HyperSTAR: Task-Aware Hyperparameters for Deep Networks

Gaurav Mittal[*†] Chang Liu[*‡] Nikolaos Karianakis[†] Victor Fragoso[†] Mei Chen[†] Yun Fu[‡]
[†]Microsoft [‡]Northeastern University
{gaurav.mittal, nikos.karianakis, victor.fragoso, mei.chen}@microsoft.com
liu.chang6@husky.neu.edu yunfu@ece.neu.edu

Abstract

While deep neural networks excel in solving visual recognition tasks, they require significant effort to find hyperparameters that make them work optimally. Hyperparameter Optimization (HPO) approaches have automated the process of finding good hyperparameters but they do not adapt to a given task (task-agnostic), making them computationally inefficient. To reduce HPO time, we present HyperSTAR (System for Task Aware Hyperparameter Recommendation), a task-aware method to warm-start HPO for deep neural networks. HyperSTAR ranks and recommends hyperparameters by predicting their performance conditioned on a joint dataset-hyperparameter space. It learns a dataset (task) representation along with the performance predictor directly from raw images in an end-to-end fashion. The recommendations, when integrated with an existing HPO method, make it task-aware and significantly reduce the time to achieve optimal performance. We conduct extensive experiments on 10 publicly available large-scale image classification datasets over two different network architectures, validating that HyperSTAR evaluates 50% less configurations to achieve the best performance compared to existing methods. We further demonstrate that HyperSTAR makes Hyperband (HB) task-aware, achieving the optimal accuracy in just 25% *of the budget required by both vanilla HB and Bayesian Optimized HB (BOHB).*

1. Introduction

Transfer learning has become a de-facto practice to push the performance boundary on several computer vision tasks [8, 49], most notably image classification [16, 19, 38]. Although transfer learning improves performance on new tasks, it requires machine learning (ML) experts to spend hours finding the right hyperparameters (*e.g.*, learning rate, layers to fine-tune, optimizer, *etc.*) that can achieve the best performance. Researchers have relied on Hyperparameter Optimization (HPO), ranging from simple random search [3] to sophisticated Bayesian Optimization [39] and Hyperband [29], to reduce manual effort and automate the process of finding the optimal hyperparameters. Though more effective than manual search, these approaches are still slow since most of them trigger the same procedure for any new task and do not leverage any information from past experiences on related tasks.

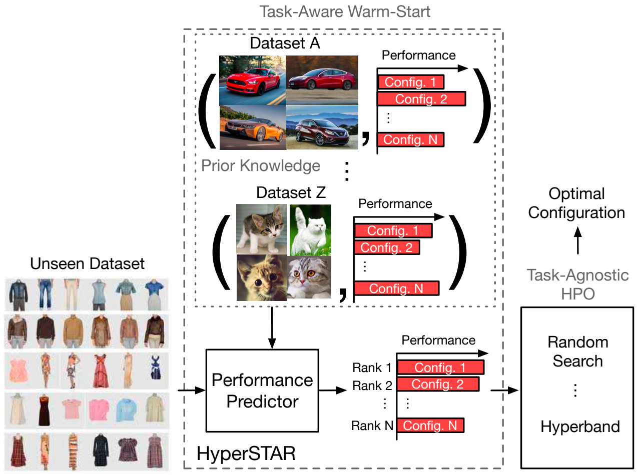

Figure 1. HyperSTAR learns to recommend optimal hyperparameter configurations for an unseen task by learning end-to-end over a joint dataset-hyperparameter space. These recommendations can accelerate existing HPO methods leading to state-of-the-art performance for resource-constrained budgets.

Many approaches accelerate HPO [24, 27, 41] including "warm-start" techniques which exploit the correlation among tasks [41, 36, 12, 2, 47]. Some of these use meta-learning to warm-start HPO by exploiting the task information (meta-features) from past searches [12]. These methods either guide the search policy for hyperparameters using a learned prior over hand-crafted dataset statistics [12, 48, 2], or pick the search policy of the most similar task from a database [47, 10]. Although these methods accelerate HPO, there is no method that leverages visual-based priors or learns deep feature representations to jointly encode dataset and hyperparameters to expedite HPO on

[*] Authors with equal contribution.
This work was done when C. Liu was a research intern at Microsoft.

large-scale image classification tasks. Having such a representation can help systems warm-start and tailor an HPO method based on the task to optimize. While Kim *et al.* [25] and Wong *et al.* [44] suggest using image features to understand the task, their efforts lack a joint representation for the tasks and hyperparameters. We argue that a joint dataset-hyperparameter representation is crucial for large-scale, real-world image classification problems.

With the advent of AutoML [20], there is a strong interest for systems [11, 23] to fully automate the process of training a model on a customer image dataset. To cater to a large number of users, it is essential for AutoML systems to be efficient in searching for the optimal hyperparameters. Given the diversity of real-world image datasets, it is also necessary to prioritize the hyperparameter configurations in a task-aware manner rather than being task-agnostic. A task-aware mechanism understands a given dataset and recommends configurations that can operate well on that dataset. On the other hand, a task-agnostic mechanism treats all datasets equally and sets off the same configuration search regardless of the task.

In order to enable task-aware HPO, we introduce HyperSTAR (System for Task-Aware Recommendation), a warm-start algorithm that prioritizes optimal hyperparameter configurations for an unseen image classification problem. HyperSTAR learns to recommend hyperparameter configurations for a new task from a set of previously-seen datasets and their normalized performance over a set of hyperparameter configurations. It comprises of two phases: an offline meta-learning phase and an online recommendation phase. In the meta-learning phase, HyperSTAR trains a network to first learn a task representation for a given dataset directly from its training images. Then, it uses the representation to learn an accuracy predictor for a given configuration. In the recommendation phase, HyperSTAR predicts the accuracy for each hyperparameter configuration given the task representation of an unseen dataset. It then exploits these predictions to generate a ranking which can be used to accelerate different HPO approaches by prioritizing the most promising configurations for evaluation. See Fig. 1 for an illustration of HyperSTAR.

Our extensive ablation studies demonstrate the effectiveness of HyperSTAR in recommending configurations for real-world image classification tasks. We also formulate a task-aware variant of Hyperband (HB) [29] using the recommendation from HyperSTAR and show that it outperforms previous variations [29, 9, 47] in limited time budget HPO settings. To the best of our knowledge, HyperSTAR is the first warm-starting method that learns to accelerate HPO for large-scale image classification problems from hyperparameters and raw images in an end-to-end fashion.

