Meta Hyperparameter Optimization with Adversarial Proxy Subsets Sampling

Yue Liu
liuyue17@mails.tsinghua.edu.cn
Tsinghua University

Xin Wang*
xin_wang@tsinghua.edu.cn
Tsinghua University
Pengcheng Laboratory

Xue Xu
xuexu@mail.ustc.edu.cn
Tencent

Jianbo Yang
jianboyang@tencent.com
Tencent

Wenwu Zhu*
wwzhu@tsinghua.edu.cn
Tsinghua University
Pengcheng Laboratory

ABSTRACT

Hyperparameter optimization (HPO), aiming at automatically searching optimal hyperparameter configurations, has attracted increasing attention in the machine learning community. HPO generally suffers from high searching costs when dealing with large-scale real-world datasets since training the model with a certain hyperparameter configuration is time-consuming. Existing works suggest sampling subsets uniformly to represent the full dataset for HPO but ignoring the complex and dynamic distribution in real-world scenarios and the exploration of hyperparameter transfer. To tackle this problem, we propose a novel meta hyperparameter optimization model with an adversarial proxy subsets sampling strategy (Meta-HPO), which can transfer hyperparameters optimized on the sampled proxy subsets to the full dataset and further adapt to the new data in an out-of-sample updating manner. In particular, a perturbation-aware adversarial sampling strategy is designed to select the proxy subsets that significantly influence the model performance. With the searched hyperparameter configurations and corresponding performance scores on the proxy subsets, we propose a meta transfer framework, named "hp-learner", to build the connection between the distribution of dataset and the optimal hyperparameter configuration. Our Meta-HPO provides a flexible and efficient hyperparameter optimization algorithm. Extensive experiments on real-world datasets validate the advantages of our proposed Meta-HPO model against existing state-of-the-art benchmarks.

CCS CONCEPTS

• **Computing methodologies** → *Search methodologies*; **Learning settings**; *Learning paradigms.*

*Corresponding authors.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482368

KEYWORDS

Hyperparameter optimization; meta-learning; transfer learning; adversarial sampling strategy

ACM Reference Format:
Yue Liu, Xin Wang, Xue Xu, Jianbo Yang, and Wenwu Zhu. 2021. Meta Hyperparameter Optimization with Adversarial Proxy Subsets Sampling. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482368

1 INTRODUCTION

Machine learning has achieved considerable success in various applications such as computer vision, natural language processing and data mining. The performance of a machine learning model largely depends on the pre-defined hyperparameters, such as the number of neurons in each layer, the initial learning rate and the choice of operators in the deep neural networks (DNNs). Automated Machine learning (AutoML) provides new ways of automatically discovering the well-performed DNN architectures and hyperparameter configurations instead of manual designs [9, 11, 28, 29]. As such, hyperparameter optimization (HPO), aiming to find the optimal hyperparameters through learning a black-box function mapping from hyperparameters to their corresponding performances, has attracted an increasing amount of research interest recently.

In practice, training machine learning models on large-scale industrial data with a given hyperparameter configuration is quite time-consuming, thus limiting the possibilities of searching different sets of hyperparameter configurations on the original large dataset. One natural way to reduce the learning cost is sampling subsets from the original dataset and searching for the best hyperparameter configuration for each subset before transferring them to the original dataset. Klein *et al.* [16] suggest sampling subsets randomly from the full dataset and optimizing the hyperparameters as well as the size of sampled subsets jointly at each iteration. Besides, existing works on hyperparameter transfer across multi-tasks use Gaussian Process to represent relationships between multiple datasets implicitly [25–27, 33], or uses DNNs to obtain good model initialization on the target dataset by taking historical trials learned from previous datasets into account [30].

However, these existing works suffer from the following weaknesses, which are challenging to address.

- The subsets are sampled uniformly, ignoring the fact that correctly selecting data points important for model training can help to search the optimal hyperparameter configurations on the subsets and in turn benefit the optimal hyperparameter discovering on the original dataset.
- The hyperparameter transfer through Gaussian Process suffers from high computational complexity (cubic) when fitting Gaussian Process, while hyperparameter transfer through neural networks fails to take the meta-information of each subset into consideration when conducting the transfer.
- They do not consider the fact that data distributions in real-world applications tend to be complex and dynamic, such that the optimal hyperparameters do not always remain the same due to the concept drift issue in newly-coming data.

To tackle these challenges, we propose a meta hyperparameter optimization model with adversarial proxy subsets sampling strategy (Meta-HPO), which is able to transfer hyperparameters optimized on the sampled proxy subsets to the full original dataset and further adapt to new data in an out-of-sample updating manner. Our proposed Meta-HPO model aims to select an appropriate group of data points important for learning well-performed hyperparameters as the proxy subsets. The training on the proxy subsets speeds up the hyperparameter selection compared with the training on the whole dataset. Specifically, we design a perturbation-aware adversarial sampling strategy to sample proxy subsets with large impacts on the estimated model performance. The adversarial sampling strategy tends to sample data points around the classification boundary such that the sampled subsets discriminatively represent the full dataset. Practically, the proposed sampling strategy significantly improves the quality of subsets by combining the adversarial sampling strategy with random sampling to construct more representative subsets of the full dataset.

With sampled proxy subsets, our Meta-HPO model searches for different hyperparameter configurations on each subset and then transfers these hyperparameters to the full large-scale dataset through a meta-learning-based transfer framework, named "hp-learner". Concretely, we learn a non-linear function capable of mapping the meta features together with the given hyperparameters to the corresponding model performance. Moreover, the meta transfer module facilitates a better hyperparameter selection for out-of-sample data since we can obtain the meta features for newly-arriving data and then forecast a well-performed hyperparameter configuration by the hp-learner.

In the training of the hp-learner, the size of training samples is limited due to the cost of obtaining a large number of hyperparameters on the proxy subsets. To avoid over-fitting to the selected hyperparameters, we apply gradient-based meta-learning paradigm to the training of hp-learner, motivated by the model-agnostic meta-learning (MAML) algorithm [8]. Once the hp-learner is trained, it is able to predict well-performed hyperparameter configurations on the full dataset. Furthermore, the meta transfer module can also adapt to out-of-sample scenarios with streaming data by forecasting hyperparameter configuration with the extracted meta feature from newly-arriving data.

