HomOpt: A Homotopy-Based Hyperparameter Optimization Method

Sophia J. Abraham SABRAHA2@ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Kehelwala D. G. Maduranga GMADURANGA@TNTECH.EDU
Department of Mathematics
Tennessee Tech University
Cookeville, TN 38505

Jeffery Kinnison JKINNISO@ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Zachariah Carmichael ZCARMICH@ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Jonathan D. Hauenstein HAUENSTEIN@ND.EDU
Department of Applied and Computational Mathematics and Statistics
University of Notre Dame
Notre Dame, IN 46556

Walter J. Scheirer WALTER.SCHEIRER@ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Editor: TBD

Abstract

Machine learning has achieved remarkable success over the past couple of decades, often attributed to a combination of algorithmic innovations and the availability of high-quality data available at scale. However, a third critical component is the fine-tuning of hyperparameters, which plays a pivotal role in achieving optimal model performance. Despite its significance, hyperparameter optimization (HPO) remains a challenging task for several reasons. Many HPO techniques rely on naive search methods or assume that the loss function is smooth and continuous, which may not always be the case. Traditional methods, like grid search and Bayesian optimization, often struggle to quickly adapt and efficiently search the loss landscape. Grid search is computationally expensive, while Bayesian optimization can be slow to prime. Since the search space for HPO is frequently high-dimensional and non-convex, it is often challenging to efficiently find a global minimum. Moreover, optimal hyperparameters can be sensitive to the specific dataset or task, further complicating the search process. To address these issues, we propose a new hyperparameter optimization method, `HomOpt`, using a data-driven approach based on a generalized additive model (GAM) surrogate combined with homotopy optimization. This strategy augments established opti-

mization methodologies to boost the performance and effectiveness of any given method with faster convergence to the optimum on continuous, discrete, and categorical domain spaces. We compare the effectiveness of `HomOpt` applied to multiple optimization techniques (*e.g.*, Random Search, TPE, Bayes, and SMAC) showing improved objective performance on many standardized machine learning benchmarks and challenging open-set recognition tasks.

1. Introduction

Selecting appropriate hyperparameters for a particular machine learning task is a challenging problem due to the vast search space, non-linear or non-monotonic effects on performance, and the complexity of the optimization landscape. Machine learning models consist of two distinct types of parameters: *elementary parameters*, which are learned during model training, and *hyperparameters*, which are higher-level free parameters that structure and control the training process. Most commonly, hyperparameters are set heuristically by practitioners before training, making the process prone to inconsistencies and biases across different individuals or experiments.

Automated machine learning (AutoML) aims to automate the entire machine learning pipeline, with automatic hyperparameter optimization (HPO) as a key subfield. HPO seeks to find the optimal hyperparameters for a given model to achieve the best possible performance on a specific task while improving reproducibility and fairness. By automating the HPO process, the search for optimal hyperparameters becomes more systematic and standardized, ensuring more consistent results when the same optimization algorithm is applied to the same problem. The performance of a model depends on the algorithm's architecture, the training data, and the chosen hyperparameters. Consequently, model selection is not solely an algorithmic determination, as hyperparameters significantly impact an algorithm's capability to learn. Hyperparameters, which can be real-valued, integer-valued, binary, or categorical, need to be set before training and differ from the elementary parameters learned from the data. Hyperparameter search spaces serve as proxy domains for loss functions, which can be defined over nonlinear, non-convex spaces with many oscillations. This complexity makes the optimization process non-trivial. Identifying the best model for a particular learning task involves selecting hyperparameters to achieve the best performance on a specified task. This process is known as the hyperparameter optimization problem.

Various kinds of automated hyperparameter search approaches have been proposed to solve this optimization problem, ranging from simple methods like grid search (Duan and Keerthi, 2005) and random search (Bergstra and Bengio, 2012), to more rigorous methods like Bayesian optimization (Bergstra et al., 2011, 2015), gradient-based learning (Bengio, 2000; Maclaurin et al., 2015), and surrogate model approaches (Zhang et al., 2015). These methods have been used in many fields and have their own strengths and drawbacks. For example, grid search and random search are relatively simple to implement but can be computationally expensive and inefficient in exploring the hyperparameter space. Bayesian optimization is more efficient in searching the space but can be sensitive to the choice of acquisition function and prior distributions. Gradient-based learning requires differentiable hyperparameters, which may not always be available, and surrogate model approaches depend on the quality of the surrogate model to guide the search effectively.

Our main contribution in this paper is a Generalized Additive Model (GAM) Surrogate Homotopy Hyperparameter Optimization strategy: `HomOpt`. Following a data-driven approach, GAMs are used to create a sequence of surrogate models of the hyperparameter space as more data is obtained and avoids the "curse of dimensionality" (Nisbet, 2018, Chap. 7). Although this cre-

ates a discrete sequence of GAMs, it is natural to consider how one GAM can be continuously deformed to the next in this sequence. A continuous deformation between objects is called a *homotopy* where the homotopies of interest are those between GAMs. Therefore, homotopy-based optimization techniques are then applicable for hyperparameter optimization by solving the family of optimization problems where the objective function is a homotopy between GAMs. For example, homotopy-based optimization methods have been effectively used for solving nonlinear equations and nonlinear optimization problems (Bates et al., 2013; Griffin and Hauenstein, 2015; Chen and Hao, 2019). In particular, `HomOpt` employs a homotopy hyperparameter optimization strategy as one moves through the sequence of GAMs aiming to boost performance and effectiveness. To the best of our knowledge, homotopy-based methods have not been applied within the hyperparameter optimization setting on black-box functions.

One key aspect is that `HomOpt` can be used to augment any base optimization strategy. For example, as a Bayesian optimization approach is collecting data regarding the hyperparameter space, one can simultaneously be employing `HomOpt` on the known data aiming to converge faster. This is demonstrated in our experiments that compare base methods with ones augmented with `HomOpt` on both closed-set and open-set learning. While closed-set learning only works on identifying predefined classes, open-set learning (Scheirer et al., 2012) trains models that have incomplete knowledge of the world they must operate in and allow the incremental learning setting, where newly identified classes are added to the recognition model over time. We tune both a multitude of common machine learning-based classifiers and the Extreme Value Machine (EVM) (Rudd et al., 2017), which is a scalable nonlinear open-set classifier.

To summarize, our contributions are the following:

1. We demonstrate that `HomOpt`, which does not rely on strong assumptions about the behavior of the underlying objective function, augments traditional methods and converges to optima faster than multiple base methods alone by approximating regions of interest rather than the entire hyperparameter surface.

2. We demonstrate the use of `HomOpt` across continuous, discrete, and categorical search domains, showcasing its versatility and effectiveness in various settings.

3. We release an easy to use, flexible, and open-sourced software package to apply `HomOpt` to model search for rapid deployment in general search problems.

2. Related Work

Hyperparameter Optimization. Hyperparameter optimization methods involve searching for optimal hyperparameter values to effectively identify high-performing models within the hyperparameter space. Non-Bayesian approaches, such as hand-tuning and grid search (Duan and Keerthi, 2005), are simple to use. However, they rely upon adequate domain knowledge, which may not readily be available. Furthermore, these methods may overlook optimal values in continuous domains and inevitably prove brittle when applied to unseen cases (Li et al., 2017). Random search (Bergstra and Bengio, 2012) mitigates this issue by removing the requirement of discretizing the search space and provides a larger coverage of the hyperparameter space. Although random search is simple to use, it is often inefficient sampling-wise. In order to narrow the scope of the search, multiple software frameworks for hyperparameter search based on random search have been proposed, including

those by Bergstra et al. (2015, 2011); Betrò (1992); Wu et al. (2019); Bengio (2000); Maclaurin et al. (2015); Ilievski et al. (2017).

Population-Based Approaches. Population-based algorithms take inspiration from biology and improve upon computational efficiency over purely random search-based methods (Loshchilov and Hutter, 2016). These methods include Evolutionary Algorithms and swarm algorithms like particle swarm optimization (PSO) (Boeringer and Werner, 2005), which iteratively update the generation of hyperparameters with a stochastic velocity term. While effective for lower dimensional spaces, methods like PSO can get stuck in a local optimum for high dimensional, complex scenarios with low convergence rates over the iterative process (Kennedy and Eberhart, 1995).

Bayesian Optimization. In order to perform the search using statistical analysis, Bayesian optimization methods have been proposed (Bergstra et al., 2015, 2011; Betrò, 1992; Wu et al., 2019) based on Bayes' theorem. It sets a prior over the optimization function and gathers the information from the previous sample to update the posterior of the optimization function. A utility function selects the next sample point to maximize or minimize the optimization function. One example of a popular Bayesian method is the Sequential Model-Based Algorithm Configuration (SMAC) (Lindauer et al., 2017) consisting of Bayesian optimization combined with a simple racing mechanism on the instances to efficiently decide which of two configurations performs better.

Tree-Structured Parzen Estimators (TPE) (Bergstra et al., 2011, 2015), another Bayesian method, is a sequential model-based optimization (SMBO) approach. SMBO methods sequentially construct models to approximate the performance of hyperparameters based on historical measurements, and then subsequently choose new hyperparameters to test with based on a constructed model.

Gradient-based Approaches. Gradient-based optimization methods (Bengio, 2000; Maclaurin et al., 2015) compute gradients of cross-validation performance with respect to all hyperparameters by chaining derivatives backwards through the entire training procedure. This is advantageous over other methods since information regarding the shape of the objective surface and behaviors including extrema in the parameter space can be acquired. Hyperparameter gradients are computed by reversing the dynamics of stochastic gradient descent. Gradients enable the optimization of the hyperparameters, including step-size, momentum schedules, weight initialization distributions, richly parameterized regularization schemes, and neural network architectures. However, information about the gradients is often unavailable, computing gradients is computationally expensive, and gradient-based approaches suffer from inefficiency when learning long-term dependencies (Bengio et al., 1994).

Surrogate-based Approaches. Surrogate-based optimization methods (Eggensperger et al., 1970; Xie et al., 2021; McLeod et al., 2018) are used when an objective function is expensive to evaluate. The Surrogate Benchmarks for Hyperparameter Optimization (Eggensperger et al., 1970) uses the following strategy: cheap-to-evaluate surrogates of real hyperparameter optimization benchmarks that yield the same hyperparameter spaces and feature-similar response surfaces. Specifically, this approach trains regression models on data representing a machine learning algorithm's performance under a broad range of hyperparameter configurations and then cheaply evaluates hyperparameter optimization methods using the model's performance predictions instead of the actual algorithm. In McLeod et al. (2018), a Gaussian Process-based (GP) model was used to identify a convex region and a probability-based approach was used to estimate a convex region centered around the posterior minimum. Our approach uses the exploitation from multiple optimization techniques to identify the region of interest for surrogate approximation.

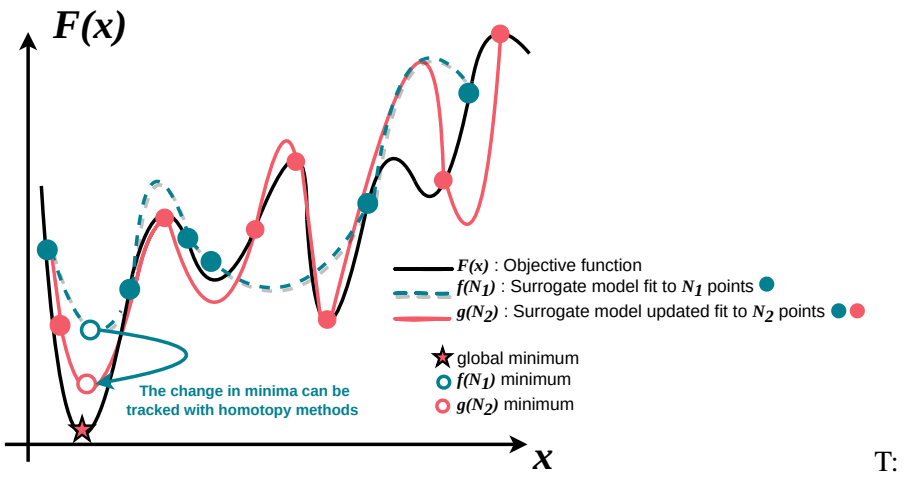
Homotopy-based Approaches. Continuation or homotopy methods (Rheinboldt, 1981; Allgower and Georg, 1990) have long served as a useful technique in numerical methods and result in accurate classification performance without excessive computational effort. Preliminary results indicate the potential of this approach with respect to both training time and classification accuracy and it is also known as globally convergent and solution exhaustive for nonlinear polynomials (Allgower and Georg, 1990; Bates et al., 2013). There has also been recent interest in homotopy continuation for training and analyzing neural networks. This was first presented by (Chow et al., 1991) and thereafter by (Pathak, 2018; Chen and Hao, 2019; Mehta et al., 2022). To train and initialize neural networks, data continuation and model continuation are used with similar objectives of decomposing the original task into a sequence of tasks with increasing levels of difficulty. To the best of our knowledge, the continuation method for training a neural network is used mainly for elementary parameter optimization, and is also data specific. We are extending the idea of using homotopy optimization into general hyperparameter optimization.