In sum, the contributions of this work are the following:

- A meta-learning framework, HyperSTAR, that recommends task-specific optimal hyperparameters for unseen real-world image datasets.
- The first method to recommend hyperparameters based on a task-representation learned jointly with a performance predictor end-to-end from raw images.
- HyperSTAR can warm-start and accelerate task-agnostic HPO approaches. We demonstrate this by integrating HyperSTAR with Hyperband which outperforms existing methods in limited budget setting.

2. Related Work

The simplest solution to find hyperparameters for an algorithm is via a grid search over all the possible parameters [3]. Since it is slow and computationally expensive, the community introduced methods such as Bayesian Optimization (BO) [39, 40, 26] that use Gaussian processes for probabilistic sampling and Hyperband [29] which uses random configuration selection and successive halving [22] to speed up HPO. Falkner *et al.* [9] proposed BOHB, a Bayesian optimization and Hyperband hybrid that exploits the tradeoff between performance and time between BO and HB. For low time budgets, BOHB and Hyperband are equally better than BO while for large time budgets, BOHB outperforms all BO, Hyperband, and random search [20].

To accelerate HPO approaches, there are methods that model learning curves [42, 27], use multi-fidelity methods for cheap approximations [24], use gradient-based methods[13, 33, 35], or train on a subset of training data and extrapolate the performance [26] to reduce the overall search time. An alternative way to speed up HPO is via "warm-start" techniques [41, 36, 12, 2, 47]. These techniques exploit correlation between tasks to accelerate HPO. Swersky *et al.* [41] learns to sample hyperparameters based on multi-task Gaussian processes. Xue *et al.* [47] clusters previously-evaluated tasks based on the accuracy on certain benchmark models. Both approaches, while exploiting HPO knowledge from multiple tasks, incur a time overhead as they need to evaluate the new task every time over a certain pool of configurations in order to speed up the search.

To avoid evaluating benchmark configurations, other approaches learn a function to map the trend in performance of a task over the configuration space with some task-based representation [2, 12, 30, 48, 10]. This function is based on multi-task Gaussian processes [2, 48] or random forests [12]. The task representations employed in these methods are based on hand-crafted features such as meta data (*e.g.*, number of samples and labels in the dataset), or first and second order statistics (*e.g.*, PCA, skewness, kurtosis, *etc.*) [20]. Since these features are neither visual-based nor learned jointly with the HPO module, they prove to be inefficient for large-scale vision tasks (see Section 4).

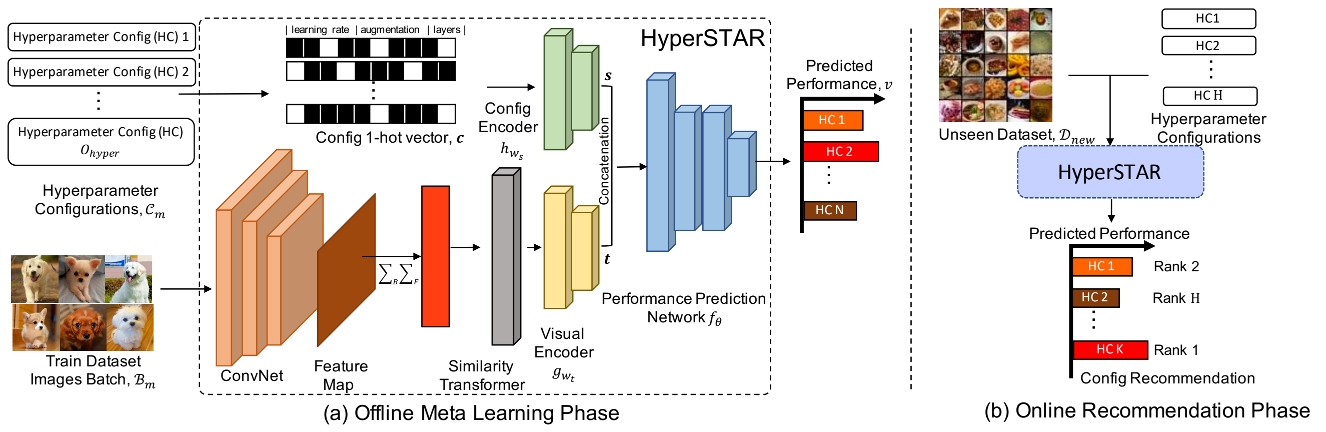
Hyperparameter Config (HC) 1
Hyperparameter Config (HC) 2
Hyperparameter Config (HC) O_{hyper}
Hyperparameter Configurations, $\mathcal{C}_m$
Train Dataset Images Batch, $\mathcal{B}_m$
learning rate | augmentation | layers
Config 1-hot vector, c
Config Encoder h_{w_s}
HyperSTAR
s
t
Concatenation
ConvNet
Feature Map
$\sum_B \sum_F$
Similarity Transformer
Visual Encoder g_{w_t}
Performance Prediction Network f_θ
Predicted Performance, v
HC 1
HC 2
HC N
(a) Offline Meta Learning Phase
Unseen Dataset, $\mathcal{D}_{new}$
HC1
HC2
HC H
Hyperparameter Configurations
HyperSTAR
Predicted Performance
HC 1
Rank 2
HC 2
Rank H
HC K
Rank 1
Config Recommendation
(b) Online Recommendation Phase

Figure 2. HyperSTAR Model Overview. (a) **Offline Meta-Learning Phase**. This phase jointly learns the functions for task representation and hyperparameter representation, and uses them as input to a performance predictor that estimates the performance of a CNN given a dataset (task) and a hyperparameter configuration. (b) **Online Recommendation Phase**. In this phase, HyperSTAR predicts the performance over the hyperparameter space for a new dataset and generates a task-aware ranking of the configurations.