The contributions of this work can be summarized as follows:

- We propose Meta-HPO, a novel meta hyperparameter optimization method that transfers hyperparameters learned from proxy subsets towards the full dataset and new out-of-sample data to accelerate the HPO process in the real-world scenarios.
- We design a perturbation-aware adversarial sampling strategy to select proxy subsets whose data points have significant impacts on the model performance, thus improving the efficacy of subsets and enhancing the transfer of hyperparameters.
- We propose a meta-learning-based transfer mechanism to build the connection between the distribution of dataset and the selection of hyperparameter configuration. The meta transfer module, hp-learner, can predict well-performed hyperparameters on the full dataset and additionally deal with new data in an out-of-sample manner.
- We conduct extensive experiments on various real-world datasets and compare Meta-HPO with several benchmarks to validate the effectiveness of our proposed method.

The remaining part of this paper is organized as follows. In Section 2, we review related work on multi-fidelity hyperparameter optimization and multi-task hyperparameter transfer. In Section 3, we introduce the proposed Meta-HPO in detail, including the perturbation-aware adversarial sampling strategy and the hyperparameter transfer module. In Section 4, we demonstrate extensive experimental results on various datasets and compare our model with existing methods to validate its effectiveness.

2 RELATED WORK

Our work is closely related to multi-fidelity hyperparameter optimization and multi-task hyperparameter transfer.

2.1 Multi-fidelity Hyperparameter Optimization

Bayesian Optimization (BO) is proved to be an effective way for hyperparameter optimization, which estimates the posterior distribution of the model performance f to select the next configuration by exploring with a few configurations θ and their corresponding model performances $f(\theta)$. Here f is a black-box function to be estimated. Some BO-based methods [1, 14, 22] have become prevalent state-of-the-art solutions. To optimize the hyperparameters on large-scale datasets, the training on the full dataset is time-consuming, especially in real-world applications. Therefore, multi-fidelity hyperparameter optimization has been studied in several works to speed up the procedure of HPO. From the perspective of sampling subsets, the work [16] proposes to model the joint space of the hyperparameters and subset size under different fidelities s, where s is the size of the sampled subset. From the perspective of limiting training budget, another work [6] allocates the computational resources dynamically through random sampling and eliminates under-performing hyperparameters by successive halving. However, the above attempts to construct subsets by random sampling method suffer from several limitations. Random samples inevitably contain numerous points with low information or even outliers, thus reducing the robustness of the model. In contrast, we design an effective perturbation-aware adversarial sampling

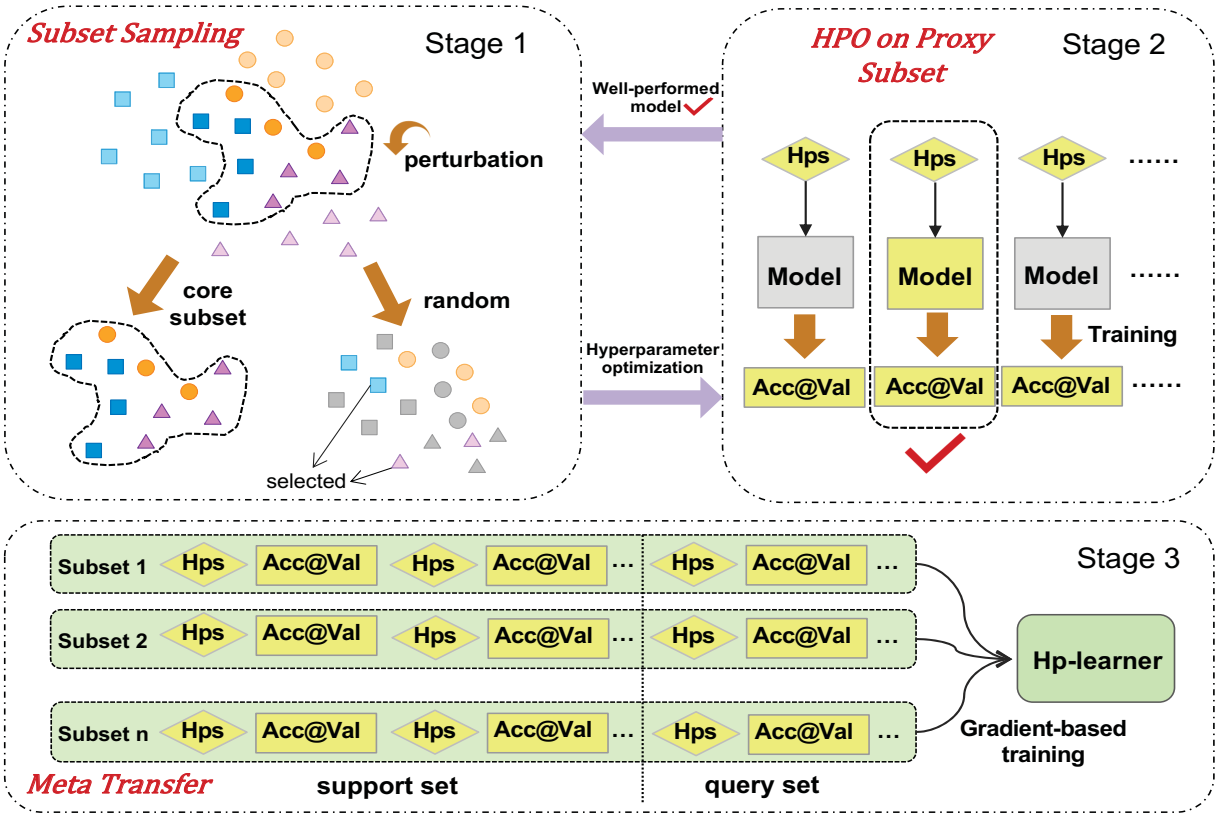


Figure 1: The framework of our Meta-HPO model includes three stages. The Meta-HPO first samples a number of proxy subsets from the full dataset with our proposed perturbation-aware adversarial sampling strategy and then optimizes hyperparameters on the proxy subsets. These two stages are run alternately. In the meta transfer stage, the hp-learner predicts the corresponding performance, such as the accuracy on the validation set, given a group of the hyperparameters "HPs" and the ground truth "Acc@Val" on multiple subsets. Best view in color.

strategy for large-scale datasets with complex and even dynamic distribution. The adversarial sampling strategy finds samples that play decisive roles in the predicted results of the classifier to achieve more effective down-sampling compared with random sampling. In addition, these multi-fidelity hyperparameter optimization approaches [6, 16] cannot transfer the optimal hyperparameters on the existing dataset to new batches of data since they cannot adapt to unseen distribution, whereas our Meta-HPO has the ability to predict well-performed hyperparameters in an online manner.