3. Homotopy-Based Hyperparameter Optimization

We propose a new data-driven hyperparameter optimization approach based on the following two core aspects. First, surrogates are used to model the objective since the function and its gradient are computationally expensive to evaluate. In particular, a sequence of surrogate models is constructed as new data is gathered. A continuous deformation from the current surrogate model to the updated one that includes the new data is formed. Second, our approach uses this continuous deformation to construct a homotopy path between a local minimum for the current model and a local minimum for the updated model. Since the data may be generated by any base optimization strategy, our approach can be used to augment other strategies aiming to converge in fewer iterations.

The first step is to select a family of surrogate models. Some examples include polynomial spline fitting with several degrees of freedom, radial basis function interpolation, and a generalized additive model (GAM) (Hastie and Tibshirani, 1990), which is a type of statistical model that is used to describe the relationship between a response variable and one or more predictor variables. GAMs are similar to generalized linear models (GLMs), but they allow for nonlinear relationships between the response (output or target variable we try to predict) and predictor variables (input features used to make the prediction) by using smooth functions to model the relationships. These can be estimated using penalized regression techniques, such as penalized splines, and they can provide more flexible and accurate models than GLMs in many cases. The results of some initial empirical experiments along with some theoretical advantages described below suggest that the extrapolation behavior is more reasonable for GAMs and, for multidimensional data, is more suitable and provides a more accurate fit. In our implementation we utilize GAMs via PyGAM (Servén et al., 2018).

From a theoretical standpoint, GAMs provide a more interpretable model than other surrogate models as the smoothing functions that are used to model the relationships between the response and predictor variables can be visualized and analyzed directly. This can be especially useful for understanding and explaining the underlying patterns and relationships in the data, which can be difficult to do with more complex and opaque models like random forests. Additionally, GAMs can provide more accurate predictions for certain types of data, such as data with nonlinear or non-monotonic relationships between the response and predictor variables. Moreover, relationships between the response and predictor variables in GAMs can provide insight into the importance of each feature. For example, the magnitude and significance of the coefficients of the smoothing functions can be



T:

Figure 1: Illustration of homotopy parametrization between the initial samples and updated data samples. The black line represents the objective function that is being evaluated. The first surrogate model $f(N_1)$ is fit on the initial set of samples indicated by the blue circles. With more samples (indicated in pink), the updated surrogate $g(N_2)$ yields a new minimum. As the number of data samples increases, the approximation of the minimum improves. These changing minima can be tracked with homotopy methods.

used to determine the relative importance of each feature, and the shape of the smoothing function can provide additional information about the nature of the relationship between the response and predictor variables. Although this is out of the scope of this paper, it could be a useful tool for understanding and interpreting the results of the hyperparameter optimization process.

Following the selection of the family of surrogate models, the second step is to construct a continuous deformation between surrogate models as new data is collected and then utilize homotopy methods to track a local minimum along this deformation as illustrated in Figure 1. For a general overview of homotopy methods, with a focus on nonlinear polynomial functions, see (Bates et al., 2013). In the context of optimization using surrogate models that depend upon continuous variables, local minima are critical points of the surrogate model, *i.e.*, the gradient of the surrogate model vanishes at each local minimum. Although computing all critical points of the objective function using homotopy methods has shown to be useful in some applications (Mehta et al., 2022; Baskar et al., 2022), the approach utilized here does not rely upon computing all critical points to provide a scalable framework to higher-dimensional and non-polynomial systems.

In our approach, one starts with a GAM $f(x)$ fit to N_1 points. Viewing f as an objective function of an unconstrained optimization problem, suppose that one has computed a local minimum x_{old} for f. As new data is collected, one creates a new GAM $g(x)$ fit to N_2 points. Thus, one can construct a continuous family of functions that deforms from f to g, such as the linear family

$$H(x,t) = t \cdot f(x) + (1-t) \cdot g(x) \quad \text{where} \quad H(x,1) = f(x) \quad \text{and} \quad H(x,0) = g(x). \tag{1}$$

Therefore, one now has a family of unconstrained optimization problems with objective function $H(x,t)$ and a known local minimum when $t = 1$ at x_{old}. This yields a homotopy path of local minima of $H(x,t)$ parameterized by t that emanates from x_{old} at $t = 1$. When f and g are ana-

Algorithm 1: Homotopy Optimization Framework

1 **Input** $\mathcal{T}$ time used for optimization, $\mathcal{N}_\mathcal{T}$ trials used for optimization, $\mathcal{W}$ samples used for surrogate approximation, $\mathcal{D}$ distance threshold controlling localization of surrogate, `Inner_Method` which is the method used to gather new sample data points, k the proportion of trial data to use
2 **Output** Optimal hyperparameter set
3 $\mathcal{C}_\mathcal{T} \leftarrow 0$ ▷ Variable to track the number of completed trials
4 $trial\_data \leftarrow$ empty array
5 **while** *TIME* $\leq \mathcal{T}$ or $\mathcal{C}_\mathcal{T} \leq \mathcal{N}_\mathcal{T}$ **do**
6 **if** $\mathcal{C}_\mathcal{T} < \mathcal{W}$ or $\mathcal{C}_\mathcal{T} \bmod 5 \in \{0, 2\}$ **then**
7 $hparams \leftarrow$ Generate hyperparameters using `Inner_Method`
8 **else if** $\mathcal{C}_\mathcal{T} \bmod 5 \in \{3, 4\}$ **then**
9 $svar \leftarrow \text{variance}(\text{best 10 hyperparameter sets in } trial\_data) \cdot \mathcal{D}$
10 $hparams \leftarrow$ lowest-loss hyperparameters in $trial\_data$
11 $hparams \leftarrow hparams + \text{uniform}(\mu = -svar, \sigma^2 = 2 \cdot svar)$
12 **else**
13 Fit f to the $\text{round}(k \cdot \mathcal{C}_\mathcal{T})$ most recent entries of $trial\_data$ (hyperparameters as features and losses as the targets)
14 Fit g to $trial\_data$
15 Use homotopy optimization (Alg. 2) to minimize along $H(x, t) = t \cdot f(x) + (1 - t) \cdot g(x)$ with the lowest-loss hyperparameters in $trial\_data$ as the initial guess
16 Predict loss values for all of the hyperparameters returned by Alg. 2 using g
17 $hparams \leftarrow$ lowest-loss hyperparameters from the previous line
18 **end**
19 $loss \leftarrow$ Evaluate $hparams$ on the function being optimized
20 Append $(hparams, loss)$ to $trial\_data$
21 **end**
22 $hparams \leftarrow$ the lowest-loss hyperparameters in $trial\_data$
23 **return** $hparams$

lytic, one can use standard homotopy theory (Sommese and Wampler, 2005) to provide guarantees on existence and smoothness of such a path based on f and g via the implicit function theorem that ends at the desired local minima x_{new} of g at $t = 0$. However, since we aim to apply this framework to general hyperparameter optimization problems, we do not rely upon smoothness or differentiability by employing techniques which do not require known derivatives. In particular, a high-level description of the homotopy optimization framework `HomOpt` is given in Algorithm 1. The key step of that algorithm is in line 15 which is described in Algorithm 2. In particular, line 7 of Algorithm 2 uses Nelder-Mead optimization (Nelder and Mead, 1965) to minimize the homotopy function. Nelder-Mead does not require or use function gradient information and is appropriate for optimization problems where the gradient is unknown or cannot be reasonably computed.

Algorithm 2: Homotopy Optimization

Input $\mathcal{N}$ number of steps along interval, homotopy function $H(x,t)$, $x^{(0)} = x^0$ a local minimum of $H(x,1)$
Output Local minimum of $H(x,0)$
$\Delta \leftarrow 1/\mathcal{N}$
$t \leftarrow 1$
for $k \leftarrow 1$ *to* $\mathcal{N}$ **do**
    $t \leftarrow t - \Delta$
    Use Nelder-Mead optimization to minimize $H(x,t)$ starting with $x^{(k-1)}$ to obtain $x^{(k)}$.
return $x^{(N)}$

3.1 One-Dimensional Illustration

To illustrate `HomOpt`, we consider the optimization of the test function $p(x)$ defined by Gramacy and Lee (2012) on the domain $[0.5, 2.5]$, where

$$p(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4. \tag{2}$$

The plot of $p(x)$ is shown in blue in Figure 2. Consider two GAMs f and g constructed from 10 and 20 sample points, respectively, as shown in Figure 2(a). Note that the number of samples used in f and g were selected arbitrarily for illustration purposes. The strategy for `HomOpt` is to consider the homotopy path from a known local minimum of $f(x)$ to a local minimum of $g(x)$ that we want to compute. Figure 2(b) shows the optimized surrogate curves $f(x)$ and $g(x)$, along with the corresponding optimal point obtained after eight iterations of the `HomOpt` method demonstrating convergence to the global minimum.

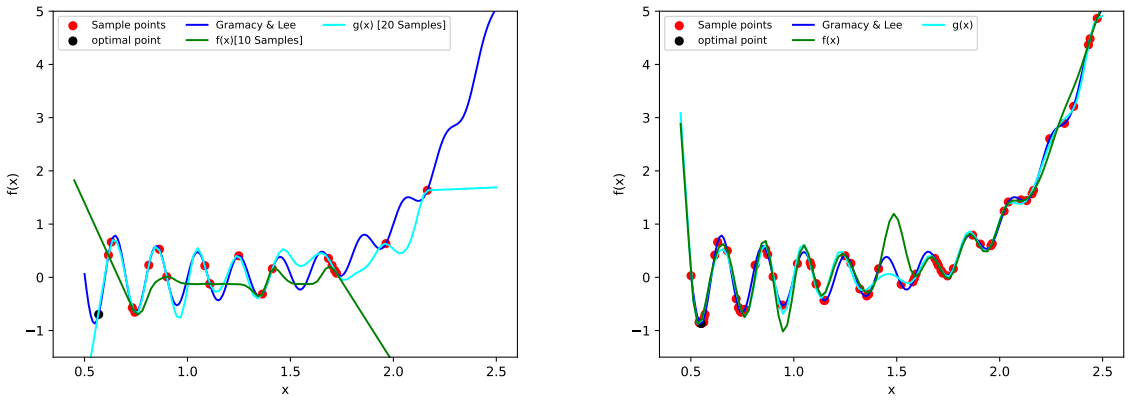
3.2 Two-Dimensional Example

Now consider the two-dimensional case. The Griewank function, introduced by Griewank (Surjanovic and Bingham, 2013), is a standard test example in optimization problems in the 2D plane. However, to prevent the global minimum from being located at the origin and increase the difficulty of the problem, a modified function is considered over the domain $[-20, 20]^2$:

$$g(x,y) = \frac{(x-5)^2 + (y+3)^2}{40} - \cos(x-5) \cdot \cos\left(\frac{y+3}{\sqrt{2}}\right) + 1 \tag{3}$$

which is plotted in Figure 3(a).

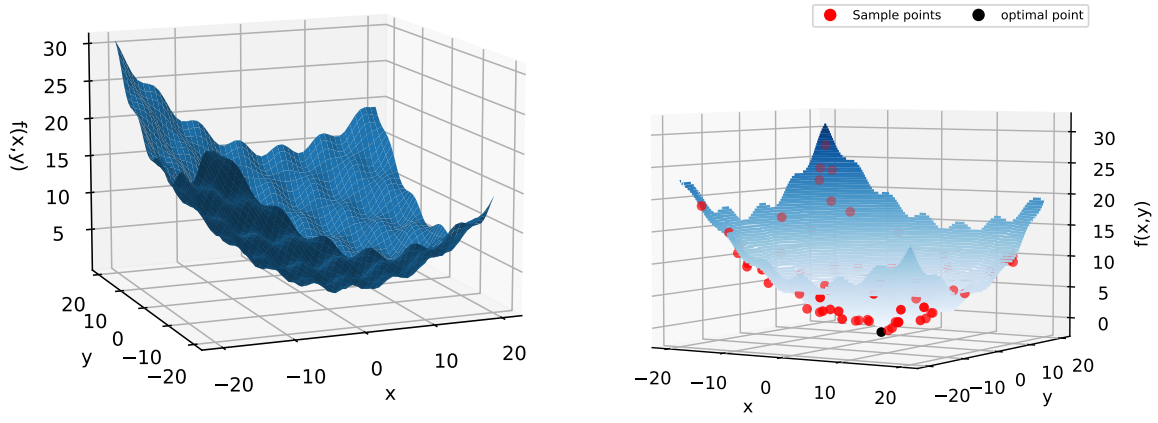
In Figure 3(b), the optimal point (in black) for the Griewank function is shown along with the sampled points (in red). `HomOpt` was able to successfully converge to the optimal point within 100 sampled points. The Griewank function is relatively simple and a low-dimensional problem and as such 100 sampled points was sufficient to accurately approximate the objective function and converge to the optimum. It is important to note that for more complex and higher dimensional problems, it may be necessary to use a larger number of sampled points. As depicted in Figure 4, the black grid representing the approximation of `HomOpt` improved as more sample points were added, learning the overall shape of the Griewank function within 40 sample points.