Achille *et al.* [1] introduced task2vec, a visual-inspired task representation but its computational cost makes it ill-suited as conditional input for hyperparameter recommendation. With respect to neural architectures [52], Kokiopoulou *et al.* [28] suggests conditioning the architecture search for natural language tasks over globally averaged features obtained from raw language based data. However, being restricted to low dimensional language tasks and without any dedicated performance based similarity regularization, these methods are not directly applicable to and effective on large scale vision tasks. For vision tasks, Wong *et al.* and Kim *et al.* [44, 25] condition the search of architectures and/or hyperparameters over deep visual features globally averaged over all images. Unlike these methods where features are either not learned or are aggregated via simple statistics (*i.e.*, a global mean), HyperSTAR is the first method that learns an end-to-end representation over a joint space of hyperparameters and datasets. By doing so, HyperSTAR learns features that are more actionable for recommending configurations and for task-aware warm-start of HPO for large-scale vision datasets.

3. HyperSTAR

The goal of HyperSTAR is to recommend tailored hyperparameter configurations for an unseen dataset (task). To introduce this task-awareness, HyperSTAR comprises of a supervised performance predictor operating over a joint space of real-world image classification datasets and hyperparameter configurations. Given a dataset and a hyperparameter configuration, our model learns to predict the performance of the dataset for the given configuration in an offline meta-learning phase. Once the model has learned this mapping, we use HyperSTAR on an unseen dataset in an online recommendation phase to predict scores and rank the hyperparameter configurations. This ranking is beneficial to warm start task-agnostic HPO approaches, as we demonstrate via our formulation of task-aware Hyperband. Figure 2 provides a detailed illustration of HyperSTAR.

3.1. Offline Meta-Learning Phase

Performance Predictor. The objective of the performance predictor is to estimate the accuracy of a hyperparameter configuration given a task representation and an encoding of the hyperparameters (*e.g.*, learning rate, number of layers to fine-tune, optimizer). Mathematically, the performance predictor f is a function that regresses the performance v of a deep-learning based image classifier given a dataset or task $\mathcal{D}$ and a hyperparameter configuration encoding $\mathcal{C}$.

Because deriving this function f analytically is challenging for real-world vision tasks, we instead learn it using a deep network architecture f_θ parameterized with weights θ. Learning f_θ requires a tensor-based representation of the dataset $\mathcal{D}$ and hyperparameter configuration $\mathcal{C}$. To learn the representation of the task $\mathbf{t} \in \mathbb{R}^d$, we search for a function $\mathbf{t} = g_{w_t}(\mathcal{D})$ parameterized by weights w_t. Similarly, we learn the representation $\mathbf{s} \in \mathbb{R}^d$ of a hyperparameter configuration encoded as one-hot vector $\mathbf{c}$ by searching for a function $\mathbf{s} = h_{w_s}(\mathbf{c})$ parameterized by the weights w_s. Mathematically, this is formulated as $v = f_\theta\left(g_{w_t}(\mathcal{D}), h_{w_s}(\mathbf{c})\right)$.

We learn the task representation $\mathbf{t}$ in an end-to-end manner directly from raw training images of dataset $\mathcal{D}$ by using a convolutional neural network followed by a transformer layer inside $g_{w_t}(\cdot)$. This enables the use of visual information into the task representation $\mathbf{t}$, leading to improved generalization over unseen vision tasks and making the method end-to-end differentiable. Jointly learning the performance predictor and the representations in an end-to-end fashion constitutes a departure from previous meta-learning approaches that represent a task using hand-crafted metadata [2] (*e.g.*, total number of training samples, number of classes, number of samples per class, *etc.*), performance-based features [47] or globally averaged features from a frozen deep network [44]. This allows our performance predictor $f_\theta(\cdot)$ to inform the feature extractors $g_{w_t}(\cdot)$ and $h_{w_s}(\cdot)$ during training about the most useful features for estimating the performance of an image classifier given a task and a hyperparameter configuration.

Meta Dataset. To jointly learn the performance predictor $f_\theta(\cdot)$, task representation $g_{w_t}(\cdot)$, and hyperparameter embedding $h_{w_s}(\cdot)$ in a supervised manner, we construct a meta dataset (*i.e.*, a dataset of datasets) $\mathcal{T}$ over the joint space of M datasets and H hyperparameter configurations. We define $\mathcal{T} = \{(\mathcal{D}_i, \mathbf{c}_j, v_{ij}) \mid i \in \{1, \ldots, M\}, j \in \{1, \ldots, H\}\}$, where v_{ij} is the target performance score (*e.g.*, top-1 accuracy) achieved for an image classifier using the hyperparameter configuration $\mathbf{c}_j$ on a dataset $\mathcal{D}_i = \{(x_i^k, y_i^k) \mid k = 0, \ldots, N_i\}$ (each (x_i^k, y_i^k) being an image-label pair in $\mathcal{D}_i$).

Performance Regression. We first find the optimal parameters for θ, w_t, w_s that minimize the difference between the estimated performance of our performance predictor and the ground-truth performance score using the loss function:

$$\mathcal{L}_{\text{perf}}(w_t, w_s, \theta) = \frac{1}{B}\sum_{i=1}^{B} \left\| v_{ij} - f_\theta\left(g_{w_t}\left(\mathcal{D}_i\right), h_{w_s}(\mathbf{c}_j)\right)\right\|_2^2, \tag{1}$$

where B is the number of instances in a batch.

The raw ground truth performance scores v_{ij} across datasets can have a large variance due to the diversity of task difficulty. To alleviate the effect of this variance on our predictor, we normalize the performance scores as, $v_{ij} \leftarrow (v_{ij} - \mu_i)\sigma_i^{-1}$, where μ_i and σ_i are the mean and standard deviation over the performance scores of dataset $\mathcal{D}_i$ for all hyperparameter configurations, respectively.

Although formulating the objective function using a ranking loss [5] seems more intuitive for recommending hyperparameters, Yogatama *et al.* [48] showed that applying the above normalization over v_{ij} makes the regression-based optimization in Eq. (1) equivalent to a rank-based optimization. A regression-based formulation has the advantage of being more time-efficient than rank-based optimization. The time complexity of learning a rank-based predictor is $\mathcal{O}(MH^2)$ while that of a regression-based predictor is $\mathcal{O}(MH)$. Consequently, our regression-based performance predictor can scale favorably to many more datasets.

