International Journal of Computer Vision (2024) 132:1913–1927
https://doi.org/10.1007/s11263-023-01961-0

HyperSTAR: Task-Aware Hyperparameter Recommendation for Training and Compression

Chang Liu[1] · Gaurav Mittal[2] · Nikolaos Karianakis[2] · Victor Fragoso[2] · Ye Yu[2] · Yun Fu[1] · Mei Chen[2]

Received: 20 December 2022 / Accepted: 19 November 2023 / Published online: 21 December 2023

Abstract

Hyperparameter optimization (HPO) methods alleviate the significant effort required to obtain hyperparameters that perform optimally on visual learning problems. Existing methods are computationally inefficient because they are task agnostic (i.e., they do not adapt to a given task). We present HyperSTAR (System for Task Aware Hyperparameter Recommendation), a task-aware HPO algorithm that improves HPO efficiency for a target dataset by using prior knowledge from previous hyperparameter searches to recommend effective hyperparameters conditioned on the target dataset. HyperSTAR ranks and recommends hyperparameters by predicting their performance on the target dataset. To do so, it learns a joint dataset-hyperparameter space in an end-to-end manner that enables its performance predictor to use previously found effective hyperparameters for other similar tasks. The hyperparameter recommendations of HyperSTAR combined with existing HPO techniques lead to a task-aware HPO system that reduces the time to find the optimal hyperparameters for the target learning problem. Our experiments on image classification, object detection, and model pruning validate that HyperSTAR reduces the evaluation of different hyperparameter configurations by about 50% compared to existing methods and, when combined with Hyperband, uses only 25% of the budget required by the vanilla Hyperband and Bayesian Optimized Hyperband to achieve the best performance.

Keywords Hyperparameter search · AutoML · Warm start · Meta-learning · Image classification · Object detection · Model pruning

Communicated by Arun Mallya.

C. Liu and G. Mittal: Equal contribution. Chang Liu was a research intern at Microsoft.

✉ Mei Chen
mei.chen@microsoft.com

Chang Liu
liu.chang6@northeastern.edu

Gaurav Mittal
gaurav.mittal@microsoft.com

Nikolaos Karianakis
nikolaos.karianakis@microsoft.com

Victor Fragoso
victor.fragoso@microsoft.com

Ye Yu
yu.ye@microsoft.com

Yun Fu
yunfu@ece.neu.edu

[1] Northeastern University, Boston, MA, USA

[2] Microsoft Cloud + AI, Redmond, WA, USA

1 Introduction

Deep neural networks (DNNs) are the main tool to solve complex recognition problems (e.g., image classification, object detection, image segmentation) (Girshick, 2015; He et al., 2016, 2017; Redmon & Farhadi, 2017; Ren et al., 2015; Vaswani et al., 2017). Despite their success and adoption for solving many recognition problems, DNNs require a significant effort in finding hyperparameters (e.g., learning rate, layers to fine-tune, optimizer) that make them operate optimally. To alleviate this effort, researchers employ transfer learning techniques (Girshick, 2015; He et al., 2016, 2017; Huang et al., 2017; Simonyan & Zisserman, 2014; Redmon & Farhadi, 2017; Ren et al., 2015) along with Hyperparameter Optimization (HPO) techniques (ranging from random search (Bergstra & Bengio, 2012) to sophisticated Bayesian Optimization (Snoek et al., 2012) and Hyperband (HB) Li et al., 2017) to automate the process of training DNNs with optimal hyperparameters.

While existing HPO methods automate and alleviate the process of finding the optimal hyperparameters of a DNN

for a given recognition problem, they are agnostic of any prior information gained from past search experiences on related datasets. The shortcoming of this is that HPO methods can be slow, especially on large-scale datasets and complex architectures. This is a challenging aspect of HPO methods as many users often have a tight computational budget, thus making these methods not suitable for complex recognition tasks. In this work, we show that prior information gained from past hyperparameter searches on related datasets and architectures can accelerate the convergence of HPO methods and satisfy the computational budget required by the users.