(a) Start of optimization

(b) After 8 optimization steps

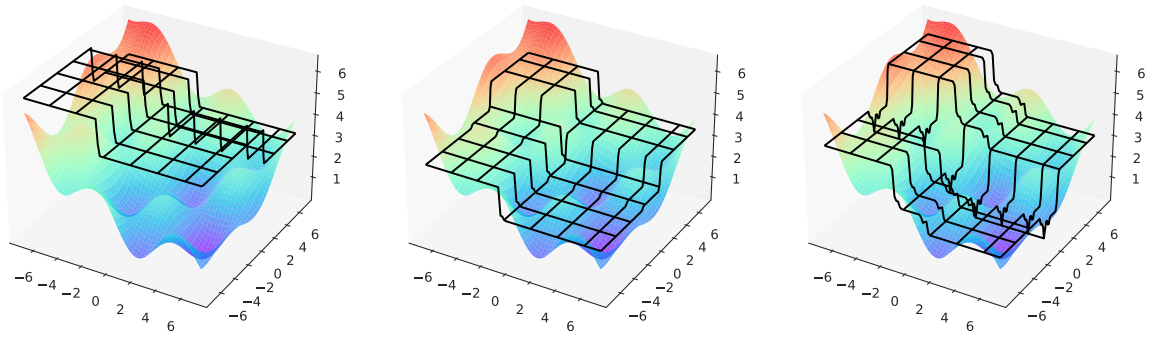
Figure 2: The Plot of the Gramacy and Lee function on the domain $[0.5, 2.5]$ (blue curve). The strategy used by `HomOpt` is to find a homotopy continuation path from the minimum of $f(x)$ to the minimum of the $g(x)$. (a) Local surrogate approximations of $f(x)$ (green curve), and surrogate approximation of $g(x)$ (cyan curve) at the initial stages. (b) Local surrogate approximations of $f(x)$ (green curve), and surrogate approximation of $g(x)$ (cyan curve) after eight homotopy optimization steps.



(a) Griewank Function

(b) `HomOpt` convergence to optimal point

Figure 3: (a) Plot of the modified Griewank function on the domain $[-20, 20]^2$. (b) `HomOpt` converged towards the optimal point shown (black point) using 100 randomly sampled points (red points).



(a) Surrogate from 10 samples (b) Surrogate from 20 samples (c) Surrogate from 40 samples

Figure 4: Plots of the modified Griewank function on the domain $[-7, 7]^2$ with the surrogate approximation plotted as a black grid fit to different numbers of samples. As the numbers of samples increases, the surrogate function begins to get a better approximation of the underlying function and converge towards the global optimum.

3.3 Optimization Software Framework

The baseline search framework utilized the massively Scalable Hardware-Aware Distributed Hyperparameter Optimization (SHADHO) software framework (Kinnison et al., 2018). SHADHO presents a comprehensive survey of existing hyperparameter optimization methods and facilitates baseline Search Strategies, such as Random search, Bayesian optimization, PSO, SMAC, and TPE. SHADHO is designed to scale effectively in distributed computing environments, making it suitable for optimizing complex machine learning models with large hyperparameter search spaces. In addition to serving as a hyperparameter optimization tool, SHADHO has the added utility of considering hardware specifications when conducting a search. This framework calculates the relative complexity of each search space and monitors performance on the learning task over all trials. These metrics are then used as heuristics to assign hyperparameters to distributed workers based on their hardware. We extended SHADHO's framework by integrating `HomOpt`. This extension is now available in the most recent release of SHADHO[1].

4. Experiments

We evaluate `HomOpt` on a multitude of tasks. We begin with a collection of machine learning benchmarks for classification tasks on tabular data using a Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest, XGBoost, and Logistic Regression provided through HPOBench (Eggensperger et al., 2021).

Additionally, we include a set of difficult *open-set* classification experiments. In the open-set scenario, models have incomplete knowledge of the world they must operate in and unknown classes are queried during testing (Scheirer et al., 2012). Open-set classification is notoriously sensitive to hyperparameters and provides a complex scenario where the loss landscape is arbitrary, polluted by noise, and may consist of steep gradients resulting in the absence of regularity. This set of

1. Code can be found at `https://github.com/jeffkinnison/shadho`

experiments also captures how well `HomOpt` can boost methods in changing environments and unseen conditions.

We use the Extreme Value Machine (EVM) (Rudd et al., 2017), which is a scalable nonlinear classifier that supports open-set classification by rejecting inputs that are beyond the support of the training set. The EVM relies on a strong feature representation and every represented sample in the feature representation becomes a point. It utilizes a binning strategy that groups all the points in their feature representation by their corresponding label. These bins are utilized to create a "1 vs. rest" classifier for each known class. It generates a classifier where a Weibull distribution is fit on the data for each known class and is made to avoid the negative data points (unknown classes). This process is repeated for all known classes. When a new data point (a sample represented by its feature vector) is provided to the EVM, it is evaluated in the feature space, and the probability of the point belonging to each representative class is determined.

Parameter	**Description**	**Domain**
Threshold	Probability threshold used to determine if an input coordinate point should be classified as 'unknown' if the point falls below this probability of inclusion.	$[0, 1]$
Tailsize	Defines how many negative samples are used to estimate the model parameters.	$[0, 0.5]$
Cover threshold	The probability threshold used to eliminate extreme vectors if they are covered by other extreme vectors with that probability.	$[0, 1]$
Distance multiplier	The multiplier to compute margin distances.	$[0, 1]$
Distance function	The distance function used to compute the distance between two samples.	Cosine or Euclidean

Table 1: Hyperparameters used in training the EVM.

The hyperparameters involved in training the EVM are included in Table 1. The datasets used in the HPOBench experiments can be found in Table 3 and the benchmark details along with their configuration spaces can be found in Table 4. For further specifications regarding specifics on the benchmarks and datasets from HPOBench please refer to the original paper (Eggensperger et al., 2021).

4.1 Evaluation Criteria

For the HPOBench classification benchmarks, the optimizer performance is evaluated based on the validation performance (*i.e.*, the objective value seen by the optimizer). This objective value was minimized over $1 -$ accuracy and a summary across all of the benchmark experiments are reported. Additional results including the corresponding values for test scores are included for each experiment in Appendix A.

With respect to the HPOBench experiments, we illustrate the performance of each method by plotting the *simple regret* of the minimization problem computed as

$$R_t = f(x_{\text{best-so-far}}) - \min_{i=1}^{t} f(x_i) \tag{4}$$

Experiment	# Search Trials	# Known Classes	# Unknown Classes	Training	Validation	Negatives	Testing
MNIST	1000	6	4	28824	12000	19176	10000
LFW	1000	34	5715	1333	2481	6110	3309

Table 2: Summary of dataset characteristics for experimental sets.

where R_t is the regret at iteration t, $\min_{i=1}^{t} f(x_i)$ is the global minimum for that dataset found by taking the minimum value across all benchmarks and methods for that dataset, and $f(x_{\text{best-so-far}})$ is the minimum value found so far. We use a log scale for regret values for easier visualization.

For the open-set experiments, the trained EVM is optimized to minimize the loss on a validation set of data containing samples from both known and unknown classes within the number of search trials listed in Table 2. This loss is defined as the negative $F1$ score with *weighted* averaging:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{5}$$

where TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively.

4.2 Experimental Setup

`HomOpt` can be used to augment any base HPO strategy. We thus compare the performance boost of `HomOpt` against the base performance of common popular hyperparameter optimization approaches: Random Search, Bayes, TPE, and SMAC. The numbers of iterations and samples used in the different optimization methods were chosen based on previous empirical experiments and theoretical considerations to balance the trade-off between the computational cost and the expected performance of the method. With TPE we used 20% of the results for the top-k mixture model seeded with 10 random search iterations and 10 generated candidates samples. In the Bayes examples, we used 20 random search iterations with 10 candidate samples. And for the SMAC runs, we used 20 random search iterations with 20 candidate samples. `HomOpt` similarly was initialized with 20 evaluations ($\mathcal{W}$ = 20) from each of the base strategies.

All experiments were run for 500 iterations across 5 separate seeds and averaged. The HPOBench classification experiments were run on a multitude of dataset tasks from OpenML (Vanschoren et al., 2014), which is open platform for sharing datasets, and the open-set experiments were conducted on the handwritten digits dataset MNIST (Deng, 2012) and Labeled Faced in the Wild (LFW) (Huang et al., 2007).

For the open-set experiments, we chose MNIST and LFW to consider two datasets of different complexity and modified each classification task to convert from a closed-set task to an open-set one. Table 2 summarizes the dataset experiments in terms of number of search trials, number of known and unknown classes, and number of training, validation, negatives, and testing samples. Negatives are samples from the unknown class without labels used to better inform the "1 vs. rest" classifiers when training the EVM. The number of trials for each experiment was determined and adjusted according to the time required to train a single model for the given dataset. If the dataset contained many images per class, fitting the EVM to the feature vectors for large samples required longer compute time and thus the number of search trials was reduced.

For the handwritten digits dataset MNIST, we designate handwritten digits 0 to 5 as the known classes and digits 6 to 9 as the unknown. Each image is represented as a flattened vector of the image pixels (784 features). For Labeled Faces in the Wild (LFW), we use classes with 30 or more face image samples to designate the known classes (34) and assign the remaining classes (5715) as the unknown set. The network used to extract features is an ArcFace (Deng et al., 2022) based feature extractor (Albiero et al., 2020) trained on the MS-Celeb-1M dataset (Guo et al., 2016) resulting in a 512 dimensional feature vector.

Threshold, cover threshold, and distance multiplier are sampled from a uniform distribution of range $[0, 1]$. The fitting algorithm for the EVM requires that the tailsize not exceed greater than half the number of training samples. For this reason, the tailsize hyperparameter was sampled from a uniform distribution between the range $[0, 0.5]$ and used as a multiplier to determine how many negative samples were included in estimating the model parameters.

Our default values for the parameters in the experiments were values found to be effective in empirical studies on simple problems and common choices found in literature. For all experiments, the distance threshold $\mathcal{D}$ is computed by the variance of the best 10% of the observed sample points scaled by 0.005 (also known as the jitter strength) as the local perturbation search around the observed minimum in Algorithm 1. In all the experiments we use 5 iterations which are the number of minimizations to compute the homotopy ($\mathcal{N}$). This strikes a balance between the computational costs and finding a good solution. Fewer iterations may not find a good solution, while more iterations may be computationally expensive without much benefit in terms of improved performance. We use a k of 0.5 which are the fraction of complete trials used to train one of the GAMs. Using half the completed trials provides a good balance between using enough data to train the model and not overfitting to the data. A smaller value of k would mean the GAM is trained on fewer data points, which may result in underfitting. A larger value of k would mean the GAM is trained on more data points, which may result in overfitting.The GAM model surrogates use a penalty term on the smooth functions of 10^{-4} and 25 splines for all experiments. The penalty term helps to control the complexity of the surrogate model. A smaller penalty term would result in a model that fits the data well but is likely to overfit. A larger penalty term would result in a model that is less likely to overfit but may not fit the data as well. Similarly, fewer splines would result in a simpler model that is less likely to overfit but may not fit the data as well, while more splines would result in a more complex model that fits the data well but is likely to overfit.

In our experiments, we have search domains that consist of both discrete and categorical hyperparameters. To perform optimization on these domains, we first cast each hyperparameter to its corresponding index in an array. Then, we convert these indices to floating-point numbers so that we can use continuous optimization algorithms. Finally, we round the output of the optimization algorithm to the nearest index to get the final hyperparameter configuration. This allows us to use continuous optimization algorithms on search domains that consist of discrete and categorical hyperparameters.