Similarity-based inter-task regularization. To learn a more meaningful task representation, we add a regularizer that imposes that two tasks must have similar representations if they have similar hyperparameter-configuration rankings. The goal of this regularizer, $\mathcal{L}_{\text{sim}}(w_t)$, is to penalize our model when the similarity of two task representations differs from a pre-computed task similarity between the two datasets. This regularizer is defined as

$$\mathcal{L}_{\text{sim}}(w_t) = \left\| r_{ij} - d(g_{w_t}\left(\mathcal{D}_i\right), g_{w_t}\left(\mathcal{D}_j\right))\right\|_2^2, \tag{2}$$

where r_{ij} is a pre-computed similarity between the i-th and j-th datasets, and $d\left(g_{w_t}\left(\mathcal{D}_i\right), g_{w_t}\left(\mathcal{D}_j\right)\right)$ is the cosine similarity between the two task representations $g_{w_t}\left(\mathcal{D}_i\right)$ and $g_{w_t}\left(\mathcal{D}_j\right)$. We pre-compute r_{ij} as $AP@K$ [51] with $k = 10$. Intuitively, r_{ij} is high when the top-k configurations of the two datasets have a large number of entries in common. $\mathcal{L}_{\text{sim}}(w_t)$ thus helps $g_{w_t}(.)$ push an unseen dataset close to a "similar" seen dataset in the manifold, thereby improving hyperparameter recommendation for this new dataset.

Reducing intra-task representation variance. In order to optimize Eq. (1), we leverage stochastic gradient descent with mini-batch size B. Consequently, this imposes the constraint that a dataset representation $\mathbf{t}_i$ computed from a batch sampled from a dataset $\mathcal{D}_i$ has to be representative of that dataset. In other words, a dataset representation $\mathbf{t}_i^a$ computed from a batch a sampled from $\mathcal{D}_i$ has to be similar to a representation $\mathbf{t}_i^b$ computed from a batch b of the same dataset. Therefore, our model has to ensure that the variance among the task representations computed from any batch of the same dataset has to be small. Inspired by domain adaptation techniques [14, 43, 17], we devise an adversarial training component with the goal of keeping the dataset representations computed from batches ($\mathbf{t}_i^l$) close to the global representation of the dataset ($\mathbf{t}_i^G$). We compute the global representation of the dataset as follows $\mathbf{t}_i^G = \frac{1}{L}\sum_{l=1}^{L} \mathbf{t}_i^l$, where the index l runs through up to last L sampled image batches of a dataset (like a sliding window). We use a discriminator $d_{w_d}(\cdot)$ to ensure that the batch-wise dataset representations $\mathbf{t}_i^l$ are close to the global representation $\mathbf{t}_i^G$. To penalize deviations, we formulate the following loss:

$$\mathcal{L}_{\text{adv}}(w_t, w_d) = \mathbb{E}\left[\log\left(d_{w_d}\left(\mathbf{t}_i^G\right)\right)\right] + \mathbb{E}\left[\log\left(1 - d_{w_d}\left(\mathbf{t}_i^l\right)\right)\right], \tag{3}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. We chose to use an adversarial training component to ensure semantic consistency between batch-wise representations $\mathbf{t}_i^l$ and the global representation $\mathbf{t}_i^G$ as suggested by Hoffman *et al.* [17].

Overall objective function. The overall task representation problem is thus the following

$$\min_{w_t, w_s, \theta} \max_{w_d} \quad \mathcal{L}_{\text{perf}}(w_t, w_s, \theta) + \alpha\mathcal{L}_{\text{sim}}(w_t) + \beta\mathcal{L}_{\text{adv}}(w_t, w_d) \tag{4}$$

where α and β are loss coefficients. We solve this problem by alternating between optimizing feature extractors $g_{w_t}(\cdot)$, $h_{w_s}(\cdot)$ and discriminator $d_{w_d}(\cdot)$ until convergence.

Implementation details of offline meta-learning phase. Algorithm 1 shows the training process for the offline meta-learning phase. The offline meta-learning phase requires two loops. The outer-most for-loop (steps 3 - 12) samples a meta-batch $\mathcal{C}_m$ for the m-th dataset $\mathcal{D}_m$ containing hyperparameter configurations and their performances. The inner-most for-loop (steps 6 - 11) samples image batches from $\mathcal{D}_m$ to update the parameters of the predictor and simultaneously aggregate the global representation $\mathbf{t}_m$ considering up to the L last image batches. In addition to leveraging stochastic gradient descent as described above, sampling image batches to represent a dataset, compared to a single-point estimate [44], helps the dataset (task) to be modeled as a richer distribution in the dataset-configuration space by effectively acting as data augmentation.

3.2. Online Recommendation Phase

Once the performance predictor of our HyperSTAR learns to effectively map a dataset-configuration pair to its corresponding performance score in the offline meta learning phase, we can use it for online recommendation on an unseen dataset $\mathcal{D}_{\text{new}}$ as shown in Algorithm 2 and Figure 2b. HyperSTAR first extracts a task representation $\mathbf{t}_{\text{new}} = g_{w_t}(\mathcal{D}_{\text{new}})$ for the new dataset and then along with a batch of previously-seen hyperparameter configuration encodings, feeds it into the offline-trained performance predictor $f_\theta(\cdot)$ to predict a sequence of performance scores corresponding to the sequence of configurations. Based on these performance scores, we can rank the configurations to prioritize which ones to evaluate.

Task-Aware HPO. This task-aware recommendation list generated by HyperSTAR can be used to warm-start and guide any of the existing HPO approaches. We prove this by proposing a task-aware variant of Hyperband [29]. In this variant of Hyperband, in each stage, we replace the random configuration sampling by evaluating the top n configurations based on the recommendation list suggested by HyperSTAR. We experiment with a thresholded list of top n configurations with Hyperband, but it can be hybridized (without much effort) to either mix a certain ratio of random configurations or sample configurations based on a probability defined over the ranked configuration list.

Implementation details of online phase. Algorithm 2 summarizes the online recommendation phase. The outer-most for-loop (steps 2 - 9) iterates over all the possible H configurations. For each configuration, the inner-most loop (steps 4 - 7) samples B batches and predicts the performance for each batch at step 6. At the end of this inner-most loop, we average all the performance predictions and use it as the performance estimate for the n-th configuration. Lastly, the algorithm ranks all the configurations based on their estimated performances and returns the ranking.