Many approaches accelerate HPO (Kandasamy et al., 2017; Klein et al., 2016; Swersky et al., 2013) by means of "warm-start" techniques that exploit the correlation among datasets (Swersky et al., 2013; Perrone et al., 2018; Feurer et al., 2015c; Bardenet et al., 2013; Kim et al., 2017; Xue et al., 2019). Some of these use meta-learning to warm-start HPO by exploiting meta-features (encoded information about datasets used in previous hyperparameter searches Feurer et al., Feurer et al. (2015c)). These methods either guide the search policy for hyperparameters using a learned prior over hand-crafted dataset statistics (Feurer et al., 2015c; Yogatama & Mann, 2014; Bardenet et al., 2013), or picking the search policy of the most similar dataset from a database (Xue et al., 2019; Feurer et al., 2015a). Although these methods accelerate HPO, there is no method that leverages visual-based priors or learns representations that jointly encode dataset and hyperparameter information to expedite HPO on large-scale visual recognition problems. Having such a representation could help systems accelerate and tailor an HPO method for both the user dataset and the architecture.

With the advent of AutoML (Hutter et al., 2018), there is a strong interest for systems (Feurer et al., 2015b; Jin et al., 2019) that can fully automate the process of training a model on a customer image dataset. We use "task" to refer to a dataset that specifies a supervised learning problem throughout the remainder of the paper. 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 hyperparameters that can operate well on that dataset and a given architecture. 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 a new image classification, object detection, or model pruning task. HyperSTAR learns to recommend hyperparameter configurations for a new task from a set of

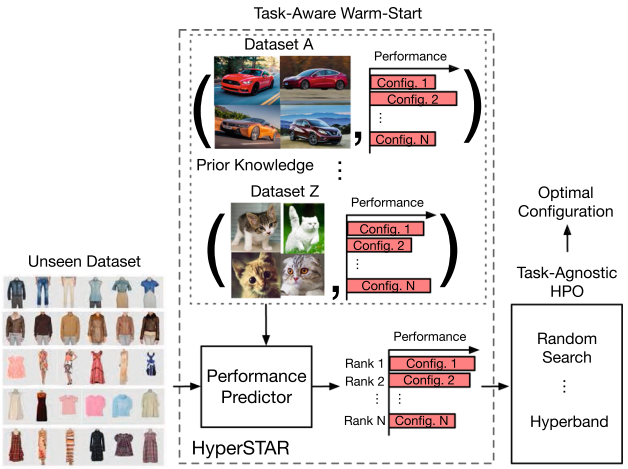

Fig. 1 HyperSTAR recommends optimal hyperparameter configurations for a new task by learning a performance predictor and a joint dataset-hyperparameter space in an end-to-end fashion. HyperSTAR learns from a prior-knowledge set, called meta-dataset, that consists of a collection of datasets finetuned on different hyperparameter configurations and their corresponding performances. The recommendations of HyperSTAR accelerate existing HPO methods leading to state-of-the-art performance within a reduced computational budget

previously-seen datasets and their normalized performance over a set of different 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 hyperparameter 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 that can be used to accelerate different HPO approaches by prioritizing the most promising configurations for evaluation. See Fig. 1 for an illustration of HyperSTAR.

Unlike the original HyperSTAR (Mittal et al., 2020) that only showed benefits on image classification problems, we present extensions of HyperSTAR that enable it to operate well on object detection problems, to recommend model compression parameters, and to include a meta-balancer that improves the model's ability to recommend less-frequent yet high-performing hyperparameter configurations.

Our extensive experiments on challenging datasets demonstrate the effectiveness of HyperSTAR in recommending hyperparameter configurations and providing significant budget savings for real-world image classification, object detection, and DNN pruning tasks compared to previous warm-start approaches. We also formulate a task-aware variant of Hyperband (Li et al., 2017) using the recommendations from HyperSTAR and show that it outperforms previous variations (Li et al., 2017; Falkner et al., 2018; Xue et al., 2019)

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, object detection, and model compression problems from both 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 for image classification, object detection, and model pruning.
- First method to recommend hyperparameters directly based on raw images that can warm-start/accelerate task-agnostic HPO like Hyperband on limited time budget settings.
- A novel balanced meta-sampling method that enables HyperSTAR to generalize well by correctly recommending less-frequent yet high-performing hyperparameters.