4.3 Results

4.3.1 HPOBench Classification Experiments

We compare the performance of `HomOpt` on the tuning of different sets of parameters on the SVM, Random Forest, Logistic Regression, MLP and XGBoost benchmarks from the HPOBench suite across multiple dataset tasks from the OpenML library. In all these experiments we use the same

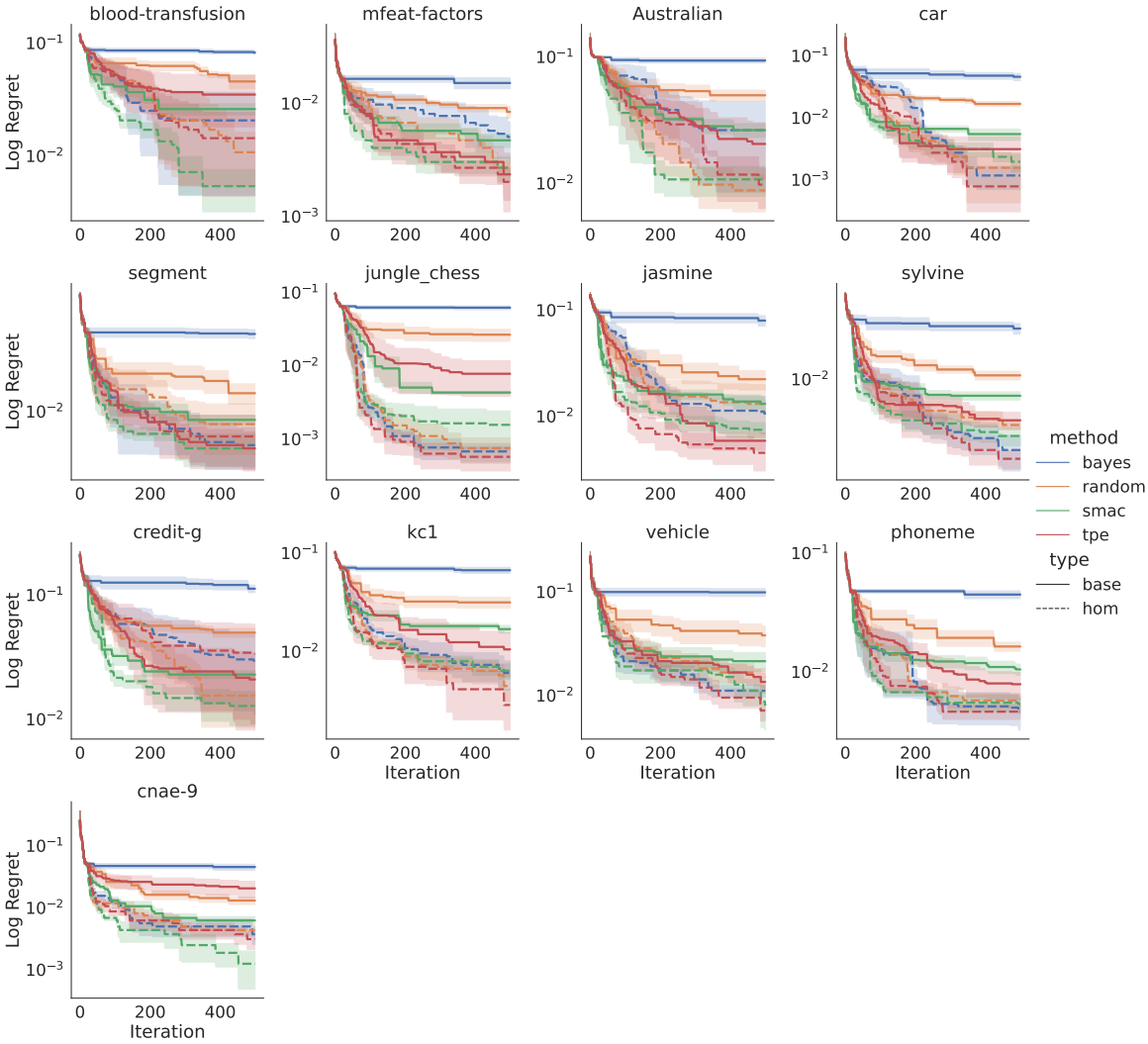


Figure 5: Comparison of all methods on 13 different tasks for the Random Forest Benchmark from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

Table 3: OpenML Task IDs used for HPOBench experiments. The table displays the total number of instances (combining the training and testing sets) (#obs) and the number of features prior to any preprocessing (#feat) for each dataset.

Name	TID	#obs	#feat
blood-transf..	10101	748	4
vehicle	53	846	18
Australian	146818	690	14
car	146821	1728	6
phoneme	9952	5404	5
segment	146822	2310	19
credit-g	31	1000	20
kc1	3917	2109	22
sylvine	168912	5124	20
kr-vs-kp	3	3196	36
jungle_che..	167119	44819	6
mfeat-factors	12	2000	216
shuttle	146212	58000	9
jasmine	168911	2984	145
cnae-9	9981	1080	856
numerai28.6	167120	96320	21
bank-mark..	14965	45211	16
higgs	146606	98050	28
adult	7592	48842	14
nomao	9977	34465	118

parameters for `HomOpt` across all benchmarks and datasets. Figures 5 - 9 reflect the regret at each iteration for all the ML benchmarks. The regret plots illustrate the difference between the performance at each iteration compared to the best value found for the entire dataset. Thus, the lower the regret, the better the optimization algorithm is performing. A summary of the performance on the best observed validation accuracy is additionally included in Appendix A.2.1 where we see an improvement in minimizing the validation loss for a majority of the datasets in 4 of the 5 benchmarks.

In the Random Forest benchmark shown in Figure 5, for each of the 13 datasets, we can see a consistent trend where `HomOpt` successfully improves the performance of all the base methods alone where the regret plots demonstrated a faster convergence to a better optima. The trials where `HomOpt` augmented the base methodologies are indicated with the dashed line which have steeper slope compared to the base methods alone (indicated by the solid lines). Each of the datasets had varying number of instances and features as indicated in Table 3. The most significant improvement of `HomOpt` over the base methods can actually be seen in the dataset with the highest number of features but relatively low number of instances (cnae-9). Random Forest has four hyperparameters (Table 4), and although the ranges may not be too wide, the combination and interaction of this

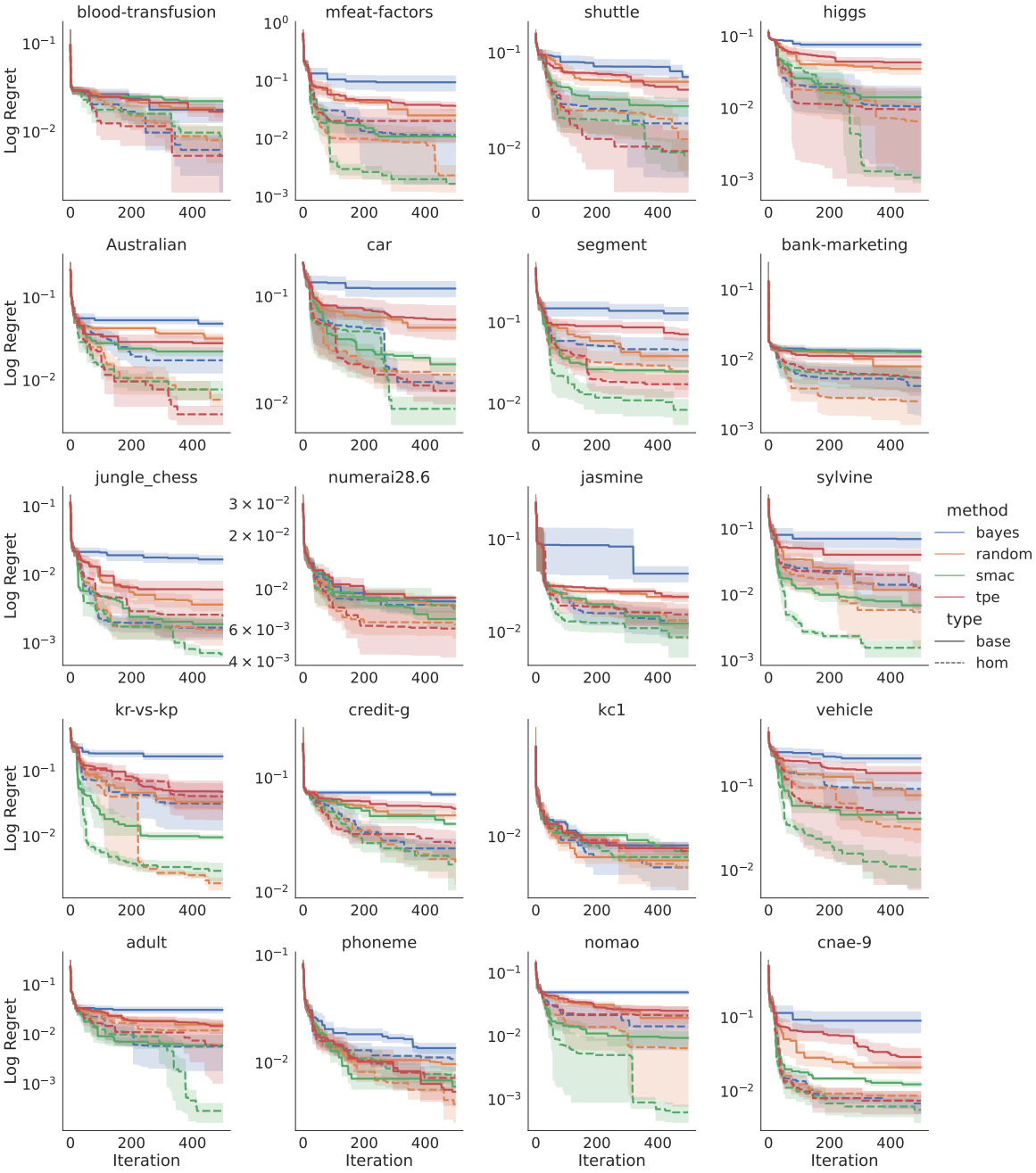


Figure 6: Comparison of all methods on 20 different tasks for the Logistic Regression Benchmark from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

Table 4: Configuration spaces for the benchmarks included in HPOBench, with hyperparameters and their ranges for each model.

Benchmark	Name	Range
SVM	C	$[2^{-10}, 2^{10}]$
	gamma	$[2^{-10}, 2^{10}]$
LogReg	alpha	[1e-05, 1.0]
	eta0	[1e-05, 1.0]
XGBoost	colsample_bytree	[0.1, 1.0]
	eta	$[2^{-10}, 1.0]$
	max_depth	[1, 50]
	reg_lambda	$[2^{-10}, 2^{10}]$
RandomForest	max_depth	[1, 50]
	max_features	[0.0, 1.0]
	min_samples_leaf	[1, 2]
	min_samples_split	[2, 128]
MLP	alpha	$[1.0e^{-08}, 1.0]$
	batch_size	[4, 256]
	depth	[1, 3]
	learning_rate_init	$[1.0e^{-05}, 1.0]$
	width	[16, 1024]

space can make it challenging. The min samples leaf and min samples split hyperparameters have relatively narrow ranges, but the interaction between these hyperparameters and max depth can make the search space challenging. For example, setting min samples leaf or min samples split too small can lead to overfitting, especially when max depth is also large. On the other hand, setting these hyperparameters too large can lead to underfitting, especially when max depth is small.

Compared to the Random Forest benchmark, the SVM benchmark illustrated in Figure 7 consists of only two hyperparameters, leading to a lower dimensionality in its search space. However, the search space for SVM is more extensive than that of the Random Forest benchmark, as both hyperparameters (C and gamma) have a range spanning from 2^{-10} to 2^{10}. These ranges are significantly large, covering several orders of magnitude, which poses a challenge when navigating through the search space. `HomOpt` demonstrates a relative boost over the base methodologies regarding in specific datasets such as the Australian, car, segment, jasmine, sylvine, and vehicle datasets. On the other hand, in datasets like blood-transfusion, mfeat-factors, credit-g, kc1, and cnae-9, the results across all methods, including those augmented with `HomOpt`, are relatively similar. The search space in these datasets may contain complex interactions between hyperparameters, making it difficult for all methods to find the optimal configurations. Consequently, the convergence to the optimum in these cases may not be as significant when compared to the base methodolo-