Algorithm 1: Offline Meta-learning Phase

```
1  Input meta-dataset $\mathcal{T}$, $M$ datasets, hyperparameter batch size
     $O_{\text{hyper}}$, image batch size $N_{\text{img}}$, number of sampled image batches
     per dataset $B_{\text{img}}$, window size $L$
2  while Not converge do
3      for m=1 to M do
4          Initialize global task embedding $\mathbf{t}_m^G = \mathbf{0}$
5          Sample hyperparameter batch $\mathcal{C}_m$ from $\mathcal{T}$ for dataset $\mathcal{D}_m$
6          for i=1 to $B_{img}$ do
7              Sample an image batch $\mathcal{B}_m^i$ from $\mathcal{D}_m$
8              Update $\theta, w_t, w_s$ by minimizing Eq. (4)
9              Compute $\mathbf{t}_m^G$ as mean of up to last $L$ image batches
10             Update $w_d$ by maximizing Eq. (4);
11         end
12     end
13 end
```

Algorithm 2: Online Recommendation Phase

```
1  Input Unseen dataset $\mathcal{D}_{\text{new}}$, meta-dataset $\mathcal{T}$, batch sampling
     iterations $B$, number of hyperparameter configurations $H$
2  for n=1 to H do
3      Get the $n$-th hyperparameter configuration $\mathbf{c}_n$ from $\mathcal{T}$
4      for i=1 to B do
5          Randomly sample an image batch $\mathcal{B}_{\text{new}}^i$ from $\mathcal{D}_{\text{new}}$ ;
6          $v_{n,i} = f_\theta\left(g_{w_t}\left(\mathcal{B}_{new}^i\right), h_{w_s}(\mathbf{c}_n)\right)$
7      end
8      $v_n = \frac{1}{B}\sum_i^B v_{n,i}$
9  end
10 Return ranked configurations $\mathbf{c}_1, \ldots, \mathbf{c}_H$ based on $v_1, \ldots, v_H$
```

4. Experiments

This section presents a series of experiments designed to evaluate the performance predictor, the generated recommendation (see Sec. 4.1), and the end-to-end HPO performance of HyperSTAR (see Sec. 4.3).

Datasets. We evaluate HyperSTAR on 10 publicly available large-scale image classification datasets: BookCover30 [21], Caltech256 [15], DeepFashion [31], Food-101 [4], MIT Indoor Scene Recognition [37], IP102 Insects Pests [45], Oxford-IIIT Pets [34], Places365 [50], SUN397 [46] and Describable Texture Dataset (DTD) [6].

Architectures. To ensure that our empirical study reflects the performance of our method on state-of-the-art network architectures, we choose SE-ResNeXt-50 [18], a powerful and large architecture; and ShuffleNet-v2-x1 [32], a compact and efficient architecture. For both networks, we operate in a transfer-learning setting [8] where we initialize the weights of the network from a model pre-trained on ImageNet [7] and fine-tune certain layers of the network while minimizing the multi-class cross entropy loss. The hyperparameter space for SE-ResNeXt-50 consists of 40 configurations varying in learning rate, choice of optimizer, number of layers to fine tune, and data augmentation policy. As ShuffleNet-v2-x1 takes less time to train, we explore a larger search space of 108 configurations over the aforementioned hyperparameter dimensions.

Meta-dataset for training the performance predictor. To construct the meta-dataset over the joint dataset-hyperparameter space, we train both SE-ResNeXt-50 and ShuffleNet-v2-x1 for every configuration in their respective hyperparameter space on each of the 10 datasets. This generates a set of 400 training samples for SE-ResNeXt-50 and 1,080 data samples for ShuffleNet-v2-x1. The meta-dataset thus contains triplets holding a one-hot encoding represent-

Table 1. AP@10 comparison for SE-ResNeXt-50 for 10 public image classification datasets across different methods.

	Test Dataset	BookCover30	Caltech256	DeepFashion	Food101	MIT Indoor	IP102 (Pests)	Oxford-IIIT Pets	Places365	SUN397	Textures (DTD)	Average
Baselines	Feurer *et al.* [12]	38.71	60.59	33.27	48.01	67.81	68.15	71.98	49.20	72.63	59.38	59.67
	Feurer *et al.* [10]	37.13	49.55	28.67	49.60	43.27	50.71	54.50	54.78	54.78	51.97	42.49
	Task-Agnostic	45.63	65.90	31.96	55.28	43.51	63.23	53.90	31.96	45.58	60.07	49.70
	Meta-data	42.10	72.57	46.69	63.32	72.31	73.09	78.11	51.78	88.59	60.13	64.87
	Global Mean	61.64	85.26	44.64	63.95	**79.41**	**89.05**	78.33	62.32	93.36	74.66	73.24
Ablations	Batchwise Mean (BM)	68.16	82.34	62.39	**70.98**	72.51	84.94	**81.43**	88.05	93.74	**82.59**	78.71
	BM + GAN	64.02	83.83	87.63	67.27	76.45	87.49	78.42	**93.41**	92.92	77.21	80.87
	BM + Similarity	62.60	80.97	82.39	67.31	78.79	83.64	79.52	90.37	**94.47**	81.63	80.17
	BM + Similarity + GAN	**68.27**	**86.72**	**91.51**	68.20	77.97	87.52	79.64	91.72	91.85	81.46	**82.49**

Table 2. AP@10 comparison for ShuffleNet-v2-x1 for 10 public image classification datasets across different methods.

	Test Dataset	BookCover30	Caltech256	DeepFashion	Food101	MIT Indoor	IP102 (Pests)	Oxford-IIIT Pets	Places365	SUN397	Textures (DTD)	Average
Baselines	Feurer *et al.* [12]	14.30	3.41	6.75	12.57	9.11	14.21	2.11	25.81	18.45	59.38	11.72
	Feurer *et al.* [10]	21.74	10.95	0.00	11.81	25.74	0.0	11.95	27.87	27.87	0.0	15.25
	Task-Agnostic Baseline	0.00	2.11	15.74	44.69	3.11	26.11	31.24	0.00	0.00	18.65	14.16
	Meta-data	33.27	15.35	31.97	33.44	35.68	54.73	25.46	40.48	37.89	34.88	34.32
	Global Mean	35.30	16.45	17.70	47.41	34.87	49.87	24.98	58.28	43.79	40.71	36.94
Ablations	Batchwise Mean (BM)	76.81	21.69	33.82	80.98	39.56	56.30	44.86	69.88	**50.49**	46.70	52.11
	BM + GAN	69.71	**24.29**	34.34	84.88	46.93	54.96	53.84	62.89	43.98	**51.43**	52.72
	BM + Similarity	**80.23**	20.51	35.28	**87.10**	38.21	**57.73**	**54.69**	71.78	44.78	46.52	53.68
	BM + Similarity + GAN	76.92	21.51	**40.10**	84.60	**47.74**	55.34	47.20	**75.25**	46.09	46.56	**54.13**

ing the hyperparameter configuration, training images of the dataset and corresponding Top-1 accuracy (to be used as performance score that HyperSTAR estimates). We normalize these Top-1 accuracies using the mean and standard deviation computed for each dataset separately over the accuracy scores across the configuration space (Section 3.1).