2.2 Multi-task Hyperparameter Transfer

Recently, multi-task HPO has attracted attention from the machine learning community. Several works have studied hyperparameter transfer within multiple datasets. Studies in [25, 26, 33] use Gaussian Process to represent the implicit relationship across multiple datasets. The transfer process is learned within the kernel matrix. However, the fitting of the Gaussian Process is time-consuming, and the complexity is cubic in the number of datasets. Another study on hyperparameter transfer learning is Gaussian Process ensembles [7, 31]. They develop weighted strategies to combine with previous models to initialize hyperparameters for the new dataset. Meta-learning-based methods are also studied in transferable HPO. Earlier literature [19, 32] use hand-crafted features as meta features to initialize the new model. Recently, the work in [20] conducts Gaussian Process, fitting with a feature map learned by neural networks. Another work [30] uses neural networks to incorporate trials on other datasets and initialize the model, which applies a metric-based attention mechanism to leverage previous observations and the transferring parameters. However, it ignores the meta-information of different datasets. The training of the neural networks requires a certain amount of data points to avoid overfitting on the previous trials. In this paper, we propose a meta transfer framework to build the connection among meta features of the proxy subsets, hyperparameters and corresponding performance and adopt a gradient-based meta-learning paradigm to ameliorate the over-fitting problem.

3 MODEL

Meta-HPO consists of three stages, as shown in Figure 1: the first stage is the adversarial proxy subsets sampling strategy to sample representative subsets from the large-scale dataset as introduced in Sec 3.2; the second stage is optimizing hyperparameters on the

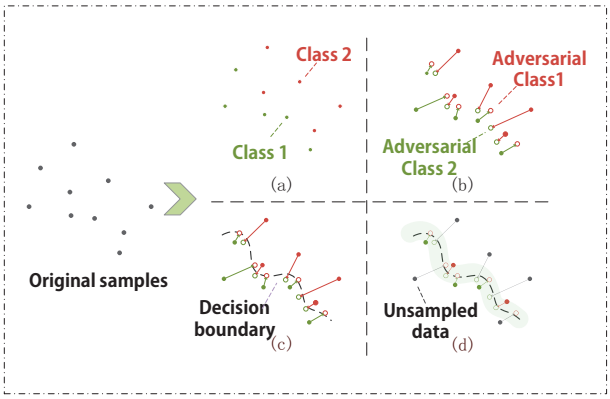


Figure 2: The illustration of the perturbation-aware adversarial sampling strategy in proxy subsets sampling.

proxy subsets; the last stage is meta transfer of hyperparameters with a meta-learning-based transfer mechanism described in Sec 3.3. We first formulate the hyperparameter optimization problem in Sec 3.1 and then introduce our model in detail.

3.1 Problem Formulation

Let $\theta = (\theta_1, \theta_2, ..., \theta_n)$ be the hyperparameters of the machine learning model sampled in the configuration domains $\Theta_1, \Theta_2, ..., \Theta_n$, respectively, which can be numerical or categorical within finite domain. The hyperparameter space is then defined as $\mathbf{\Theta} = \Theta_1 \times \Theta_2 \times ... \times \Theta_n$. Receiving the dataset $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}\}$, we should search in $\mathbf{\Theta}$ for an optimum configuration θ according to the performance of the model, which is trained on the train set $\mathcal{D}_{train}$ and tested on the validation set $\mathcal{D}_{valid}$. Here we take the test error on validation set as the performance of learning model and set the objective as minimizing the test error. Actually, it is the same case if we maximize a metric (i.e. AUC score) of model performance. The hyperparameter optimization problem for a given dataset $\mathcal{D}$ is to minimize $f(\theta, \mathcal{D})$:

$$\min_{\theta \in \Theta} f(\theta, \mathcal{D}) = \mathcal{L}_{val}(\theta, \mathcal{D}_{train}, \mathcal{D}_{valid}), \tag{1}$$

where $\mathcal{L}_{val}(\theta, \mathcal{D}_{train}, \mathcal{D}_{valid})$ is the validation error on $\mathcal{D}_{valid}$.

Since the objective function $f(\theta, \mathcal{D})$ is not differentiable w.r.t θ, we can only acquire a number of inputs θ and their responding outputs. For large-scale dataset, the procedure of acquiring a group of θ and $f(\theta, \mathcal{D})$ is time-consuming. To reduce the time cost of hyperparameter optimization, we sample a number of proxy subsets $\{\mathcal{D}_i\}_{i=1}^N$ with N as the number of subsets and acquire a group of hyperparameter configurations and their corresponding performances $\{\theta_{ik}, f(\theta_{ik}, \mathcal{D}_i)\}_{k=1}^K$ for each subset $\mathcal{D}_i$, where K is the number of configurations. The optimization objective is then defined as predicting an optimum point θ for the full dataset or newly arrived data with previous obtained hyperparameter configurations on several subsets $\{\theta_{ik}, f(\theta_{ik}, \mathcal{D}_i)\}_{k=1}^K$. We should learn the relationship between the distribution of dataset, hyperparameter configuration and its corresponding performance with a meta-learning-based transfer mechanism and then obtain a well-performed hyperparameter configuration on a specific dataset.

3.2 Adversarial Proxy Subsets Sampling

Training a model on a large-scale dataset is prohibitively time-consuming for HPO. We improve the efficiency of HPO by optimizing hyperparameters on sampled proxy subsets instead of the full dataset. However, randomly selected subsets are often unrepresentative. Thus how to build the proxy subsets is the key to the problem. Inspired by [3], we introduce adversarial attack into proxy subsets sampling to measure the importance of points in the full dataset. Generally, data samples that influence the predicted results of a model are distributed near the decision boundary [34]. Thus we take the difficulty of distinguishing samples from different categories as the metric of the importance of samples and select samples that play decisive roles in the learning model.