2 Related Work

The simplest solution to find hyperparameters for an algorithm is via a grid search over all the possible parameters (Bergstra & Bengio, 2012). Since it is slow and computationally expensive, the community introduced methods such as Bayesian Optimization (BO) (Snoek et al., 2012, 2015; Klein et al., 2017) that use Gaussian processes for probabilistic sampling and Hyperband (Li et al., 2017) which uses random configuration selection and successive halving (Jamieson & Talwalkar, 2016) to speed up HPO. Falkner et al. (2018) 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 (Hutter et al., 2018).

To accelerate HPO approaches, there are methods that model learning curves (Swersky et al., 2014; Klein et al., 2016), use multi-fidelity methods for cheap approximations (Kandasamy et al., 2017), use gradient-based methods (Franceschi et al., 2017; Maclaurin et al., 2015; Pedregosa, 2016), or train on a subset of training data and extrapolate the performance (Klein et al., 2017) to reduce the overall search time. An alternative way to speed up HPO is via "warm-start" techniques (Swersky et al., 2013; Perrone et al., 2018; Feurer et al., 2015c; Bardenet et al., 2013; Xue et al., 2019). These techniques exploit the correlation between tasks to accelerate HPO. Swersky et al. (2013) learns to sample hyperparameters based on multi-task Gaussian processes. Xue et al. (2019) clusters previously-evaluated tasks based on the accuracy of 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 (Bardenet et al., 2013; Feurer et al., 2015c; Lindauer & Hutter, 2018; Yogatama & Mann, 2014; Feurer et al., 2015a). This function is based on multi-task Gaussian processes (Bardenet et al., 2013; Yogatama & Mann, 2014) or random forests (Feurer et al., 2015c). 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.) (Hutter et al., 2018). 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 Sect. 1).

Achille et al. (2019) introduced task2vec, a visual-inspired task representation but its computational cost makes it ill-suited for hyperparameter recommendation under tight budget. Kokiopoulou et al. (2019) 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. (2018), and Kim et al. (2017) 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.

Subsequent works can be categorized into: HPO, Neural Architecture Search (NAS), and Model selection from the zoo of pre-trained models. For HPO, HyperFD (Yan et al., 2022) extends our HyperSTAR with a privacy-preserving module and continuous learning framework. AT2 (Xiao et al., 2021) leverages cheap-to-obtain low-fidelity observations for measuring inter-task dependency efficiently. To cover a further range of components in AutoML, several works propose a joint optimization of Data augmentation policy, Hyperparameter, and Neural Architecture in image classification [DHA (Zhou et al., 2021)] and medical image segmentation [T-AutoML (Yang et al., 2021)]. For Model selection from a Zoo of models, Li et al. (2020) considers model selection as a recommendation problem and to learn from the past train-

ing history. In addition, NASOA (Xu et al., 2021) proposes a faster fine-tuning pipeline that seamlessly combines the training-efficient NAS and online adaption algorithm.

Discussion: HPO, NAS, and Model selection are highly related tasks.

- HPO and NAS: Both HPO and NAS focus on optimizing aspects of machine learning models. While HPO tunes hyperparameters for a fixed architecture, NAS optimizes the architecture itself. The two can be combined, where HPO is used to optimize hyperparameters within the architectures suggested by the NAS process.
- HPO and Model Selection: In the context of model selection, hyperparameters play a crucial role. Different pre-trained models might require different hyperparameter settings to perform well on a specific task. HPO can help find the optimal hyperparameters for a selected pre-trained model. In our experiments, we only adopt two pretrained models: SE-ResNeXt-50 and ShuffleNet-v2-x1. Our work can be combined with Model selection with a larger Zoo of models.

In conclusion, these concepts are interconnected in the sense that they all contribute to the process of optimizing machine learning models. They address different levels of optimization-ranging from tuning hyperparameters, selecting architectures, to choosing pre-trained models-ultimately leading to improved model performance and efficiency.

3 HyperSTAR

HyperSTAR recommends tailored hyperparameter configurations for an unseen dataset (task). To introduce this task-awareness, it comprises of a supervised performance predictor operating over a joint space of real-world image 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 enables the warm starting of task-agnostic HPO approaches, as demonstrated by our task-aware Hyperband formulation. Figure 2 provides a detailed illustration of HyperSTAR.