Evaluation metric. We use Average Precision @ 10 (AP@10) [51] metric (as described in Section 3.1) to assess the ranking of configurations that HyperSTAR produces. This metric reflects the quantity and relative ranking of the relevant configurations predicted by HyperSTAR. We first build a ground-truth hyperparameter recommendation list based on decreasing order of actual Top-1 accuracies. We then compute AP@10 by comparing this list with the predicted recommendation list generated based on the decreasing order of the predicted accuracies from HyperSTAR.

4.1. Performance Predictions and Rankings

We present a quantitative study comparing the task representation, regularization functions and the performance predictor introduced by HyperSTAR with existing methods.

Task Representation Comparison. For this comparison, we use meta-data based task representation as the first baseline. The representation is a subset of statistics used in previous methods [12, 2] which are applicable to our vision datasets (such as number of images, classes and images per class in the dataset). As the second baseline, we consider global mean features based task representation. The global mean is computed by taking an average of the deep visual features obtained from the penultimate layer of ResNet-50 pretrained on ImageNet [44] over all training images of a dataset. In comparison, our task representation is a batchwise mean (BM) taken as mean over end-to-end learned features over a batch of $N_{\text{img}} = 64$ training images. We take $B_{\text{img}} = 10$ of these batches and take an average to obtain the task representation. For training, the size of the hyperparameter batch is $O_{\text{hyper}} = 10$. For each setting, we train our performance predictor and compute AP@10 averaged over 10 trials. We can observe from Tables 1 and 2 that our end-to-end learned task representation (BM) outperforms meta-data-based and global-mean-based task representations by 17.62% and 9.25%, respectively, for SE-ResNeXt-50. The performance gains are similar for ShuffleNet-v2-x1 (see Table 2). This suggests that learning end-to-end visually inspired task representation helps HyperSTAR to recommend better task-aware configurations. It further suggests that representing the dataset as a distribution over a large number of randomly sampled batches is better than representing it as a point estimate using global mean.

Regularization Ablation Study. We perform an internal ablation study comparing AP@10 achieved when using batchwise mean (BM) in HyperSTAR with and without imposing similarity and adversarial based regularization. We can observe from Tables 1 and 2 that imposing the regularizations improves the AP@10 for 6 out of 10 datasets for SE-ResNeXt-50 and 9 out of 10 dataset for ShuffleNet-v2-x1. This suggests that, on expectation, imposing regularization allows the task representations to learn meaningful features over the joint dataset-configuration space. Although there is a time cost associated with introducing regularizations, they provide an added dimension for the user to explore and further improve the configuration recommendation compared to the plain batch-wise mean setting.

Performance Predictor Comparison. We compare HyperSTAR and existing meta-learning based warm-starting HPO methods. We first compare HyperSTAR with Feurer *et al.* [12] that uses random forest regression over a joint vector of meta-data and one-hot encoding of hyperparameters to predict the corresponding Top-1 accuracy and use that to build a recommendation list of configurations. We also compare HyperSTAR with Feurer *et al.* [10] that finds the most sim-

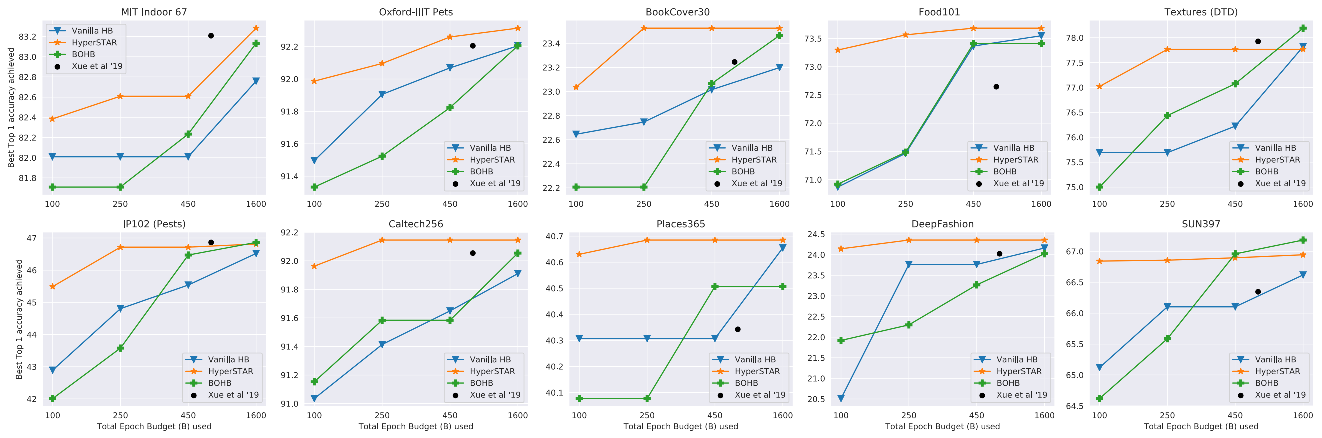

Figure 3. End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for SE-ResNeXt-50. HyperSTAR outperforms other methods when performing on low epoch budgets (100, 250, 450).