Since samples with significant influence on the model are generally distributed around the decision boundary, adding a small perturbation to them can result in misclassification of the model. Therefore, the perturbation added to a sample that results in misclassification is a criterion for the importance of points near the decision boundary. We propose a perturbation-aware adversarial sampling strategy for proxy subsets sampling. The sampling process is illustrated in Figure 2. In this process, we iteratively add perturbation to a sample, ensuring the perturbation is the minimum value that results in misclassification of the data sample. These misclassified samples are called adversarial samples. Finally, we estimate the distance of each sample to the decision boundary and select the samples with closer distances to the decision boundary.

Formally, we define such an optimization objective as finding the minimum perturbation ϵ added to the sample x so that $C(x+\epsilon) \neq C(x)$ and $x+\epsilon$ is within the same range of the original x. Here C is the classification result of the classifier. We constrain the range of perturbation value $x+\epsilon \in [x_{min}, x_{max}]^d$, where d is the dimension of the original sample x. Formally, the optimization objective is:

$$\begin{aligned} \min_{\epsilon} \quad & \|\epsilon\|_2^2, \\ s.t. \quad & C(x) \neq C(x+\epsilon). \end{aligned} \tag{2}$$

Here, the optimization constrain $C(x) \neq C(x+\epsilon)$ can be formulated as a criteria of the classification results before and after attack:

$$g(x, \epsilon) = \max(\max_{i \neq t} Z(x+\epsilon)_i - Z(x+\epsilon)_t, -c), \tag{3}$$

where $Z(x+\epsilon)_i$ is the probability that $x+\epsilon$ is classified to i-th class, t is the target class of the adversarial attack and c is a constant that controls the strength of attack. As for regression task, we set a positive threshold σ to measure the difference between the predicted values before and after attack. The constrain function in a regression task is:

$$g(x, \epsilon) = \max(\sigma - |Z(x+\epsilon) - Z(x)|, -c), \tag{4}$$

where $Z(x)$ is the predicted value of the regression model and c is a constant to control the strength of attack.

Considering the different importance of various dimensions in the features of a sample, we further use a trainable matrix $W \in \mathbb{R}^d$ to learn different weights of the perturbations for each dimension. The final optimization function is as follows:

$$\min_{\epsilon} \lambda_1 \|W \cdot \epsilon\|_2^2 + \lambda_2 \cdot g(x, \epsilon), \tag{5}$$

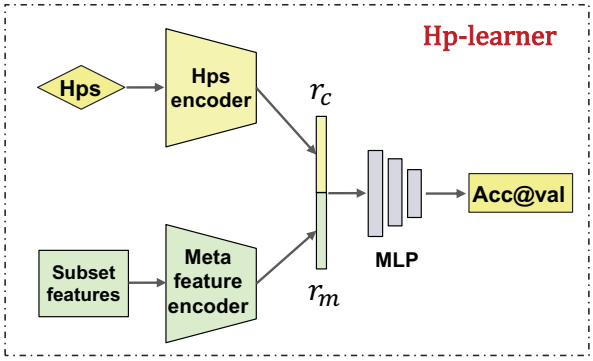


Figure 3: The architecture of our proposed "hp-learner". It consists of a meta feature encoder to extract meta feature from a given subset, an hps encoder to project hyperparameters to the latent space and an MLP predictor to predict corresponding accuracy of given hyperparameters.

where λ_1 and λ_2 are positive factors that balance the perturbation and the constrain function $g(x, \epsilon)$ of the attack. We use Frank-Wolfe algorithm proposed by [21] to explore the values of (λ_1, λ_2) to determine the control factors, thus reduce the time consumption for parameter searching significantly compared with traditional binary search strategy.

Through the perturbation-aware adversarial attack sampling, we obtain proxy subsets containing samples that play significant roles in the model as good abstractions for the full set. After adversarial sampling, the successfully sampled points often account for a small part of the total set. We further balance the representativeness and the diversity of the proxy subsets by supplementing the proxy subsets by randomly selecting additional samples from the unsuccessfully sampled points. In this way, the distribution of the full dataset is well represented by the proxy subsets. In brief, we improve the effectiveness of the sampling module by the proposed perturbation-aware adversarial sampling strategy instead of random sampling.

3.3 Meta Transfer of Hyperparameters

The key idea of the proposed Meta-HPO is to optimize hyperparameters on the proxy subsets and then transfer the well-performed hyperparameter configurations to the full dataset and newly arrived datasets. Given a group of selected proxy subsets, we conduct hyperparameter optimization parallelly on the subsets using BOHB [5] to obtain hyperparameter configurations and corresponding performance $\{\theta_{ik}, f(\theta_{ik}, \mathcal{D}_i)\}_{k=1}^{K}$ for each subset $\mathcal{D}_i$. Then the well-performed hyperparameter configurations are recorded as training examples for our meta-learning-based transfer module, named hp-learner. After training, the hp-learner could predict well-performed hyperparameters with given meta features of the new datasets as inputs.

Meta features extraction

For different proxy subsets, the well-performed hyperparameter configurations are different. The meta feature of $\mathcal{D}_i$ is defined as M_i, which is extracted from the trained neural networks and reflects the distribution characteristics of the proxy subset $\mathcal{D}_i$. Intuitively, machine learning models with the same hyperparameter θ trained on subsets with similar meta features should have similar model performances. We explore the distributions of various datasets by extracting the meta feature of each proxy subset through a designed neural network as the contextual information of the training of hp-learner. Specifically, we first extract the raw features of samples from the trained neural network ϕ_e and apply kernel embedding [23] to obtain the meta feature of each subset. We denote X as the input instance with distribution $P(X)$ and variable Y as the label with distribution $P(Y)$. Given the proxy subset $D_i = \{x_1, ..., x_m\}$ of size m, where these samples are supposed i.i.d with distribution $P(X)$. Then the marginal distribution of X is calculated as the mean embedding of samples:

$$\mu_X = \mathbb{E}[\phi_e(X)] \approx \frac{1}{m}\sum_{i=1}^{m}\phi_e(x_i), \tag{6}$$

where ϕ_e is the trained neural networks to extract features from input data. In addition to the marginal distribution of X, we consider the conditional distribution $P(Y|X)$ as a part of the meta feature. The kernel embedding of conditional distribution $P(Y|X)$ is defined as:

$$\mu_{Y|x} = \mathbb{E}_{Y|x}[\psi_y(Y)] = \int_y \psi_y(y)dP(y|x), \tag{7}$$

where ψ_y is the embedding function of variable Y. Based on the relation between conditional expectation and covariance operators, the study in [24] shows that $\mu_{Y|x}$ can be written as:

$$\mu_{Y|x} = \Psi_y(\Phi_x\Phi_x^T + \lambda I)^{-1}\Phi_x, \tag{8}$$

where Φ_x and Ψ_y are embedding matrices of variable X and Y, respectively. λ is the additional regularization parameter to avoid over-fitting. The final meta feature of subset $\mathcal{D}_i$ is the combination of μ_X and $\mu_{Y|x}$. We concatenate them as $M_i = [\mu_X, \mu_{Y|x}]$.