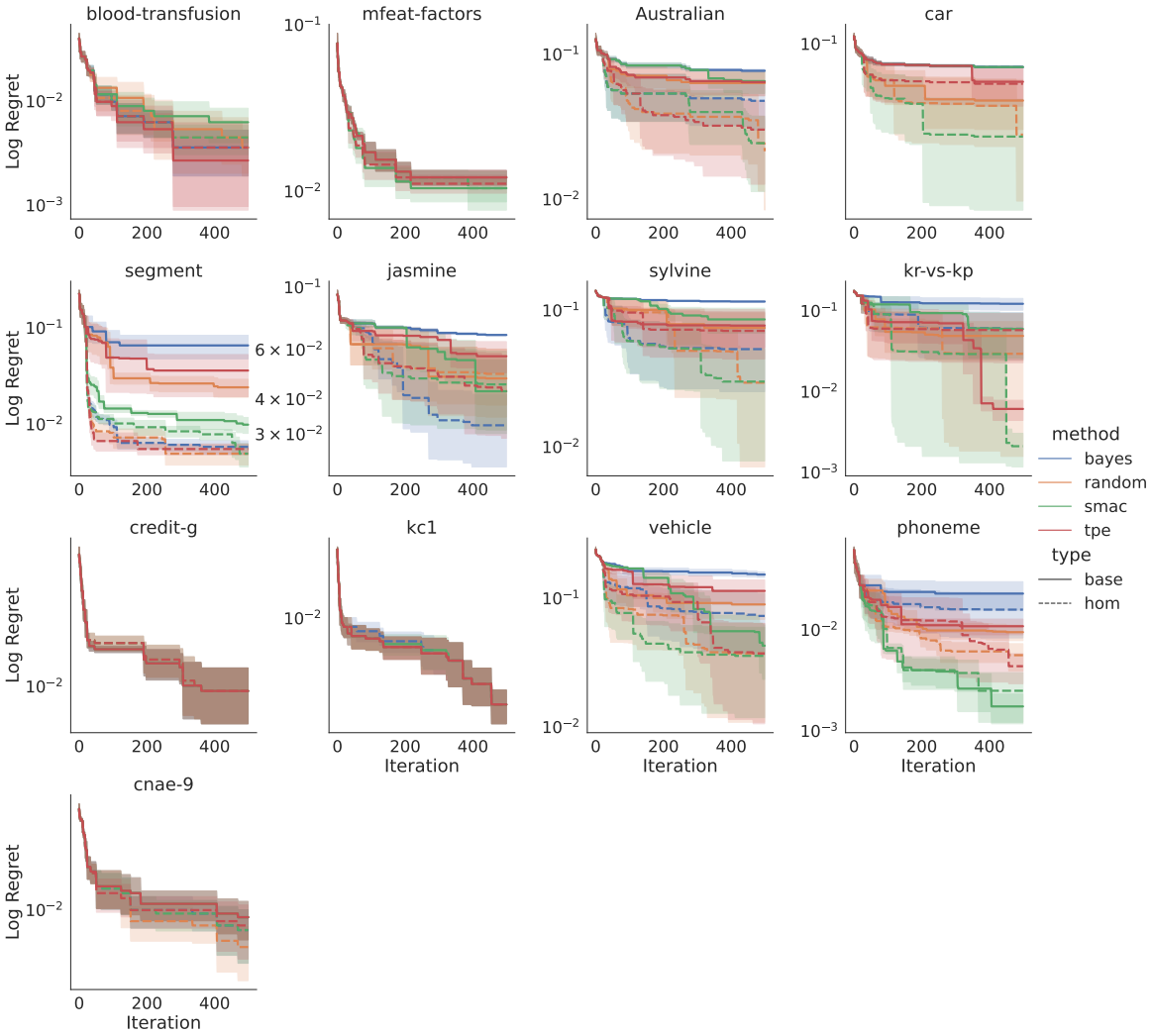


Figure 7: Comparison of all methods on 13 different tasks for the SVM Benchmark from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

gies for these datasets. While `HomOpt` with the default parameters demonstrates an overall boost in many of the datasets concerning the overall optima, the convergence to the optimum is not as pronounced. In several cases, the results are comparable or only marginally better than the base methodologies. This suggests that `HomOpt` might offer advantages in specific scenarios, but its performance may be influenced by a combination of factors, such as the inherent complexity of the datasets, the optimization algorithm's ability to navigate the search space, and the interactions between hyperparameters in the SVM benchmark.

The search space for the Logistic Regression benchmark is relatively straightforward, consisting of only two hyperparameters: alpha and eta0. These hyperparameters have a range of 1e-05 to 1.0, which is notably smaller compared to the broader ranges observed in SVM's hyperparameters.

This simpler search space allows for more manageable exploration and optimization, as it does not involve as many complex interactions between hyperparameters as seen in other benchmarks. The impact of `HomOpt` on the Logistic Regression benchmark (Figure 6) exhibits a trend similar to the results observed in the Random Forest benchmark. In both cases, the trials incorporating `HomOpt` demonstrate an overall improvement in performance across all 20 datasets. Additionally, `HomOpt` facilitates a faster convergence towards a lower optimal value, suggesting that the optimization algorithm is more efficient in navigating the search space and identifying better hyperparameter configurations than the base methodologies alone. This improved performance of `HomOpt` in the Logistic Regression benchmark can be attributed to several factors. The relatively simple search space, with fewer hyperparameters and a smaller range, enables a more effective exploration of possible configurations. Additionally, the inherently lower complexity of the Logistic Regression model, compared to models with larger search spaces or more hyperparameters, could also contribute to the improved performance observed when using `HomOpt`.

The MLP model has a larger search space with its five hyperparameters, leading to a greater number of possible configurations. The interactions between these hyperparameters are quite complex, further contributing to the challenge of optimizing this model. For instance, the depth and width of the network directly impact its capacity and complexity, while the alpha (L2 regularization) and learning rate init parameters control the model's generalization performance. Additionally, the batch size parameter influences not only the convergence speed but also the quality of the final solution. Navigating these intricate interactions within the MLP search space can be extremely challenging for HPO algorithms. Despite these challenges, `HomOpt` has demonstrated a significant boost in performance compared to the base methodologies in 5 different datasets (Figure 8). This improvement is evidenced by lower optimal scores and faster convergence rates. Furthermore, the standard error of the regret in the MLP benchmark is consistently lower than that of the SVM benchmark across the five repetitions. This suggests that the optimization process for MLP is more stable and less prone to random fluctuations compared to SVM. This could be attributed to the fact that the MLP search space is more constrained compared to SVM due to the smaller range of hyperparameters, resulting in a more focused search. This, in turn, could lead to a more stable optimization process with a more prominent boost in the performance.

XGBoost has four hyperparameters: colsample bytree, eta, max depth, and reg lambda. The search space complexity is higher than that of SVM and Logistic Regression due to the increased number of hyperparameters. The ranges of these hyperparameters are quite different, with some having smaller ranges (e.g., colsample bytree) and others having larger ranges (e.g., max depth and reg lambda). Among all five of the benchmarks from HPOBench, `HomOpt` demonstrated the least noticeable improvement in the XGBoost experiments. As seen by the regret plots in Figure 9 the results among all the methods appeared to have a similar effect in regards to the convergence and the overall optimum. We can see a small boost in performace for random and bayes with `HomOpt` in the blood-transfusion dataset which has only 4 features and 748 instances but otherwise a noticeable boost can not be seen. One possible reason why `HomOpt` did not perform as well on the XGBoost benchmark compared to the other benchmarks could be due to the interactions between the four hyperparameters. The interactions may be more intricate and complex compared to the other benchmarks, making it more challenging for `HomOpt` to effectively explore the search space and find the optimal solution. These hyperparameters are also interdependent with each other. For instance, a higher eta value may require a higher reg lambda value to counteract the increased learning rate. The ranges of the hyperparameters in XGBoost vary widely. For example, the colsample

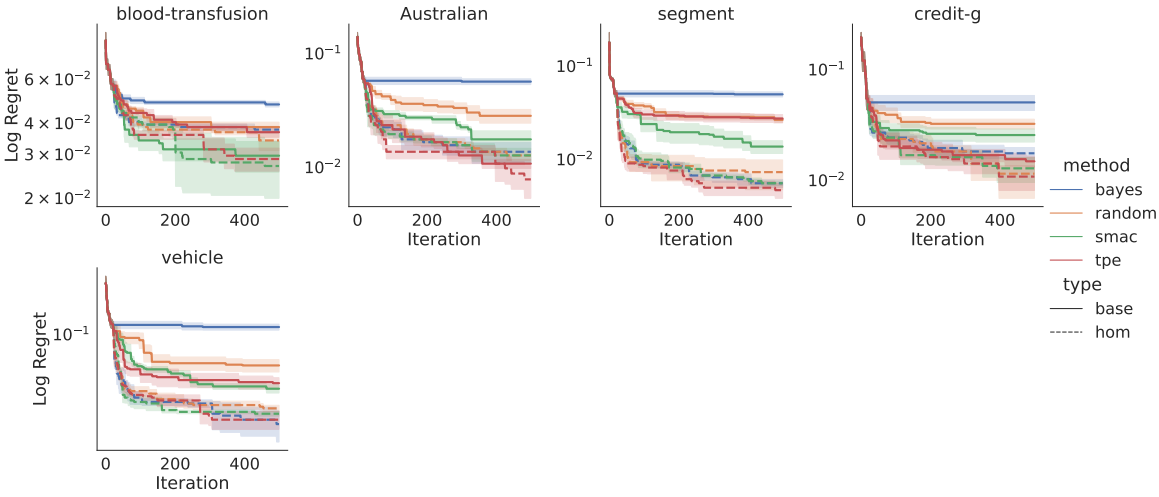


Figure 8: Comparison of all methods on 5 different tasks for the MLP Benchmark from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

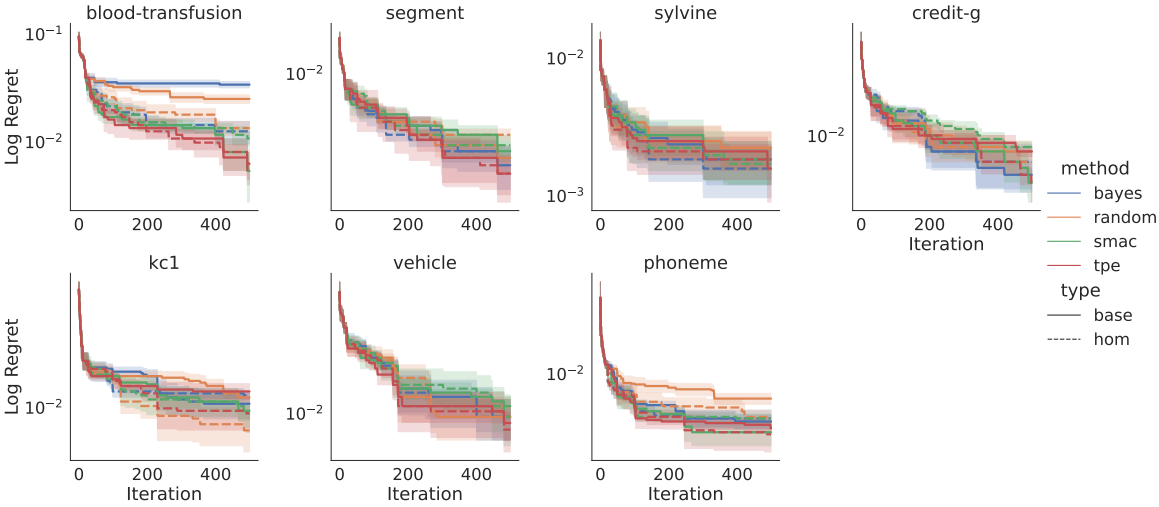


Figure 9: Comparison of all methods on 14 different tasks for the XGBoost Benchmark from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

bytree parameter has a range of [0.1, 1.0], while the max depth and reg lambda parameters have ranges of [1, 50] and $[2^{-10}, 2^{10}]$, respectively. This heterogeneity in the ranges could also contribute to the difficulty of exploring the search space efficiently.

Since `HomOpt` demonstrated minimal improvement on the XGBoost benchmark, we performed an additional study on one of the datasets to illustrate the sensitivity of parameters within `HomOpt` on the performance of the optimization. On dataset credit-g, we run `HomOpt` for each of the base

Variable	Description	Domain
$\mathcal{D}$	Local perturbation factor	[5, 5e-1, 5e-2, 5e-3, 5e-4, 5e-5]
$\mathcal{W}$	Number of warm-up samples	[10, 30, 50, 70, 90]
$\mathcal{N}$	Number of minimization steps to compute homotopy	[3, 6, 9]
k	Fraction of completed trials to train the GAMs	[0.2, 0.4, 0.6, 0.8, 1]

Table 5: `HomOpt` parameters and domain spaces for ablation study on XGBoost benchmark.

sampling methods (Bayes, Random, SMAC, TPE), with the method parameters and domains indicated in Table 5. This search was done over 100 iterations for a single seed for each combination of method parameters. The studies visualized in the contour plot (Figure 10) with SMAC samples, revealed that the performance of the optimization using `HomOpt` is sensitive to the choice of corresponding method parameters and domains. In our experiments our default method parameters use a k of 0.5, five iterations to compute the homotopy, a jitter strength of 0.005 and 20 warm up samples. The contour plot demonstrates how changes in these method parameter values can affect the optimization performance, where certain regions have higher performance (indicated by a lower loss value which are the blue regions). In this case, a higher k value of 0.6 and more warm samples (30+) resulted in higher performance for this dataset.