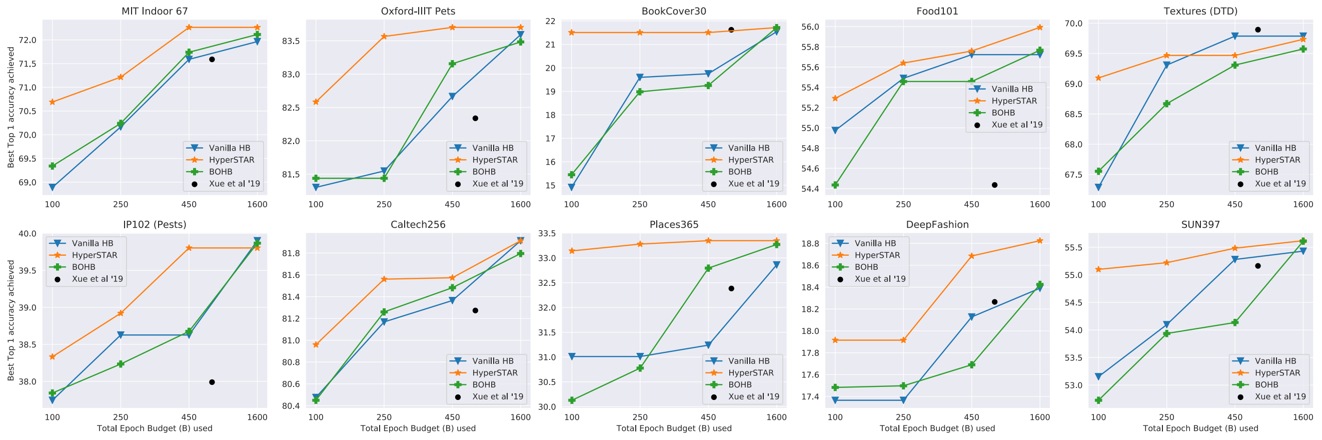

Figure 4. End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for ShuffleNet-v2-x1. HyperSTAR outperforms other methods when performing on low epoch budgets (100, 250, 450).

ilar training dataset for a given test dataset with respect to meta-data features and use the ground truth list of recommended configurations for the training dataset as the prediction for test dataset. We further set up a task-agnostic baseline to compare the effectiveness of our task-aware recommendations. For this baseline, we disregard the characteristics of the test dataset and build a predicted list of recommended configurations by sorting the configurations in decreasing order of their average Top-1 accuracy over the training datasets. From Table 1 and 2, we can observe that HyperSTAR surpasses each of the baselines with average AP@10 margin of at least 25% for SE-ResNeXt-50 and 37% for ShuffleNet-v2-x1. We also observe from the Tables that similarity based approaches (task-agnostic and Feurer *et al*. [10]) have a higher variance in performance across datasets compared to task-representation-based approaches (HyperSTAR and Feurer *et al*. [12]).

4.2. Warm-Starting with Recommendations

We test the configurations recommended by HyperSTAR and other baseline methods by evaluating their ranking order. We plot a curve showing the best Top-1 accuracy achieved after k hyperparameter configurations for $k = 1 \ldots H$ as shown in Figure 5a. We can observe that using the recommendation from HyperSTAR achieves the same performance in just 50% of evaluated configurations as needed by baseline recommendations. This suggests that compared to other baseline methods and task representations, raw-pixel based end-to-end learned task representations of HyperSTAR are more informative for prioritizing hyperparameter configurations. HyperSTAR takes $422ms$ on an Nvidia 1080Ti to generate configuration recommendations which is negligible compared to multiple GPU hours required to evaluate even a single configuration.

4.3. Task-Aware Hyperband

We warm-start Hyperband (HB) [29] using the task-aware hyperparameter recommendation from HyperSTAR and compare it with the vanilla Hyperband [29] and BOHB [9]. We design the experiment to demonstrate a common scenario where the time available to search for the optimal hyperparameters for an unseen dataset is limited.

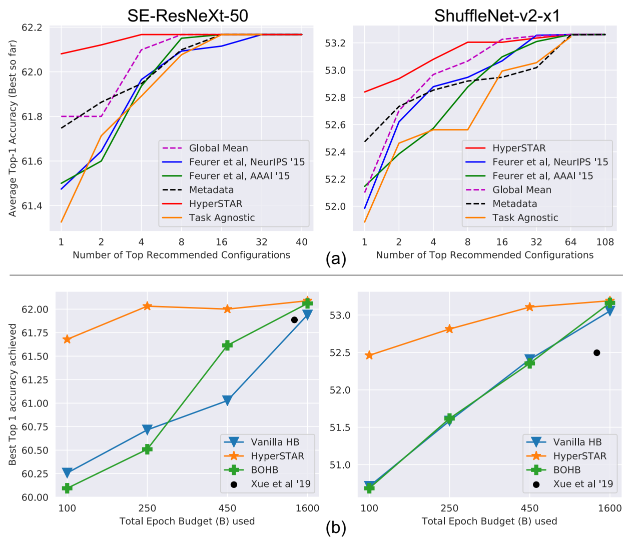

Figure 5. (a) Comparison of evaluating configurations recommended by HyperSTAR with baseline methods in ranked order. Compared to baselines, HyperSTAR achieves the best performance by evaluating 50% less configurations. (b) Comparison of warm-starting Hyperband (HB) with HyperSTAR vs. baseline approaches across different epoch budgets. HyperSTAR achieves optimal accuracy in 25% of the budget required by other methods. It also achieves 1.5% higher best Top-1 accuracy on average for the smallest budget setting on both network architectures.

We run all the methods for different amounts of total budget. The budget is defined in terms of epochs to keep the evaluation time consistent across different hyperparameter configurations, datasets and architectures. The maximum number of epochs for any given configuration is $R = 100$. We consider a budget of 1600 epochs ($\eta = 3$, large-budget setting) and smaller budgets of 450, 200 and 100 epochs ($\eta = 2$, low-budget settings). Figs. 3 and 4 show the best Top-1 accuracy achieved by the different methods for different budgets for all 10 test datasets for SE-ResNeXt-50 and Shufflent-v2-x1, respectively. Fig. 5b further shows the average over all the test datasets for the two network architectures. We observe that HyperSTAR outperforms vanilla HB and BOHB in the low-budget settings for all datasets achieving around 1.5% higher best Top-1 accuracy on average for the smallest budget setting on both network architectures. In fact, HyperSTAR achieves the optimal accuracy in just 25% of the budget required by the other two methods. This happens because the initial set of hyperparameters suggested by both vanilla HB and BOHB do not follow any prior and are chosen randomly, *i.e.*, they are task agnostic.