Meta transfer module

The transfer of hyperparameters from the previously sampled subsets to the full dataset or new dataset is conducted by the meta transfer module, hp-learner. In the hp-learner, we design a mapping

Algorithm 1 Training of the hp-learner

Require: Parameter ϕ_m
training samples $\{M_i, \theta_{ik}, y_{ik}\}_{k=1}^{K}, i = 1, ..., N$
training epoch T, step size factors α, β
1: Randomly initialize parameter ϕ_m
2: **for** $i = 1$ to T **do**
3: Sample batch of subsets D_i
4: **for** all D_i **do**
5: Evaluate $\nabla_{\phi_m}\mathcal{L}_{tr}(\phi_m)$ with respect to ***support*** samples
6: Compute the inner parameters through:
$\phi'_m = \phi_m - \alpha\nabla_{\phi_m}\mathcal{L}_{tr}(\phi_m)$
7: **end for**
8: Compute the outer parameters on ***query*** samples through:
$\phi_m \leftarrow \phi_m - \beta\nabla_{\phi_m}\sum_{D_i}\mathcal{L}_{tr}(\phi'_m)$
9: **end for**
10: **return** ϕ_m

function ϕ_m from the meta features and hyperparameter configurations to the corresponding performance as a neural network, as shown in Figure 3. Denoting the hp-learner as a function ϕ_m, the objective of the hp-learner is to fit the function ϕ_m with several data points. The data points used in training consist of the input meta features and hyperparameter configurations and the output model performance. Specifically, the feature encoder takes as input the proxy subset and encodes the sample features into a latent space of meta feature r_m. On the other hand, the hyperparameters encoder projects hyperparameter configurations into a latent space of r_c. The feature encoder and hyperparameters encoder are both designed as two-layer perceptrons. Then the meta feature r_m and hyperparameters feature r_c are concatenated for subsequent performance prediction. The concatenated feature is fed to a trainable predictor to predict the performance score $\hat{y}$. The predictor is deployed using a three-layer perceptron, and the output of the predictor is the model performance score, such as test accuracy, AUC score and so on. The ground truth of performance score y is obtained from the previous hyperparameter optimization on proxy subsets. The objective function of the training of hp-learner is defined as mean-square-error (MSE) between the ground truth y and the predicted performance value $\hat{y}$. The loss function is formulated as:

$$\mathcal{L}_{tr}(\phi_m) = ||\hat{y} - y||_2^2 = ||\phi_m(M, \theta) - y||_2^2. \tag{9}$$

The meta features and well-performed hyperparameter configurations for previously sampled proxy subsets are considered as training samples for the hp-learner. Through training the hp-learner with the supervision of ground truth y, we construct implicit relation between data distribution and the selection of hyperparameters. A trained hp-learner has the ability to predict the performance score $\hat{y}$ given meta feature of a dataset and a group of hyperparameters. Since the number of the training samples is small, each including a pair of meta feature, hyperparameter configuration and its corresponding performance score, the neural network tends to over-fit on the training samples. To alleviate the over-fitting problem, we introduce a gradient-based meta-learning method to the training of the hp-learner, motivated by model-agnostic meta-learning algorithm [8]. In the training phase, different subsets are considered as different tasks, and each training sample consists of the meta feature, hyperparameter configuration and its corresponding performance score, in the form of $\{M, \theta, y\}$. The training samples are randomly divided into ***support*** set and ***query*** set. The samples in the support set are used in the inner loop gradient descent, and the query samples are used for updating the model in the outer loop. The training process of the hp-learner is described in Algorithm 1.

Hyperparameter Prediction

After the training of the well-designed hp-learner, we can obtain the predicted performance scores with given meta features of various datasets and hyperparameter configurations. We take the hp-learner ϕ_m as a black-box function and use L-BFGS [18], a standard quasi-Newton procedure, to find the optimal hyperparameter configuration θ for a specific dataset. For the full dataset, we compute the meta features as the average of meta features extracted from all proxy subsets. Furthermore, the hp-learner is able to provide adaptive hyperparameter selections in an out-of-sample manner. For the newly arrived dataset, we randomly select a group of hyperparameters for initial training and obtain the meta feature of the new dataset. The experimental results in the next section verify the effectiveness of our Meta-HPO in selecting hyperparameters on the given dataset with the meta transfer module. Overall, our meta-learning-based hyperparameter transfer method is capable of capturing the general relationships between datasets and hyperparameter selections with fewer samples and providing better transfer capabilities on the new dataset.

4 EXPERIMENTS

We conduct various experiments to evaluate the proposed Meta-HPO algorithm in hyperparameter optimization with several benchmark datasets. Furthermore, we visualize the experimental results for better understanding the optimization procedures of Meta-HPO and compared baselines.

4.1 Experimental Settings

Datasets

We choose four datasets, including both public and industrial datasets, to evaluate our Meta-HPO in the experiments: Cifar10 and Cifar100 [17] for image classification; DataCVR and Movielens-10M [10] in the recommendation system. For each of the public datasets, we use the original division of training set and test set, where the former is used for hyperparameter optimization, and the latter is used for performance testing. In Cifar10 and Cifar100, the training set consists of 50,000 images of the size 28×28 in 10 or 100 classes, and the test set includes 10,000 images. Movielens-10M consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. For the industrial dataset, DataCVR is collected from the advertisement recommendation system, which is deployed for providing personal advertisements to users. Here we consider advertisements as items in the recommendation system. The dataset contains 24 user-side features and 14 item-side features, and the user conversion rates are used as labels. We extract 800K historical data of users and items from the database for the advertising conversion rate prediction task, where 600K pieces of data are used for training in HPO, 100K for validation and the remaining 100K for testing the performance of selected hyperparameter configurations.