4.3.2 Open-Set Benchmarks

Optimizing the EVM model can be challenging due to the complexity of its hyperparameters and the high-dimensional nature of the data it handles. The EVM model has five hyperparameters, each with its own domain and range, which must be tuned to achieve optimal performance. The tailsize parameter, in particular, can be difficult to optimize, as it determines the number of negative samples used to estimate the model parameters. This parameter can have a significant impact on the model's performance, but it is also highly dependent on the characteristics of the input data. Additionally, the distance function parameter can be difficult to optimize because it affects the way the model computes distances between samples, and this can have a significant impact on the model's accuracy. Finding the right combination of hyperparameters to optimize the EVM model can be a time-consuming and challenging task for any HPO algorithm. In Figure 11, we evaluate the performance of `HomOpt` in tuning the 5 hyperparameters of an open-set recognition model, the EVM on two separate datasets over 5 random seeds. `HomOpt` again demonstrates a faster convergence to a better optimum value for both datasets, boosting all of the methods that were tested and also on average found a better objective value (observed $F1$ score). Compared to some of the other models in HPOBench, the EVM has a relatively small and constrained hyperparameter search space. The threshold, tailsize, cover threshold, and distance multiplier parameters all have ranges between 0 and 1, while the distance function is limited to two options: Cosine or Euclidean distance. This limited search space may partially explain why `HomOpt` was able to perform well on the EVM benchmark.

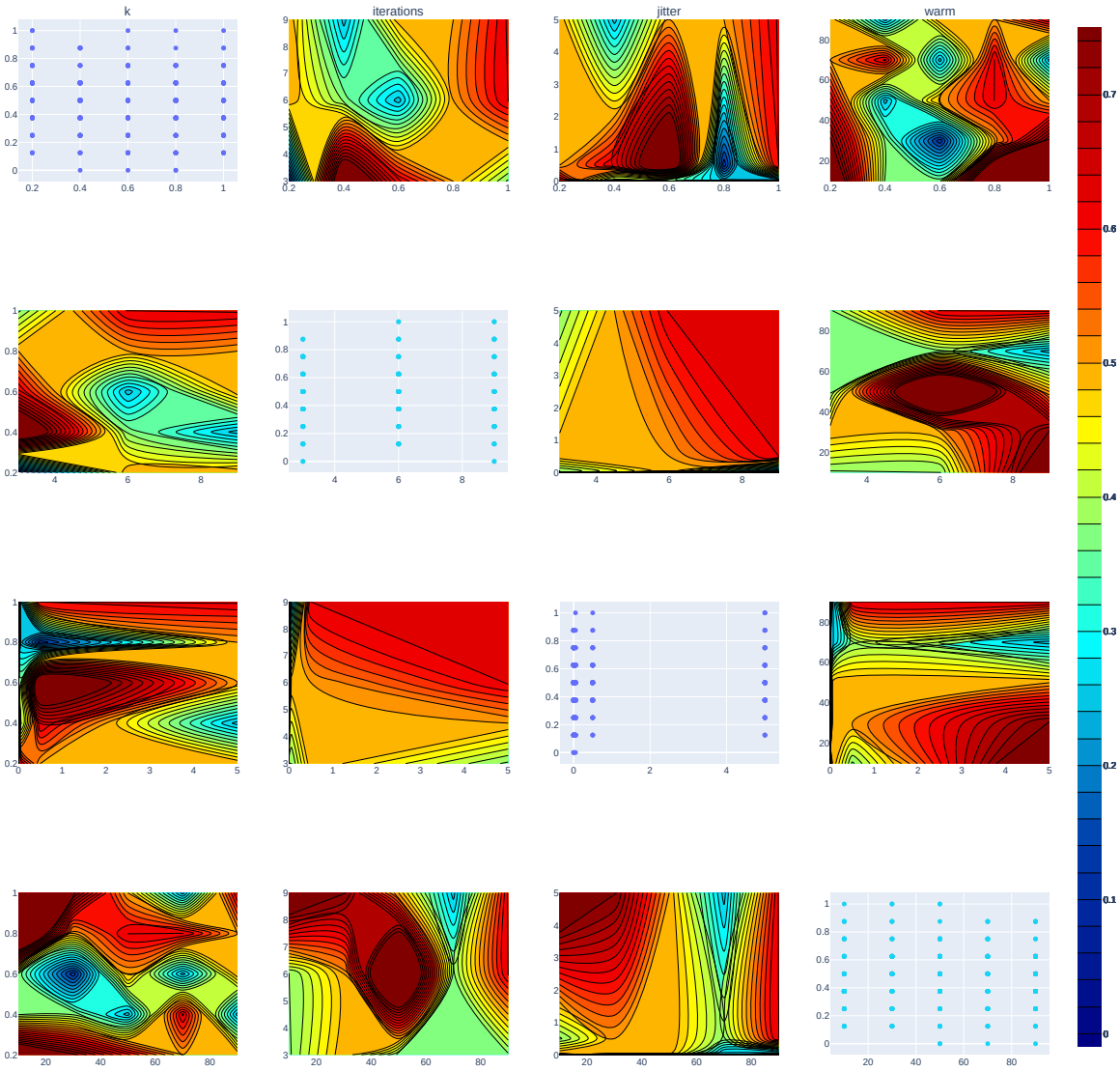


Figure 10: Contour plots visualizing the interactions between pairs of method parameters with SMAC samples on XGBoost sensitivity study. In this plot k represents the proportion of trial data used to train one of the GAMs, $iterations$ are the number of minimizations to compute the homotopy, $jitter$ refers to the distance threshold as the local perturbation search around the observed minimum and $warm$ represents the number of samples used for the surrogate approximation. Changes in the method parameter values can affect the optimization performance, where certain regions have higher performance (indicated by a lower loss value which are the blue regions)

5. Conclusion & Limitations

We introduced `HomOpt`, a hyperparameter optimization search strategy that builds a continuous deformation between GAM surrogate models and employs homotopy methods to track local minima along the deformation. `HomOpt` effectively identifies approximate regions of interest on the hyperparameter surface, allowing for efficient space exploration and faster convergence to an op-

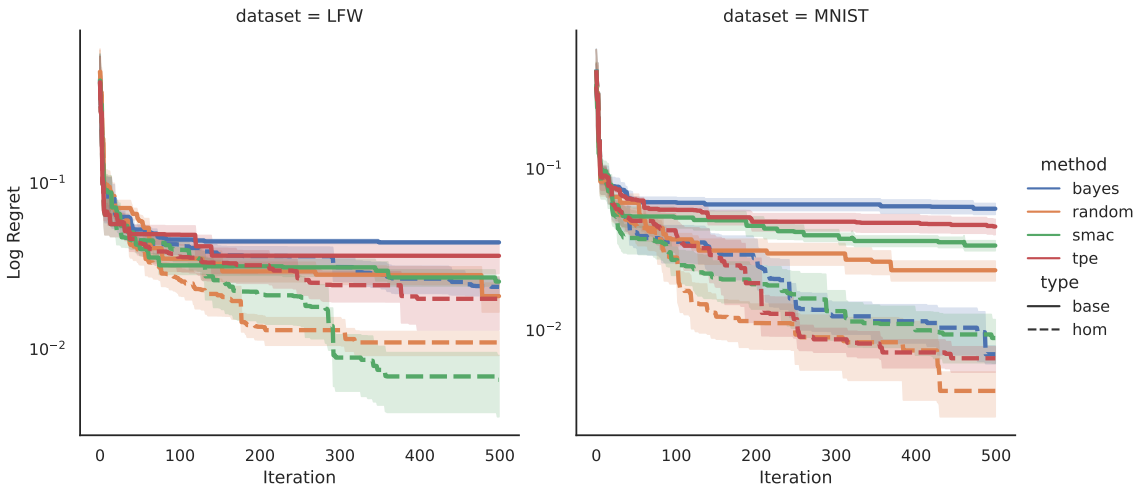


Figure 11: Comparison of all methods on MNIST and LFW for the open-set benchmarks. The mean and standard error of the regret at each iteration are displayed across 5 repetitions. `HomOpt` boosts the performance of all the methods for both of the datasets.

timal solution. We demonstrated `HomOpt`'s compatibility with various popular HPO methods, as it consistently speeds up their convergence and often finds better optima across numerous benchmarks. These benchmarks represent a wide range of challenging optimization problems, showcasing the rigor of our evaluation. Furthermore, `HomOpt` is adaptable to any HPO method and exhibits improved performance with faster convergence for both closed- and open-set models, without requiring strong assumptions about the objective. Future work will explore extending `HomOpt` to other optimization scenarios, incorporating additional surrogate models, and investigating its potential in multi-objective optimization problems. Ultimately, `HomOpt` holds promise in advancing HPO methods and facilitating the development of more robust and efficient models across various applications.

Although `HomOpt` demonstrated robust performance across multiple benchmarks, there are a few limitations to consider. For this work, we utilized a GAM as a surrogate model which in itself contains hyperparameters. In all the experiments, recall our GAM model surrogates use a penalty term on the smooth functions of 10^{-4} and 25 splines. `HomOpt` also has configurable parameters that are difficult to intuitively choose posing an additional challenge in driving the optimization. Potential ways to mitigate this would be to adaptively select all of the meta-parameters with methods such as reinforcement learning (Sutton and Barto, 2018). The choice of surrogate model may also affect the performance of the method. Although this remains outside of the scope of this paper, alternatives to the GAM include random forests (Breiman, 2001), Bayesian networks (Ghahramani, 2006), gradient boosting machines (Friedman, 2001), among many others (Bhosekar and Ierapetritou, 2018). However it is important to note that the GAM provides a more interpretable surrogate model against these suggested alternative approaches.

In the current configuration of `HomOpt`, only one optimal point is found. However, this can be modified by utilizing a different surrogate which tracks the change of multiple minima at spread out regions. This can also be modified by taking the number of points within the top $\alpha\%$ instead

of utilizing a single point. We also consider the adaption of `HomOpt` for other applications key to hyperparameter optimization such as incorporating domain knowledge. `HomOpt` provide a framework integrating the the use of homotopies to track the transition between minimums which can enable the incorporation of domain knowledge in the form of constraints or heuristics that better guide the optimization process.

Acknowledgements

This work was funded by DEVCOM Army Research Laboratory under cooperative agreement, W911NF-20-2-0218.

References

Vítor Albiero, Kai Zhang, and Kevin W Bowyer. How does gender balance in training data affect face recognition accuracy? In *2020 ieee international joint conference on biometrics (ijcb)*, pages 1–10. IEEE, 2020.

Eugene L. Allgower and Kurt Georg. *Numerical continuation methods*, volume 13 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1990. ISBN 3-540-12760-7. doi: 10.1007/978-3-642-61257-2. URL `https://doi-org.proxy.library.nd.edu/10.1007/978-3-642-61257-2`. An introduction.

Aravind Baskar, Mark Plecnik, and Jonathan D. Hauenstein. Computing saddle graphs via homotopy continuation for the approximate synthesis of mechanisms. *Mechanism and Machine Theory*, 176:104932, 2022.

Daniel J Bates, Andrew J Sommese, Jonathan D Hauenstein, and Charles W Wampler. *Numerically solving polynomial systems with Bertini*. SIAM, 2013.

Yoshua Bengio. Gradient-Based Optimization of Hyperparameters. *Neural Computation*, 12(8):1889–1900, 08 2000. ISSN 0899-7667. doi: 10.1162/089976600300015187. URL `https://doi.org/10.1162/089976600300015187`.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL `http://jmlr.org/papers/v13/bergstra12a.html`.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, jul 2015. doi: 10.1088/1749-4699/8/1/014008. URL `https://doi.org/10.1088/1749-4699/8/1/014008`.

Bruno Betrò. Bayesian methods in global optimization. *Operations Research '91*, page 16–18, 1992. doi: 10.1007/978-3-642-48417-9_6.

Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, 2018.

D.W. Boeringer and D.H. Werner. Efficiency-constrained particle swarm optimization of a modified bernstein polynomial for conformal array excitation amplitude synthesis. *IEEE Transactions on Antennas and Propagation*, 53(8):2662–2673, 2005. doi: 10.1109/TAP.2005.851783.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Qipin Chen and Wenrui Hao. A homotopy training algorithm for fully connected neural networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2231):20190662, 2019.

J. Chow, L. Udpa, and S.S. Udpa. Homotopy continuation methods for neural networks. In *1991., IEEE International Sympoisum on Circuits and Systems*, pages 2483–2486 vol.5, 1991. doi: 10.1109/ISCAS.1991.176030.

Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):5962–5979, 2022. doi: 10.1109/TPAMI.2021.3087709. URL https://doi.org/10.1109/TPAMI.2021.3087709.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Kai-Bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Proceedings of the 6th International Conference on Multiple Classifier Systems*, MCS'05, page 278–285, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3540263063. doi: 10.1007/11494683_28. URL https://doi.org/10.1007/11494683_28.