3.1 Offline Meta-Learning Phase

Performance Predictor This component estimates the accuracy of a hyperparameter configuration given a task representation and an encoding of the hyperparameters (e.g., learning rate, optimizer, layers to finetune). Mathematically, the performance predictor f is a function that regresses the performance v of a DNN on a dataset (task) $\mathcal{D}$ for a hyperparameter configuration $\mathcal{C}$.

Because deriving this function f analytically is challenging, we instead learn it using a deep network architecture f_θ parameterized with weights θ. 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, the predictor is formulated as $v = f_\theta\left(g_{w_t}(\mathcal{D}), h_{w_s}(\mathbf{c})\right)$.

Task representations for Image Classification We learn the task representation $\mathbf{t}$ in an end-to-end manner directly from

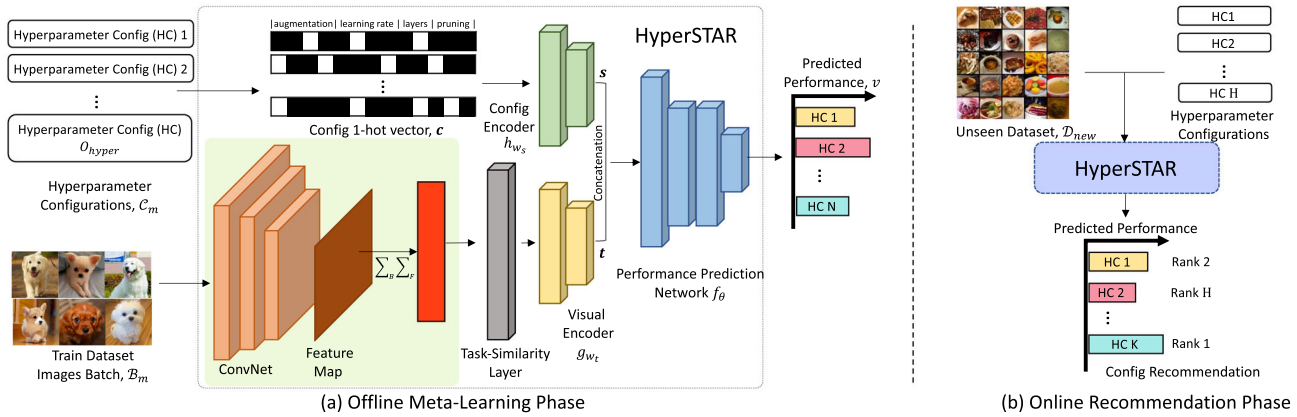

Fig. 2 HyperSTAR Model Overview for Image Classification. **a** Offline Meta-Learning Phase. This phase jointly learns the functions for task representation and hyperparameter representation, and uses them as input to a performance predictor that estimates the performance of a CNN given a dataset (task) 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

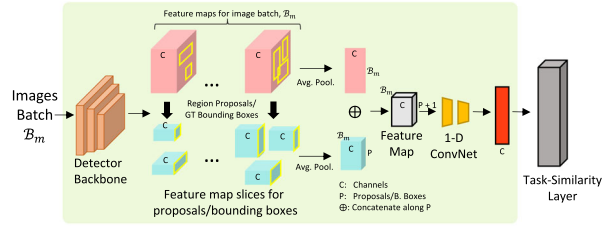

Fig. 3 Visual encoder for object detection tasks. The visual encoding requires adaptations in object detection to account for object localization and multi-label classification

pixels of training images in dataset $\mathcal{D}$ using a CNN followed by a task-similarity layer inside $g_{w_t}(\cdot)$. To do so, we globally average the feature maps from the CNN backbone of image classifier along spatial and batch dimensions before feeding them to the task-similarity layer (Fig. 2). This encodes the visual information into the task representation $\mathbf{t}$, leading to improved generalization over unseen vision tasks and an end-to-end differentiable method.

Task representations for Object Detection Contrary to image classification, object detection also comprises object localization features and the object classification is multi-label. Therefore, we combine object-specific and full-image features in the task representations for object detection. This feature combination effectively informs HyperSTAR about the nature and distribution of objects in the images, object-specific local context, and full-image-based global context.