Baselines

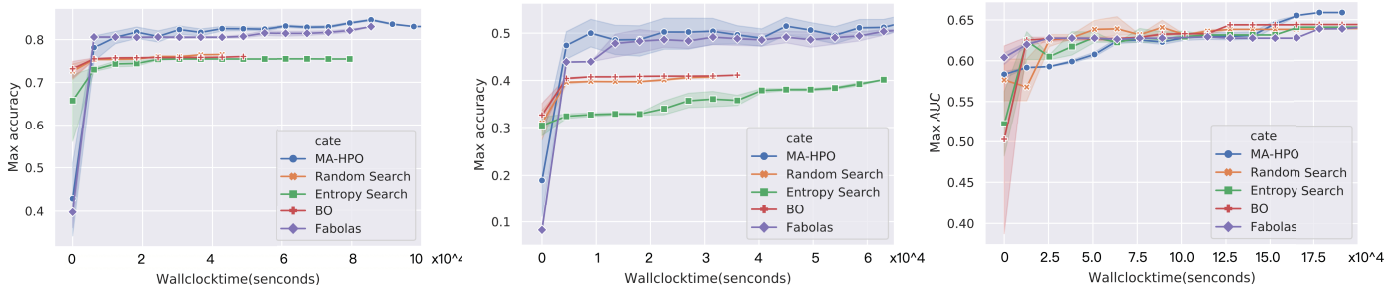
We compare our Meta-HPO method to several hyperparameter optimization baselines: random search (RS) [2], entropy search (ES) [12], Bayesian Optimization (BO) [14] and Fabolas [16]. In the offline validation, we track the model performance on the test set over wall clock time, storing the incumbent returned after every iteration for each method. In the out-of-sample validation scenario, we evaluate the optimum hyperparameter configuration found by Meta-HPO on the new arrival data, visualize the testing results, and use the incumbent configurations as initialization in the four baselines. For all experiments, we take the average performance of 3 runs and report the final results.

Implementation details

In the experiments on Cifar10 and Cifar100, we adopt a three-layer CNN with the kernel size 3×3 to extract features of input images. Then we adopt three fully connected layers as classification layers. The hidden sizes of the classification layers are 2048, 2048,

Table 1: Offline performance comparisons with baselines on five benchmarks. In the image classification experiments, the metric is the classification accuracy of the model trained on the full dataset. In the recommendation experiments, AUC score is used as metric in classification task on DataCVR and Movielens-10M. And MSE is used as metric in the regression task on Movielens-10M. "w/ RS" denotes the meta transfer model with randomly sampled subsets.

Model	Classification				Regression
	Cifar10 ↑	Cifar100 ↑	DataCVR ↑	Movielens ↑	Movielens ↓
RS	0.765	0.401	0.795	0.639	1.357
ES	0.754	0.375	0.792	0.642	1.363
BO	0.767	0.410	0.790	0.644	1.372
Fabolas	0.839	0.507	0.802	0.656	1.364
Meta-HPO w/ RS	0.857	0.545	0.815	0.671	1.308
Meta-HPO (Ours)	**0.872**	**0.568**	**0.827**	**0.678**	**1.302**



(a) Evaluations on Cifar10 (b) Evaluations on Cifar100 (c) Evaluations on Movielens-10M

Figure 4: Test performance of the optimized hyperparameters over wall-clock time on datasets Cifar10(a), Cifar100(b), and Movielens-10M(c). At each time, the curves show the test performance of each method's respective incumbent.

and the number of classes, respectively. There are five hyperparameters to be optimized: the number of kernels of each convolutional layer, the training batch size and the initial learning rate of the Adam optimizer. We evaluate Meta-HPO on the two-tower DNN model [4, 13] for the recommendation system. In the experiments on Movielens-10M, we optimize a total number of ten hyperparameters, including the embedding size of each attribute, the hidden size of each fully connected layer and the initial learning rate of Adam optimizer [15]. As for the DataCVR, we select five sets of hyperparameters as our optimization goal, including continuous hyperparameters: the initial learning rate, and discrete hyperparameters: the number of neurons in the fully connected layer. We search for the optimum hyperparameters on the training set using our method and other baselines and then test the found configurations on the test set. At each iteration, the sampling ratio accounts for 10% of the total set; the maximum training epoch is set as 100, and the minimum training epoch for BOHB [5] is set as 3. The detailed ranges of hyperparameters are listed in Table 2-3.

4.2 Evaluation on Image Classification

Cifar10 and Cifar100

We conduct experiments on image classification tasks, optimizing the validation accuracy of three-layer CNNs on Cifar10 and Cifar100. The optimized hyperparameters are the number of channels in each layer, the training batch size and the initial learning rate of Adam optimizer [15]. The experimental results on Cifar10 and Cifar100 are reported in Table 1. Our Meta-HPO with proxy subsets sampling strategy performs best with the test accuracy of 87.2% on Cifar10 and 56.8% on Cifar100. This validates the effectiveness of the proposed adversarial proxy subsets sampling strategy and meta transfer of hyperparameters. If we use random sampling instead of perturbation-aware adversarial sampling, the test accuracy decreases to 85.7% and 54.5% on Cifar10 and Cifar100, respectively. On the one hand, Fabolas performs best in the four baselines since it samples various subsets for hyperparameter optimization and optimized the size of the subset at each iteration. On the other hand, entropy search has the lowest test accuracy, which indicates that the configuration it finds is not the optimal value.

In addition to the final test accuracy, we compare the searching efficiency of our Meta-HPO and baselines by taking the best test accuracy on the full dataset as the metric of model performance at each iteration. The hyperparameter optimization processes are

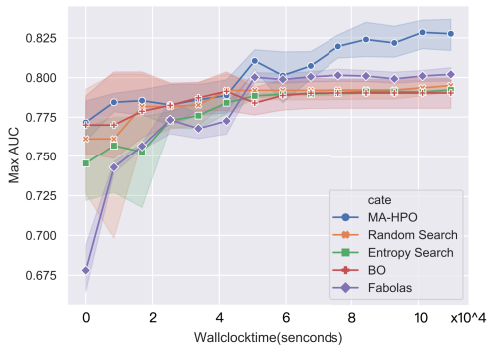


Figure 5: Test performance of all methods in offline scenario on DataCVR.