Katharina Eggensperger, F. Hutter, H. Hoos, and Kevin Leyton-Brown. Surrogate benchmarks for hyperparameter optimization: Semantic scholar, Jan 1970. URL https://www.semanticscholar.org/paper/Surrogate-Benchmarks-for-Hyperparameter-Eggensperger-Hutter/efe1f3273411272e467b72a7ad4fdeadfeb93bfe.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor H. Awad, Marius Lindauer, and Frank Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/93db85ed909c13838ff95ccfa94cebd9-Abstract-round2.html.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Zoubin Ghahramani. Learning dynamic bayesian networks. *Adaptive Processing of Sequences and Data Structures: International Summer School on Neural Networks "ER Caianiello" Vietri sul Mare, Salerno, Italy September 6–13, 1997 Tutorial Lectures*, pages 168–197, 2006.

Robert B. Gramacy and Herbert K. H. Lee. Cases for the nugget in modeling computer experiments. *Stat. Comput.*, 22(3):713–722, 2012.

Zachary A Griffin and Jonathan D Hauenstein. Real solutions to systems of polynomial equations and parameter continuation. *Advances in Geometry*, 15(2):173–187, 2015.

Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016.

Trevor J Hastie and Rob J Tibshirani. Generalized additive models, volume 43 of. *Monographs on statistics and applied probability*, 15, 1990.

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

Ilija Ilievski, Taimoor Akhtar, Jiashi Feng, and Christine Annette Shoemaker. Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates, 2017. URL `https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14312`.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

J. Kinnison, N. Kremer-Herman, D. Thain, and W. Scheirer. Shadho: Massively scalable hardware-aware distributed hyperparameter optimization. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 738–747, 2018. doi: 10.1109/WACV.2018.00086.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18 (1):6765–6816, 2017.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. `https://github.com/automl/SMAC3`, 2017.

Ilya Loshchilov and Frank Hutter. CMA-ES for hyperparameter optimization of deep neural networks. In *International Conference on Learning Representations (ICLR 2016). Workshop Track.*, pages 1–4, 2016.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning, 2015.

Mark McLeod, Michael A. Osborne, and Stephen J. Roberts. Optimization, fast and slow: optimally switching between local and bayesian optimization, 2018.

Dhagash Mehta, Tianran Chen, Tingting Tang, and Jonathan D. Hauenstein. The loss surface of deep linear networks viewed through the algebraic geometry lens. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5664–5680, 2022.

J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4): 308–313, 01 1965. ISSN 0010-4620. doi: 10.1093/comjnl/7.4.308. URL `https://doi.org/10.1093/comjnl/7.4.308`.

Robert Nisbet. *Handbook of statistical analysis and data mining applications*. Academic Press, an imprint of Elsevier, London, United Kingdom, second edition edition, 2018.

Harsh Nilesh Pathak. *Parameter Continuation with Secant Approximation for Deep Neural Networks*. PhD thesis, Worcester Polytechnic Institute, 2018.

Werner C. Rheinboldt. Numerical analysis of continuation methods for nonlinear structural problems. *Computers & Structures*, 13(1):103–113, 1981. ISSN 0045-7949. doi: https://doi.org/10.1016/0045-7949(81)90114-0. URL https://www.sciencedirect.com/science/article/pii/0045794981901140.

Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017.

Walter J. Scheirer, Anderson Rocha, Archana Sapkota, and Terrance E. Boult. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35:1757–1772, July 2012.

Daniel Servén, Charlie Brummitt, Hassan Abedi, and hlink. dswah/pygam: v0.8.0, October 2018. URL https://doi.org/10.5281/zenodo.1476122.

Andrew J. Sommese and Charles W. Wampler, II. *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005.

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Griewank function, 2013. URL https://www.sfu.ca/~ssurjano/griewank.html.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. ISSN 1674-862X. doi: https://doi.org/10.11989/JEST.1674-862X.80904120. URL https://www.sciencedirect.com/science/article/pii/S1674862X19300047.

Weicheng Xie, Wenting Chen, Linlin Shen, Jinming Duan, and Meng Yang. Surrogate network-based sparseness hyper-parameter optimization for deep expression recognition. *Pattern Recognition*, 111:107701, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107701. URL https://www.sciencedirect.com/science/article/pii/S0031320320305045.

Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on particle swarm optimization algorithm and its applications, Oct 2015. URL https://doi.org/10.1155/2015/931256.

Appendix A. Extended Results

A.1 EVM Results

Here we include additional results from the experiments. In the EVM experiments, we report the corresponding Normalized Mutual Information (NMI) score for each experiment which measures the similarity between the predicted label and the true class label as well.

$$\mathrm{NMI}(C,T) = \frac{2 \times I(C,T)}{H(C) + H(T)} \tag{6}$$

$$\mathrm{H}(C) = -\sum_{c} p(c) \log_2 p(c) \tag{7}$$

$$\mathrm{H}(T) = -\sum_{t} p(t) \log_2 p(t) \tag{8}$$

Table 6 reported the validation and testing score for MNIST data set experiments for open set recognition. In general, we can see the results for `HomOpt` are improved against all the base strategies random, TPE, Bayes, and SMAC. We can see consistent results (see Table 7) in the LFW experiment improves among all the base methods. It is noticeable that the highest validation and testing scores for MNIST data set achieves from the `HomOpt` with random seeds used for warm-up, while LFW data set achieves the highest scores from the `HomOpt` with SMAC seeds used for warm-up trials.

Method	Validation F1	Validation Accuracy	Validation NMI	Testing F1	Testing Accuracy	Testing NMI
Random	0.8982 ± 0.006	0.8958 ± 0.007	0.7547 ± 0.014	0.8988 ± 0.005	0.8969 ± 0.007	0.7579 ± 0.013
PSHHO+Random	$\mathbf{0.9160 \pm 0.003}$	$\mathbf{0.9127 \pm 0.004}$	$\mathbf{0.7889 \pm 0.008}$	$\mathbf{0.9039 \pm 0.006}$	$\mathbf{0.9022 \pm 0.007}$	$\mathbf{0.7657 \pm 0.012}$
TPE	0.8889 ± 0.011	0.8859 ± 0.011	0.7431 ± 0.022	0.8788 ± 0.004	0.8766 ± 0.004	0.7287 ± 0.009
PSHHO+TPE	0.9140 ± 0.003	0.9106 ± 0.004	0.7838 ± 0.005	0.8982 ± 0.007	0.8964 ± 0.006	0.7579 ± 0.012
Bayes	0.8771 ± 0.008	0.8749 ± 0.006	0.7200 ± 0.015	0.8669 ± 0.010	0.8655 ± 0.008	0.7054 ± 0.018
PSHHO+Bayes	0.9132 ± 0.002	0.9092 ± 0.003	0.7793 ± 0.003	0.8995 ± 0.003	0.8981 ± 0.004	0.7583 ± 0.004
SMAC	0.9034 ± 0.005	0.9001 ± 0.006	0.7650 ± 0.013	0.8924 ± 0.006	0.8907 ± 0.007	0.7468 ± 0.013
PSHHO+SMAC	0.9110 ± 0.006	0.9081 ± 0.005	0.7797 ± 0.009	0.8992 ± 0.004	0.8976 ± 0.005	0.7599 ± 0.008

Table 6: Average validation/testing score comparison for each base method and corresponding homotopy approach across 5 separate seeds for the EVM experiments with MNIST dataset.

Method	Validation F1	Validation Accuracy	Validation NMI	Testing F1	Testing Accuracy	Testing NMI
Random	0.8556 ± 0.009	0.9659 ± 0.004	0.8254 ± 0.019	0.8191 ± 0.058	0.9596 ± 0.028	0.7927 ± 0.011
PSHHO+Random	0.8654 ± 0.004	0.9673 ± 0.001	0.8309 ± 0.009	$\mathbf{0.8248 \pm 0.015}$	0.9610 ± 0.002	0.7981 ± 0.011
TPE	0.8400 ± 0.018	0.9618 ± 0.006	0.8060 ± 0.026	0.7992 ± 0.026	0.9554 ± 0.005	0.7729 ± 0.024
PSHHO+TPE	0.8563 ± 0.016	0.9632 ± 0.007	0.8140 ± 0.030	0.8175 ± 0.014	0.9571 ± 0.005	0.7844 ± 0.023
Bayes	0.8426 ± 0.007	0.9617 ± 0.002	0.8049 ± 0.010	0.7919 ± 0.013	0.9542 ± 0.003	0.7657 ± 0.014
PSHHO+Bayes	0.8622 ± 0.008	0.9671 ± 0.003	0.8288 ± 0.016	0.8161 ± 0.015	0.9589 ± 0.003	0.78855 ± 0.015
SMAC	0.8510 ± 0.004	0.9657 ± 0.003	0.8241 ± 0.014	0.8218 ± 0.016	0.9602 ± 0.003	0.7951 ± 0.003
PSHHO+SMAC	$\mathbf{0.8698 \pm 0.005}$	$\mathbf{0.9699 \pm 0.001}$	$\mathbf{0.8433 \pm 0.007}$	0.8231 ± 0.010	$\mathbf{0.9613 \pm 0.002}$	$\mathbf{0.8001 \pm 0.009}$

Table 7: Average validation/testing scores comparison for each base method and corresponding homotopy approach across 5 separate seeds for the EVM experiment with LFW dataset.

A.2 HPOBench

In this section, we present the additional results for the HPOBench experiments, specifically focusing on the validation and test scores. The validation and test scores showcase the average percent improvement and standard error achieved over 5 trials for each dataset experiment within the benchmark.

A.2.1 Validations Scores

Table 8: **Average percent improvement and standard error for the best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the SVM Benchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**5.22** ± 1.63	**4.35** ± 2.38	**8.33** ± 1.32	**5.22** ± 2.54
12	**1.05** ± 0.43	**1.05** ± 0.43	**1.63** ± 0.94	**1.40** ± 0.47
146818	**29.52** ± 17.26	**55.24** ± 12.74	**52.38** ± 12.78	-40.00 ± 44.97
146821	-3.24 ± 1.01	**54.29** ± 26.30	**57.84** ± 25.82	**16.76** ± 20.82
146822	**67.24** ± 1.09	**40.65** ± 2.62	-8.24 ± 5.46	**47.78** ± 1.36
168911	**21.87** ± 5.96	-40.45 ± 11.20	**3.23** ± 10.08	**15.11** ± 8.84
168912	**38.63** ± 16.36	**45.12** ± 15.89	**47.43** ± 14.96	**27.10** ± 16.45
3	**56.34** ± 25.81	**9.68** ± 85.50	**98.47** ± 0.68	-587.50 ± 416.03
31	-10.00 ± 5.39	-10.00 ± 5.39	-10.00 ± 5.39	-10.00 ± 5.39
3917	**2.50** ± 1.67	**2.50** ± 1.67	**2.50** ± 1.67	**2.50** ± 1.67
53	**41.54** ± 17.77	**28.67** ± 22.20	**0.00** ± 28.69	-33.75 ± 42.09
9952	-14.86 ± 17.23	-1.59 ± 4.29	-1.72 ± 3.53	-1.64 ± 3.77
9981	-2.75 ± 1.27	-2.00 ± 0.94	-2.75 ± 1.27	-3.00 ± 1.40

Table 9: **Average percent improvement and standard error for the best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the Random Forest benchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**35.12** ± 8.78	**28.24** ± 3.55	**17.14** ± 1.75	-5.00 ± 9.26
12	**60.00** ± 10.95	**35.00** ± 6.12	**48.00** ± 8.00	-120.00 ± 37.42
146818	**61.82** ± 16.61	**63.08** ± 4.49	**42.22** ± 6.48	**16.67** ± 11.79
146821	**93.04** ± 1.06	**80.00** ± 6.48	**0.00** ± 15.81	-40.00 ± 24.49
146822	**69.70** ± 3.95	**38.95** ± 11.36	**24.62** ± 7.46	**3.64** ± 16.91
167119	**46.36** ± 0.15	**34.72** ± 0.27	**3.99** ± 1.19	**0.59** ± 0.30
168911	**54.29** ± 2.81	**22.32** ± 8.68	**13.21** ± 2.00	-0.00 ± 2.83
168912	**73.75** ± 2.00	**40.00** ± 1.17	**28.00** ± 3.58	**14.74** ± 3.07
31	**45.71** ± 13.24	**38.00** ± 6.80	**15.24** ± 5.51	-41.18 ± 42.78
3917	**61.61** ± 2.00	**49.33** ± 1.91	**29.41** ± 2.08	**24.83** ± 2.76
53	**64.44** ± 1.04	**42.50** ± 3.33	-9.09 ± 7.61	**1.67** ± 3.12
9952	**63.68** ± 1.95	**29.18** ± 2.22	**25.26** ± 2.58	**1.43** ± 2.45
9981	**81.67** ± 3.12	**40.00** ± 6.12	**30.00** ± 12.25	**60.00** ± 6.32