Figure 3 shows the modifications to the task representation module for object detection. Our adaptation works for both one-stage [e.g., YoloV2 (Redmon & Farhadi, 2017)] and two-stage [e.g., Faster RCNN (Ren et al., 2015)] object detectors. We first feed-forward a batch of images for a dataset through the backbone network to generate a batch of feature maps. From each feature map, we obtain object-specific feature maps by extracting parts of the feature map using the top 1000 region proposals generated by a two-stage detector and ground truth bounding boxes for a one-stage detector. Then, we globally average each of the extracted object-feature maps over height and width as well as the full-image feature map. Next, we stack the combination along the channel dimension and feed forward the stack of feature vectors through a couple of 1D convolutional layers to obtain a single feature vector that is similar in dimension to that of image classification. Lastly, we feed this feature vector to the task-similarity layer after which the remaining of the HyperSTAR framework is applied in the same way as image classification.

Learning the performance predictor and task representations in an end-to-end manner constitutes an innovation over previous meta-learning methods that represent a task using hand-crafted metadata (e.g., number of training samples, classes, samples per class, etc.) (Bardenet et al., 2013), performance-based features (Xue et al., 2019), or globally averaged features from a frozen deep network (Wong et al., 2018). This enables 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 or object detector 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 $\mathcal{T} = \{(\mathcal{D}_i, \mathbf{c}_j, v_{ij}), i \in \{1, \ldots, M\}, j \in \{1, \ldots, H\}$ where v_{ij} is the target performance score (e.g., top-1 accuracy) achieved for an image classifier/object detector using the hyperparameter configuration $\mathbf{c}_j$ on a dataset $\mathcal{D}_i = \{(x_i^k, y_i^k), k \in 1, \ldots, N_i\}$.

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:

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

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 (Chen et al., 2009) seems more intuitive for recommending hyperparameters, Yogatama and Mann (2014) 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 regularizer, $\mathcal{L}_{\text{sim}}(w_t)$, penalizes 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:

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

where r_{ij} is a pre-computed similarity between the ith and jth datasets, and $d\left(g_{w_t}(\mathcal{D}_i), g_{w_t}(\mathcal{D}_j)\right)$ is the cosine similarity between the two task representations $g_{w_t}(\mathcal{D}_i)$ and $g_{w_t}(\mathcal{D}_j)$. We pre-compute r_{ij} as $AP@K^{\mathcal{D}_{ij}}$ (Zhu, 2004) with $k = 10$. Intuitively, r_{ij} is high when the top-k configurations of 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 To optimize Eq. (1), we use stochastic gradient descent with mini-batch size B. This enforces 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. Thus, our model enforces that the variance among the task representations computed from any batch of the same dataset is small. Inspired by domain adaptation techniques (Ganin & Lempitsky, 2015; Tzeng et al., 2017; Hoffman et al., 2018), we devise an adversarial training component to keep 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:

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

where $\mathbb{E}[\cdot]$ is the expectation operator. We 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. (2018).

Balanced Meta Sampling Similar to the long-tail distribution problem in visual recognition (Kozerawski et al., 2020; Wang et al., 2017), a set of hyperparameter configurations (the head configurations) occur more frequently as the high-performing configurations than the remaining configurations (the tail configurations). This would make HyperSTAR prone to rank these frequently-occurring hyperparameter configurations high even for datasets that perform poorly under these configurations. To alleviate this 'long-tail' bias in HyperSTAR, we develop a balanced meta-sampling protocol using the similarity metric $AP@K$.