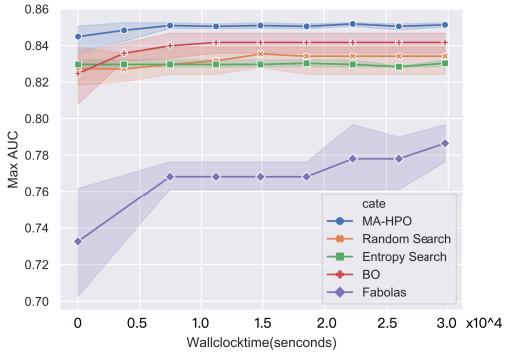


Figure 6: Test performance of all methods in the out-of-sample scenario on DataCVR.

illustrated in Figure 4 (a) and (b). From the test accuracy curves, we can observe that random search and Bayesian optimization methods are capable of finding better configurations at the early stage of searching than other methods. Besides, random search, Bayesian optimization and entropy search stop earlier than others at points of the non-optimal solutions with close performances in searching for optimal hyperparameters. As for HPO methods with subset sampling strategy, Fabolas performs well on Cifar10 and Cifar100, achieving 80% test accuracy in the searching procedure. Compared with Fabolas, our Meta-HPO has a similar upward trend in the early stage but performs better over time, which indicates that our model is able to find a better hyperparameter configuration through constructing the connection among the meta feature of subset, hyperparameters and the performance. The implicit knowledge learned from the proxy subsets facilitates the selection of hyperparameters on the full dataset.

4.3 Evaluation on Recommendation System

In the two-tower DNN model, we obtain the representations of user and item with two separate encoders, and the representations are projected into a shared embedding space through full-connected layers. The similarity between user and item is calculated as the dot product of their embeddings. The metric of the hyperparameter performance is the AUC score in the classification task and the mean-square-error (MSE) in the regression task.

DataCVR

We evaluate the ability of Meta-HPO in a real-world industrial scenario, DataCVR. In the experiment, we optimize five hyperparameters: the number of neurons of four fully-connected layers and the initial learning rate of the Adam optimizer. The testing AUC scores of different methods are shown in Table 1. We can conclude that Fabolas performs best in the baselines with an average AUC 0.802, and the other three baselines have similar performance on DataCVR. Our Meta-HPO achieves the best performance with an AUC score of 0.827. The test AUC scores validate the effectiveness of Meta-HPO in searching for hyperparameters with adversarial proxy subsets sampling and meta transfer of hyperparameters. Without the perturbation-aware adversarial sampling strategy, the testing AUC decreases from 0.827 to 0.815, which indicates the proposed sampling strategy improves the HPO performance in the recommendation system compared with random sampling.

Besides, we compare the search efficiency of different methods since HPO is generally time-consuming on the large-scale dataset. We record the best testing AUC score over wall-clock time and report the curves in Figure 5. From the searching curves, we conclude that our Meta-HPO has better performance at the beginning with the help of perturbation-aware adversarial sampling. The searching curve of Meta-HPO improves steadily and exceeds others after 40,000 seconds. The Meta-HPO achieves the best results in hyperparameter optimization on the DataCVR. We also observe that random search, entropy search and Bayesian optimization perform at an average benchmark point at the beginning. Random search shows a huge variance than other curves at the beginning. On the contrary, Bayesian optimization keeps relatively stable in the early stage and then flattens out. Although the entropy search curve is not as good as the other two curves in the early stage, it shows steady improvement over time. The curve of Fabolas is significantly lower in the initial stage because the random sampling strategy cannot represent the full dataset well. Its performance is the worst in a short period but surpasses other methods with the increase of iterations.

Movielens-10M

To evaluate the proposed method on the public recommendation task, we conduct classification and regression experiments on Movielens-10M. In the regression experiment, the output of the two-tower DNN is the predicted rating score. And we use MSE on the test set as the metric of model performance. In the classification experiment, we divide the training data into positive and negative samples according to the rating scores. Samples with a score greater than 3 are positive, and others are negative ones. We use the AUC score as the measure of the model performance.

The test results on the full dataset are shown in Table 1. Lower MSE and higher AUC scores both indicate better performance of the HPO model. In the classification task, the four baselines perform similarly, and our Meta-HPO has 0.678 test AUC on Movielens-10M, which indicates that the Meta-HPO has the ability to find better hyperparameters for the two-tower DNN recommendation model. A similar performance also appears in the regression task

of Movielens-10M. The random search method has the best performance in the four baselines with the MSE of 1.357. The regression MSE on the test set of Meta-HPO is 1.302, outperforming other baselines. The MSE score increases to 1.308 without adversarial sampling strategy, indicating that the perturbation-aware adversarial sampling strategy is capable of sampling representative subsets for hyperparameter searching. In addition, the training curves on Movielens-10M shown in Figure 4(c) depict the comparisons of our Meta-HPO and baselines. The test accuracy on the full large dataset increases over time. With the searching iteration increasing, the best-performed hyperparameter configurations found by these methods perform similarly. When the training time reaches 150, 000 seconds, our Meta-HPO has a better performance compared with baselines. The experimental results on Movielens-10M demonstrate the effectiveness of our Meta-HPO in hyperparameter optimization with adversarial proxy subsets sampling and meta transfer mechanism.

4.4 Evaluation in Out-of-sample Scenario

In industrial scenarios, new data arrives continually in streaming. The newly arrived dataset usually has concept drift of distribution. The compared baselines here cannot adapt to the concept drift of new data efficiently. However, our Meta-HPO is capable of predicting well-performed configurations by the hp-learner naturally. After training the hp-learner, we take the meta feature of new data as conditional input and find the optimal configurations for the new dataset. In this way, our Meta-HPO can be applied to out-of-sample industrial scenarios, such as advertisement recommendations.