Table 10: **Average percent improvement and standard error for the best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the Logistic Regression Benchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**9.41** ± 1.44	**6.40** ± 1.60	**4.08** ± 1.12	**7.60** ± 1.47
12	**78.79** ± 18.21	**90.00** ± 4.84	**75.00** ± 0.00	**35.79** ± 37.19
146212	**39.34** ± 15.76	**46.21** ± 7.74	**37.63** ± 5.04	**44.88** ± 7.69
146606	**17.02** ± 2.16	**4.57** ± 1.40	**6.01** ± 0.04	**5.94** ± 2.32
146818	**16.88** ± 3.37	**19.33** ± 0.67	**12.14** ± 1.43	**20.67** ± 0.67
146821	**29.49** ± 1.67	**27.44** ± 3.82	**16.77** ± 2.14	-3.46 ± 3.24
146822	**7.93** ± 10.59	**1.91** ± 7.37	**9.21** ± 2.23	**25.00** ± 3.17
14965	**4.33** ± 2.44	**4.82** ± 1.30	**2.91** ± 2.01	**3.03** ± 2.28
167119	**2.96** ± 0.17	**1.04** ± 0.20	**0.67** ± 0.02	**2.73** ± 0.37
167120	-0.05 ± 0.13	**0.38** ± 0.21	-0.45 ± 0.21	**0.66** ± 0.39
168911	**19.55** ± 1.75	**3.85** ± 2.27	-1.62 ± 1.77	-0.44 ± 2.24
168912	**25.96** ± 6.75	**4.63** ± 4.89	**8.83** ± 0.57	**9.50** ± 8.18
3	**69.43** ± 10.67	**67.37** ± 0.49	**19.37** ± 2.69	**49.68** ± 17.90
31	**17.25** ± 1.87	**10.56** ± 2.73	**8.29** ± 2.98	**9.46** ± 2.93
3917	**0.92** ± 1.23	-0.23 ± 1.24	**1.36** ± 0.23	-1.40 ± 0.23
53	**17.78** ± 14.46	**18.06** ± 3.86	**20.00** ± 1.81	**31.85** ± 13.24
7592	**7.64** ± 2.28	**3.89** ± 3.50	**4.28** ± 0.08	-1.36 ± 3.21
9952	**2.13** ± 1.08	**2.98** ± 0.61	-0.35 ± 0.63	-3.18 ± 0.31
9977	**24.91** ± 9.65	**20.36** ± 8.15	**25.20** ± 0.29	**7.44** ± 11.08
9981	**83.08** ± 4.49	**30.00** ± 9.35	**55.00** ± 14.58	**65.71** ± 11.61

Table 11: **Average percent improvement and standard error for the best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the MLP Benchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**3.72** ± 0.57	**0.98** ± 1.65	**7.14** ± 3.69	**3.90** ± 1.65
146818	**57.50** ± 3.64	**17.50** ± 3.06	-32.00 ± 8.00	**20.00** ± 7.28
146822	**59.22** ± 0.73	**43.59** ± 4.51	**30.67** ± 0.67	**50.73** ± 1.42
31	**40.00** ± 2.07	**35.56** ± 5.05	**25.83** ± 4.04	**4.44** ± 4.44
53	**78.46** ± 5.10	**44.62** ± 2.88	**40.00** ± 2.23	**40.00** ± 8.37

Table 12: **Average percent improvement and standard error for the best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the XGBoost Benchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**17.33** $\pm$ 2.21	**13.79** $\pm$ 1.54	-10.91 $\pm$ 1.82	-5.45 $\pm$ 0.91
146822	-11.43 $\pm$ 5.35	-22.86 $\pm$ 3.50	-17.14 $\pm$ 5.35	-5.71 $\pm$ 5.71
168912	-2.67 $\pm$ 6.18	-9.33 $\pm$ 6.53	-11.43 $\pm$ 5.35	**1.25** $\pm$ 5.00
31	-5.00 $\pm$ 4.59	**6.67** $\pm$ 2.08	-7.50 $\pm$ 1.25	**10.00** $\pm$ 2.08
3917	-27.00 $\pm$ 2.00	**9.60** $\pm$ 4.66	-8.18 $\pm$ 4.64	-3.48 $\pm$ 5.74
53	**5.45** $\pm$ 6.17	-8.00 $\pm$ 3.74	-4.00 $\pm$ 8.12	**0.00** $\pm$ 5.48
9952	-1.28 $\pm$ 2.48	**6.27** $\pm$ 1.90	-1.28 $\pm$ 1.28	**0.43** $\pm$ 3.10

Table 13: **Average percent improvement and standard error for the corresponding test loss (1-accuracy) at best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the RandomForestBenchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

method dataset	**bayes**	**random**	**smac**	**tpe**
10101	-11.76 ± 3.22	-2.22 ± 5.98	-13.33 ± 2.22	-9.47 ± 5.10
12	**11.11** ± 6.09	**2.86** ± 5.35	-11.43 ± 5.35	-36.00 ± 9.80
146818	-8.57 ± 5.71	**20.00** ± 8.89	-0.00 ± 10.46	**7.50** ± 9.35
146821	**56.92** ± 5.76	**4.00** ± 16.00	-26.67 ± 26.67	-86.67 ± 34.32
146822	-15.29 ± 4.40	-32.86 ± 8.63	**5.88** ± 6.17	-35.71 ± 3.91
167119	-16.40 ± 1.16	-7.15 ± 0.82	**0.27** ± 0.85	-0.94 ± 0.89
168911	**2.76** ± 2.01	-5.00 ± 1.91	-3.21 ± 1.54	**2.86** ± 1.34
168912	-0.54 ± 3.35	**2.11** ± 3.16	-1.11 ± 4.94	-1.11 ± 1.67
31	**15.17** ± 5.52	**15.20** ± 5.28	**12.80** ± 10.23	**15.56** ± 3.95
3917	**2.76** ± 4.28	**8.97** ± 3.71	**5.33** ± 3.27	-3.08 ± 4.93
53	**10.43** ± 3.53	**8.70** ± 5.99	**11.82** ± 1.82	-6.32 ± 6.09
9952	**22.00** ± 4.25	-6.27 ± 2.93	**3.16** ± 4.35	**16.77** ± 2.42
9981	**23.33** ± 12.47	**3.33** ± 9.72	-120.00 ± 37.42	**4.00** ± 9.80

A.2.2 TEST SCORES

Table 14: **Average percent improvement and standard error for the corresponding test loss (1-accuracy) at best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the SVMBenchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

method dataset	**bayes**	**random**	**smac**	**tpe**
10101	-3.75 ± 2.50	-5.00 ± 2.34	**2.35** ± 2.35	-5.00 ± 2.34
12	**0.22** ± 0.14	**0.22** ± 0.14	**4.11** ± 3.84	**1.44** ± 1.31
146818	**32.26** ± 19.75	**62.58** ± 15.72	**64.52** ± 16.42	-105.00 ± 74.96
146821	**0.00** ± 0.00	**27.33** ± 41.12	**59.23** ± 24.18	**18.46** ± 18.46
146822	**57.08** ± 1.06	-5.71 ± 5.91	**17.50** ± 5.17	**53.06** ± 2.41
168911	**33.15** ± 8.37	-71.51 ± 20.25	-9.64 ± 16.05	**25.37** ± 11.27
168912	**36.48** ± 15.05	**35.40** ± 16.11	**37.20** ± 15.49	**23.12** ± 14.28
3	**56.34** ± 23.00	-31.72 ± 99.04	**94.25** ± 0.48	-1224.00 ± 708.84
31	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00
3917	-0.00 ± 0.96	-0.00 ± 0.96	-0.00 ± 0.96	-0.00 ± 0.96
53	**37.46** ± 14.15	-2.86 ± 24.42	-23.33 ± 22.92	-62.22 ± 35.94
9952	-16.94 ± 22.46	-15.25 ± 6.72	**9.41** ± 2.96	**0.60** ± 8.32
9981	-0.65 ± 0.43	-1.09 ± 0.69	-1.54 ± 0.56	-0.65 ± 0.55

Table 15: **Average percent improvement and standard error for the corresponding test loss (1-accuracy) at best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the LRBenchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

method dataset	**bayes**	**random**	**smac**	**tpe**
10101	-4.44 ± 1.11	-6.67 ± 2.08	**5.26** ± 2.35	**8.00** ± 2.55
12	**64.71** ± 11.91	**36.00** ± 7.48	**43.33** ± 6.67	**52.50** ± 17.74
146212	**39.01** ± 15.89	**45.74** ± 7.84	**36.41** ± 5.63	**44.18** ± 7.26
146606	**16.73** ± 1.74	**3.42** ± 1.37	**5.84** ± 0.12	**5.89** ± 1.97
146818	-8.57 ± 16.66	**20.00** ± 7.07	**11.11** ± 11.65	-33.33 ± 10.54
146821	**30.34** ± 4.55	**29.66** ± 3.55	**18.46** ± 5.63	-4.55 ± 3.80
146822	**22.45** ± 8.09	-2.86 ± 9.52	-16.30 ± 1.89	**36.73** ± 2.66
14965	**2.85** ± 1.97	-0.99 ± 1.64	**1.65** ± 2.42	**1.03** ± 2.29
167119	**4.02** ± 0.34	-0.25 ± 0.43	**1.17** ± 0.19	**3.58** ± 0.43
167120	**0.30** ± 0.20	**0.96** ± 0.14	**0.58** ± 0.48	**1.16** ± 0.62
168911	**4.51** ± 1.29	-8.57 ± 1.19	-8.20 ± 1.16	-2.46 ± 2.73
168912	**20.79** ± 5.17	**1.11** ± 4.25	**1.13** ± 0.96	**13.82** ± 5.36
3	**67.89** ± 12.47	**57.78** ± 3.23	**33.33** ± 4.22	**46.15** ± 18.84
31	**10.00** ± 6.43	**10.71** ± 7.41	**12.86** ± 4.16	**19.37** ± 4.98
3917	**1.88** ± 2.90	-14.29 ± 2.53	**7.27** ± 0.74	**1.88** ± 2.90
53	**0.87** ± 17.25	**24.17** ± 6.37	**5.56** ± 3.04	**18.26** ± 12.33
7592	**9.55** ± 2.40	**5.15** ± 4.00	**1.66** ± 0.28	-1.59 ± 3.88
9952	**0.32** ± 1.50	-0.16 ± 0.60	-2.58 ± 0.16	-2.15 ± 1.00
9977	**20.85** ± 8.66	**21.50** ± 7.28	**18.12** ± 0.63	**7.77** ± 9.79
9981	-3.33 ± 20.68	-30.00 ± 14.58	**10.00** ± 11.30	**5.71** ± 16.66

Table 16: **Average percent improvement and standard error for the corresponding test loss (1-accuracy) at best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the NNBenchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

method dataset	**bayes**	**random**	**smac**	**tpe**
10101	**0.00** ± 1.76	**1.05** ± 1.97	-0.00 ± 2.88	-3.33 ± 2.22
146818	-57.14 ± 10.10	-25.00 ± 11.18	**1.82** ± 6.68	-32.50 ± 9.35
146822	**39.13** ± 4.96	**20.00** ± 11.42	**12.00** ± 4.90	**30.59** ± 6.81
31	**3.85** ± 2.43	**8.46** ± 3.73	-3.48 ± 1.63	**9.23** ± 4.65
53	**21.54** ± 7.46	**21.33** ± 3.27	**20.00** ± 4.71	**1.54** ± 9.23

Table 17: **Average percent improvement and standard error for the corresponding test loss (1-accuracy) at best observed loss over five trials of `HomOpt` over the base methodology for each dataset and method in the XGBoostBenchmark. Bold values indicate where `HomOpt` outperforms the base methodology.**

method dataset	**bayes**	**random**	**smac**	**tpe**
10101	-30.67 ± 5.42	-8.89 ± 4.16	-13.33 ± 4.16	**7.62** ± 6.67
146818	**13.33** ± 8.89	-11.43 ± 12.29	-20.00 ± 7.28	-64.00 ± 21.35
146822	-5.33 ± 3.89	-2.67 ± 4.52	-15.71 ± 6.93	**2.35** ± 6.06
168912	**1.62** ± 2.51	-10.29 ± 1.46	**8.72** ± 2.08	-3.53 ± 4.78
31	**11.67** ± 6.77	-2.50 ± 4.08	-17.14 ± 2.43	**15.71** ± 2.90
3917	-32.50 ± 2.43	-2.00 ± 5.44	-33.33 ± 2.95	-27.50 ± 8.50
53	-15.29 ± 8.44	-12.63 ± 6.14	-10.00 ± 3.24	-7.78 ± 2.83
9952	**3.51** ± 3.80	**10.82** ± 1.23	**4.56** ± 1.43	-4.29 ± 1.07