The difference in the Top-1 accuracy achieved by all three methods gradually diminish with increasing time budget and eventually becomes negligible for the largest budget setting. The accuracy is also at par with the best Top-1 accuracy achievable for the given hyperparameter space. This happens for vanilla HB and BOHB because over time, they explore the hyperparameter space sufficiently enough to be able to discover the best possible configuration. Although HyperSTAR-based Hyperband has been designed to improve HPO efficiency for low-budget setting, being able to achieve the best possible performance suggests that it is also sound for large-budget setting. Given sufficient budget, our method can achieve at par (if not better) performance compared to other HPO methods. Our plots also show BOHB being comparable to vanilla HB for low-budget setting while being better than vanilla HB in the large-budget setting. This is because of the Bayesian sampling prior of BOHB that gets better than random sampling over time, thus helping BOHB outperform vanilla HB.

We also compare our task-aware Hyperband with Tr-AutoML [47]. For a fair comparison, we considered the time Tr-AutoML spends to group the 9 training datasets as part of offline training and exclude it from the time comparison. We randomly choose 10 configurations to group the datasets and evaluate on the test dataset for finding the most similar training dataset. We consider the more time efficient scenario of Tr-AutoML where we do not run Hyperband and compute the Top-1 accuracy achieved over the unseen dataset using the best configuration for the most similar training dataset. As shown in Figures 3, 4 and 5b, since the total evaluation time comprises of running on the benchmark 10 configurations and then finally on the best found configuration, the Top-1 accuracy is reported as achieved after 1100 epochs. From the figures, we can observe that, on expectation, our task-aware HB is able to achieve the same performance in as little as 10 times less budget. This reinforces that learning a dataset embedding from raw pixels significantly reduces the time required to predict the optimal hyperparameters compared to Tr-AutoML.

5. Conclusion

We present HyperSTAR, the first efficient task-aware warm-start algorithm for hyperparameter optimization (HPO) for vision datasets. It operates by learning an end-to-end task representation and a performance predictor directly over raw images to produce a ranking of hyperparameter configurations. This ranking is useful to accelerate HPO algorithms such as Hyperband. Our experiments on 10 real-world image classification datasets show that HyperSTAR achieves the optimal performance in half the number of evaluated hyperparameter configurations compared to state-of-the-art warm-start methods. Our experiments also show that HypterSTAR combined with Hyperband achieves an optimal performance in 25% of the budget of other HB variants. HyperSTAR is especially helpful in performing HPO without requiring large computational time budgets.

6. Acknowledgements

Special thanks to Microsoft Custom Vision team for their valuable feedback and support.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proc. of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019. 3

[2] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *Proc. of the International Conference on Machine Learning*, pages 199–207, 2013. 1, 2, 3, 6

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

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *Proc. of the European Conference on Computer Vision*, 2014. 5

[5] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *Proc. of the Advances in Neural Information Processing Systems*, pages 315–323, 2009. 4

[6] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 5

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5

[8] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proc. of the International Conference on Machine Learning*, pages 647–655, 2014. 1, 5

[9] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proc. of the International Conference on Machine Learning*, 2018. 2, 7

[10] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Proc. of the Advances in Neural Information Processing Systems*, pages 2962–2970, 2015. 1, 2, 6, 7

[11] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Proc. of the Advances in Neural Information Processing Systems*, pages 2962–2970. Curran Associates, Inc., 2015. 2

[12] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 1, 2, 6, 7

[13] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *Proc. of the International Conference on Machine Learning*, pages 1165–1173. JMLR. org, 2017. 2

[14] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 4

[15] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007. 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[17] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. of the 35th International Conference on Machine Learning*, 2018. 4

[18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 5

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 1

[20] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book. 2

[21] Brian Kenji Iwana, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, and Seiichi Uchida. Judging a book by its cover. *arXiv preprint arXiv:1610.09204*, 2016. 5

[22] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proc. of the Artificial Intelligence and Statistics*, pages 240–248, 2016. 2

[23] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proc. of the ACM International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956, 2019. 2

[24] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabas Poczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proc. of the 34th International Conference on Machine Learning*, pages 1799–1808. JMLR. org, 2017. 1, 2

[25] Jungtaek Kim, Saehoon Kim, and Seungjin Choi. Learning to warm-start bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*, 2017. 2, 3

[26] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Proc. of the Artificial Intelligence and Statistics*, pages 528–536, 2017. 2

[27] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. *ICLR*, 2016. 1, 2

[28] Efi Kokiopoulou, Anja Hauth, Luciano Sbaiz, Andrea Gesmundo, Gabor Bartok, and Jesse Berent. Fast task-aware architecture inference. *arXiv preprint arXiv:1902.05781*, 2019. 3

[29] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, Jan. 2017. 1, 2, 5, 7

[30] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[31] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 5

[32] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proc. of the European Conference on Computer Vision*, pages 116–131, 2018. 5

[33] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proc. of the International Conference on Machine Learning*, pages 2113–2122, 2015. 2

[34] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 5

[35] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *Proc. of the International Conference on Machine Learning*, 2016. 2

[36] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Proc. of the Advances in Neural Information Processing Systems*, pages 6845–6855, 2018. 1, 2

[37] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009. 5

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1

[39] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Proc. of the Advances in Neural Information Processing Systems*, pages 2951–2959, 2012. 1, 2

[40] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *Proc. of the International Conference on Machine Learning*, pages 2171–2180, 2015. 2

[41] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Proc. of the Advances in Neural Information Processing Systems*, pages 2004–2012, 2013. 1, 2

[42] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. 2

[43] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017. 4

[44] Catherine Wong, Neil Houlsby, Yifeng Lu, and Andrea Gesmundo. Transfer learning with neural automl. In *Proc. of the Advances in Neural Information Processing Systems*, pages 8356–8365, 2018. 2, 3, 4, 5, 6

[45] Xiaoping Wu, Chi Zhan, Yu-Kun Lai, Ming-Ming Cheng, and Jufeng Yang. Ip102: A large-scale benchmark dataset for insect pest recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8787–8796, 2019. 5

[46] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. 5

[47] Chao Xue, Junchi Yan, Rong Yan, Stephen M Chu, Yonggang Hu, and Yonghua Lin. Transferable automl by model sharing over grouped datasets. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9002–9011, 2019. 1, 2, 4, 8

[48] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *Proc. of the Artificial intelligence and statistics*, pages 1077–1085, 2014. 1, 2, 4

[49] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proc. of the Advances in Neural Information Processing Systems*, pages 3320–3328, 2014. 1

[50] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5

[51] Mu Zhu. Recall, precision and average precision. 2004. 4, 6

[52] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3