Given a dataset $\mathcal{D}_i$ in the meta dataset $\mathcal{T}$ and H hyperparameter configurations, we define a ranking order over the configurations with respect to $\mathcal{D}_i$ as $\mathbb{C}^{\mathcal{D}_i} = \left\{\mathbf{c}_1^{\mathcal{D}_i}, \mathbf{c}_2^{\mathcal{D}_i}, \ldots, \mathbf{c}_H^{\mathcal{D}_i}\right\}$, where $\mathbf{c}_1^{\mathcal{D}_i} \geq \mathbf{c}_2^{\mathcal{D}_i} \geq \ldots \geq \mathbf{c}_H^{\mathcal{D}_i}$. Setting this as the reference configuration order, we can compute the precision $P@k^{\mathcal{D}_{ij}}$ at position k where $k \in \{1, \ldots, K\}$, $K \leq H$ for the configuration orders $\mathbb{C}^{\mathcal{D}_j}$ where $j \in \{1, \ldots, M\}$ and $j \neq i$. The average of these precisions up to K represents Average Precision AP@K between $\mathcal{D}_i$ and $\mathcal{D}_j$ denoted by $AP@K^{\mathcal{D}_{ij}} = \frac{1}{K}\sum_{k=1}^{K}\text{P@k}^{\mathcal{D}_{ij}}$. $AP@K^{\mathcal{D}_{ij}}$ represents how similar the hyperparameter configuration ranking of $\mathcal{D}_j$ is to that of $\mathcal{D}_i$. To measure how similar a dataset $\mathcal{D}_i$ is with respect to the other datasets in the meta dataset, we define a similarity score: $\text{sim}AP@K^{\mathcal{D}_i} = \sum_{j=1, j\neq i}^{M} AP@K^{\mathcal{D}_{ij}}$.

Higher $\text{sim}AP@K$ suggests that the dataset shares its high performing hyperparameter configurations with many other datasets in the meta dataset. While lower $\text{sim}AP@K$ suggests that the high performing configurations of the dataset are uncommon among other datasets. Subjecting HyperSTAR to datasets with lower $\text{sim}AP@K$ more often during offline meta-learning phase can boost the representation of configurations that occur less commonly as high-performing configurations and offset the bias from hyperparameter configurations that occur more frequently as high performing configurations. To this end, we define the probability of sampling a dataset $\mathcal{D}_m$ in any given training iteration of the offline phase as,

$$P(\mathcal{D}_m) = \frac{e^{-\text{sim}AP@K^{\mathcal{D}_m}}}{\sum_{j=1}^{M} e^{-\text{sim}AP@K^{\mathcal{D}_j}}}. \tag{4}$$

We use $P(\mathcal{D}_m)$ as a sampling weight to form the meta-batch because $-\text{sim}AP@K^{\mathcal{D}_i}$ will be higher for a dataset with less-frequent high-performing configurations, resulting in higher probability of getting sampled for training.

Overall Objective Function We formulate the overall task representation problem as follows:

$$\min_{w_t, w_s, \theta} \max_{w_d} \quad \mathcal{L}_{\text{perf}}(w_t, w_s, \theta) + \alpha\mathcal{L}_{\text{sim}}(w_t) + \beta\mathcal{L}_{\text{adv}}(w_t, w_d), \tag{5}$$

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 that requires two loops. The outer-most for-loop (steps 3–12) runs for M' iterations sampling a

Algorithm 1: Offline Meta-learning Phase

Input meta-dataset $\mathcal{T}$, M' datasets to sample, hyperparameter batch size O_{hyper}, image batch size N_{img}, number of sampled image batches per dataset B_{img}, window size L
while *Not converge* **do**
  for *m=1 to M'* **do**
    Sample dataset $\mathcal{D}_m$ by balanced meta sampling (Eq. 4) ;
    Initialize global task embedding $\mathbf{t}_m^G = \mathbf{0}$;
    Sample hyperparameter batch $\mathcal{C}_m$ from $\mathcal{T}$ for $\mathcal{D}_m$;
    for *i=1 to* B_{img} **do**
      Sample an image batch $\mathcal{B}_m^i$ from $\mathcal{D}_m$;
      Update θ, w_t, w_s by minimizing Eq. (5) ;
      Compute $\mathbf{t}_m^G$ as mean of up to last L image batches;
      Update w_d by maximizing Eq. (5);
    end
  end
end

Algorithm 2: Online Recommendation Phase

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

dataset per iteration using balanced meta sampling (Eq. 4). For the mth sampled dataset $\mathcal{D}_m$, a meta-batch $\mathcal{C}_m$ is sampled 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 (Wong et al., 2018), 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 HyperSTAR learns to predict the performance score of a given dataset-configuration in the offline meta learning phase, we use HyperSTAR for online recommendation on an unseen dataset $\mathcal{D}_{\text{new}}$ (Algorithm 2 and Fig. 2b). We first extract a task representation $\mathbf{t}_{\text{new}} = g_{w_t}(\mathcal{D}_{\text{new}})$ for the new dataset. Then, along with encodings for a sequence of previously-seen hyperparameter configurations, we feed $\mathbf{t}_{\text{new}}$ into the offline-trained performance predictor $f_\theta(\cdot)$ to estimate the performance score of each configuration. Based on these performance scores, we rank the configurations to prioritize their evaluations.

