda^{\text{min}} \leftarrow \lambda$
return λ^{min}

F ADDITIONAL EXPERIMENTAL RESULTS

F.1 HYPER-PARAMETER SENSITIVITY ANALYSIS

We optimize our response model by minimizing Equation 10, which includes the dataset identification task 9 and the similarity-driven regularizion task, Equation 4, with auxiliary weights δ and α assigned

to both respectively. We report below the performance of our universal response for different auxiliary weights. The results confirm the importance of the auxiliary dataset identification task, as we notice that the overall perforamance is best when $\delta = 1$. Also increasing the weight of the regularization loss improves the performance in general. The reported results throughout the paper are based on $\delta = 1$ and $\alpha = 0.5$. .

.

Table 6: Final results of our universal response model optimized with different auxiliary weights. The numbers reported are the average normalized regret after 20 trials.

Method	Layout Md	Regularization Md	Optimization Md
$\delta = 0.5; \alpha = 0.1$	6.906	6.548	6.834
$\delta = 0.5; \alpha = 0.5$	6.061	6.906	6.474
GROSI(Ours):$\delta = 1; \alpha = 0.5$	**5.438**	**5.859**	**6.242**

F.2 Ablation Study

We perform several ablation experiments to analyze the contribution of each objective to the overall performance. The results are detailed in Table 7, as the normalized average regret across 120 datasets based on a 5-fold cross validation.
Treating zero-shot HPO as a simple regression model by optimizing Equation 3 alone is suboptimal and does not scale across all meta-datasets. We notice that adding the auxiliary dataset identification task, Equation B.1 brings on significant improvement, similarly with the similarity driven regularization, Equatiion 4. This reinforces the notion the responses of similar datasets behave similarly with regards to the hyper-parameters. Both losses help generate more expressive meta-features, the former more directly, by optimizing the inter- and intra-dataset similarities, and the latter indirectly by penalizing the difference in the predicted response.

We also initialize the meta-feature extractor, ϕ, by pretraining it independently. Turns out that it is worse than random initialization. This might be due to the overfitting of the meta-feature extractor parameters, or due to the fact that by learning from scratch, the resulting meta-features are better suited for the task of response prediction.

Table 7: Final results of our universal response model optimized with different objectives. The numbers reported are the average normalized regret after 20 trials.

Method	Layout Md	Regularization Md	Optimization Md
$\mathcal{O}$	7.496	6.516	12.972
$\mathcal{O} + \delta\mathcal{P}$	5.798	6.483	6.332
$\mathcal{O} + \alpha\mathcal{R}$	6.172	6.090	6.488
Pretrained Meta-feature Extractor			
$\mathcal{O} + \alpha\mathcal{R}$	6.514	6.212	6.583
$\mathcal{O} + \alpha\mathcal{R} + \delta\mathcal{P}$	7.411	6.616	10.937
GROSI(Ours)	**5.438**	**5.859**	**6.242**

F.3 Additional Results

Q1: Zero-shot HPO as a stand-alone problem

One of the most important aspects of zero-shot HPO is that it is embarrassingly parallelizable. Consequently, we are not only interested in the final performance, but also the time required to arrive at the optimal hyper-parameter. In Table 8, we report the average time required by each zero-shot approach to find the optimal hyper-parameter. The results do *not* reflect the time required to provide these hyper-parameters, but the time training required time of underlying model. Although our approach is slower in the final dataset, we compensate for this by obtaining better hyper-parameters.

Table 8: Time (seconds) required to evaluate the best hyper-parameter in 20 trials

Method	Layout Md	Regularization Md	Optimization Md
Random	58.9	100.3	129.7
Ellipsoid	76.9	65.9	87.5
NN-MF1	86.8	148.8	**59.9**
NN-MF2	98.9	166.5	69.5
NN-D2V	100.3	127.1	62.4
GROSI(Ours)	**47.8**	**61.4**	75.2

Q2: Zero-shot HPO as an initialization strategy for single-task sequential HPO methods

In Table 9 we report the final peformance of GP warm-started with 20 hyper-parameters. We observe that coupling GP with our initialization strategy improves the overall performance. In Figure 2 we depict the results across the entire 50 trials.

Table 9: Final results on several GP with warm-start benchmarks. The numbers reported are the average normalized regret after 50 trials for 120 tasks per meta-dataset.

Method	Layout Md	Regularization Md	Optimization Md
GP-Ellipsoid	2.153	2.170	3.516
GP-MF1	**1.815**	2.177	3.434
GP-MF2	2.385	2.144	4.120
GP-D2V	2.095	2.115	3.353
GP-Ours	2.196	**2.014**	**3.313**

Q3: Sequential Gray-box Function Optimization

We refit the universal response model to the observations of the response on the target dataset by optimizing the objective $\mathcal{O}$, Equation 3. We illustrate the performance of the different benchmarks in Figure 4. Interestingly, single task GP outperforms some state-of-the art solutions in transfer learning for HPO. Aside from the final performance, our approach suggests better hyper-parameters initially, before TST-R catches up. We follow a greedy policy that exploits the response model unlike the benchmarks that depend on uncertainty modeling to improve exploration. The competitive performance of our approach proves that HPO does not neccessarily need to rely on BO techniques.

Table 10: Final results on several hyper-parameter optimization benchmarks. The numbers reported are the average normalized regret after 50 trials.

Method	Layout Md	Regularization Md	Optimization Md
GP	2.469	2.743	3.268
TST-R	1.982	1.879	2.611
RGPE	4.069	4.706	5.000
ABLR	2.770	3.140	3.775
MetaBO	7.422	7.145	6.468
TAF-R	4.270	3.969	2.714
GROSI(+1)	**1.614**	1.793	2.598
GROSI(+5)	1.774	**1.349**	**2.558**