To evaluate the performance of Meta-HPO in the out-of-sample scenario, we use DataCVR as the benchmark here. We divide DataCVR into different batches of training samples from the oldest to the latest in sequence. For the i-th batch, we use this batch of samples to train and update the model and then use the $(i+1)$-th batch for evaluation. In this way, the data distribution changes along with batches, which simulates the industrial application scenario. Since the industrial machine learning models often require real-time performance, we limit the running time of the optimization to 30, 000 seconds. From the performance curves in Figure 6, we can observe that random search, entropy search and Bayesian optimization show similar performance in a relatively short period. Due to the distribution shift in the streaming data, they may encounter the cold-start problem on new datasets. Thus they perform worse than Meta-HPO when the searching time increases. Fabolas achieves poor performance in the out-of-sample scenario with new data. Randomly sampling subsets in the early stage encourages the model to find well-performed hyperparameters quickly. But when faced with newly-arriving data, Fabolas adapts to the changes of data slowly. When the running time reaches 25, 000 seconds, an apparent increasing trend begins in the curve of Fabolas, which is too slow to deploy in the out-of-sample scenario with real-time requirements. On the other hand, with the help of the adversarial sampling strategy and meta transfer module, our Meta-HPO achieves the best performance in a short period. This indicates that Meta-HPO improves the generalization performance of the hyperparameters optimization model in a short time in out-of-sample forecasting, which is quite useful in industrial scenarios.

Table 2: Optimized hyperparameters on Cifar10 and Cifar100.

Hyperparameter	Range
number of kernels in first convolutional layer	[64, 32]
number of kernels in second convolutional layer	[128, 64]
number of kernels in third convolutional layer	[256, 128]
batch size	[512, 128]
learning rate	[1e-6, 1e-1]

Table 3: Optimized hyperparameters on Movielens-10M.

Hyperparameter	Range
embedding size of user id	[64, 20]
embedding size of user gender	[64, 20]
embedding size of user age	[64, 20]
embedding size of user job	[64, 20]
embedding size of movie id	[64, 20]
embedding size of movie type	[64, 20]
number of neurons in first fully connected layer	[128, 64]
number of neurons in second fully connected layer	[64, 32]
number of neurons in third fully connected layer	[32, 16]
learning rate	[1e-5, 1e-2]

5 CONCLUSIONS

This paper proposes a novel method Meta-HPO to optimize hyperparameters in the large-scale dataset and further newly-arriving data. We design a superior perturbation-aware adversarial sampling strategy, instead of traditional random sampling, to sample representative subsets more comprehensively and effectively. Besides, We design a meta transfer framework for hyperparameter forecasting through mining the connection between meta features and hyperparameter configurations. Compared with state-of-the-art HPO methods, our method has the ability to transfer the previously learned knowledge of hyperparameters to the full dataset or new data. In future work, the two-stage optimization with hyperparameter optimization on the proxy subsets and meta transfer of hyperparameters can be further improved with end-to-end optimization.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant No.2020AAA0106301, National Natural Science Foundation of China No.62050110 and Tsinghua GuoQiang Research Center Grant 2020GQG1014.

REFERENCES

[1] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*. 2546–2554.

[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 1 (2012), 281–305.

[3] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on Security and Privacy (SP)* (2017), 39–57.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[5] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774* (2018).

[6] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Practical Hyperparameter Optimization for Deep Learning. In *International Conference on Learning Representations*.

[7] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. 2018. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at International Conference on Machine Learning*.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 1126–1135.

[9] Chaoyu Guan, Xin Wang, and Wenwu Zhu. 2021. Autoattend: Automated attention representation search. In *International Conference on Machine Learning*. PMLR, 3864–3874.

[10] F Maxwell Harper and Joseph A Konstan. 2016. The MovieLens Datasets: History and Context. *Ksii Transactions on Internet and Information Systems* 5, 4 (2016), 19.

[11] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems* 212 (2021), 106622.

[12] Philipp Hennig and Christian J Schuler. 2012. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research* 13, 1 (2012), 1809–1837.

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management*. 2333–2338.

[14] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *LION*.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv: Learning* (2014).

[16] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Vol. 54. 528–536.

[17] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (05 2012).

[18] Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1 (1989), 503–528.

[19] Mohamed Maher and Sherif Sakr. 2019. Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms.

[20] Valerio Perrone, Rodolphe Jenatton, Matthias W. Seeger, and Cédric Archambeau. 2018. Scalable Hyperparameter Transfer Learning. In *Advances in Neural Information Processing Systems*. 6846–6856.

[21] Ozan Sener and Vladlen Koltun. 2018. Multi-Task Learning as Multi-Objective Optimization. In *Advances in Neural Information Processing Systems*. 525–536.

[22] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*. 2960–2968.

[23] Le Song, Kenji Fukumizu, and Arthur Gretton. 2013. Kernel Embeddings of Conditional Distributions: A Unified Kernel Framework for Nonparametric Inference in Graphical Models. *IEEE Signal Processing Magazine* 30, 4 (2013), 98–111.

[24] Le Song, Jonathan Huang, Alexander J. Smola, and Kenji Fukumizu. 2009. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, Vol. 382. 961–968.

[25] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. 2013. Multi-Task Bayesian Optimization. In *Advances in Neural Information Processing Systems*. 2004–2012.

[26] Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, and Wenwu Zhu. 2019. AutoNE: Hyperparameter Optimization for Massive Network Embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 216–225.

[27] Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. 2021. Explainable automated graph representation learning with hyperparameter importance. In *International Conference on Machine Learning*. PMLR, 10727–10737.

[28] Xin Wang and Wenwu Zhu. 2021. Automated Machine Learning on Graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4082–4083.

[29] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling Architecture Search for Graph Classification. In *International Conference on Information and Knowledge Management*.

[30] Ying Wei, Peilin Zhao, Huaxiu Yao, and Junzhou Huang. 2019. Transferable Neural Processes for Hyperparameter Optimization. *arXiv preprint arXiv:1909.03209* (2019).

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

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

[33] Dani Yogatama and Gideon Mann. 2014. Efficient Transfer Learning Method for Automatic Hyperparameter Tuning. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Vol. 33. 1077–1085.

[34] Han-Wei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. 2019. Walking on the Edge: Fast, Low-Distortion Adversarial Examples. *ArXiv* abs/1912.02153 (2019).