Task-Aware HPO We make HPO methods task-aware by warm-starting them using the ranked list of hyperparameter configurations produced by HyperSTAR. We propose a task-aware variant of Hyperband Li et al. (2017) that replaces the random configuration sampling with the evaluation of the top n configurations based on the HyperSTAR recommendation. To reduce the risk of bad performance due to bad HyperSTAR predictions, our proposed Hyperband variant also includes random configurations. The number of considered random configurations is controlled by setting the desired fraction of the total n configurations to evaluate.

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 the average as the performance estimate for the nth configuration. Lastly, the algorithm ranks all the configurations based on their estimated performances and returns the ranking.

4 Experiments

This section presents experiments to evaluate HyperSTAR in terms of its performance predictor, hyperparameter recommendations, and as a warm-start to other HPO methods.

Datasets We evaluate HyperSTAR on 10 publicly-available large-scale image classification datasets: BookCover30 (Iwana et al., 2016), Caltech256 (Griffin et al., 2007), DeepFashion (Liu et al., 2016), Food-101 (Bossard et al., 2014), MIT Indoor Scene Recognition (Quattoni & Torralba, 2009), IP102 Insects Pests (Wu et al., 2019), Oxford-IIIT Pets (Parkhi et al., 2012), Places365 (Zhou et al., 2017), SUN397 (Xiao et al., 2010) and Describable Texture Dataset (DTD) (Cimpoi et al., 2014). We also evaluate HyperSTAR on 10 publicly-available real-world object detection datasets: Chess Pieces,[1] DeepFashion (Liu et al., 2016), Flickr32 (Romberg et al., 2011), IP102 Insects Pests Detection (Wu et al., 2019), KITTI Vision (Geiger et al., 2012), MOT17Det (Milan et al., 2016), Oktoberfest Food Dataset (Ziller et al., 2019), SKU-110k (Goldman et al., 2019), SVHN (Netzer et al., 2011) and Pascal VOC 2007 (Everingham et al., 2010).

Architectures To ensure that our empirical study reflects the performance of our method on state-of-the-art network

[1] Dataset available at http://bit.ly/3r16oIA.

architectures, we choose SE-ResNeXt-50 (Hu et al., 2018), a powerful and large architecture; and ShuffleNet-v2-x1 (Ma et al., 2018), a compact and efficient architecture for image classification tasks. For both networks, we operate in a transfer-learning setting (Donahue et al., 2014) where we initialize the weights of the network from a model pre-trained on ImageNet (Deng et al., 2009) 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.

For object detection, we choose YOLOv2 (Redmon & Farhadi, 2017) and Detectron-based Faster R-CNN (R50-FPN-x1) (Ren et al., 2015) to demonstrate the applicability of HyperSTAR on both state-of-the-art one-stage and two-state object detectors. The hyperparameter space of both YOLOv2 and Faster R-CNN consists of 48 hyperparameter configurations varying in learning rate, learning rate schedule, number of layers to finetune, data augmentation policy and input image size.

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 image classification datasets. We similarly train YOLOv2 and Faster R-CNN for every configuration in their respective hyperparameter space on each of the 10 object detection datasets. This generates a set of 400 training samples for SE-ResNeXt-50, 1,080 samples for ShuffleNet-v2-x1, 480 samples for YOLOv2, and 480 samples for Faster R-CNN. The meta-dataset thus contains triplets holding a one-hot encoding representing the hyperparameter configuration, training images of the dataset, and corresponding Top-1 accuracy for image classification and mean average precision (mAP@0.5) for object detection - these accuracies are used as performance scores that HyperSTAR estimates. We normalize these Top-1 accuracies and mAPs using the mean and standard deviation computed for each dataset separately over the performance scores across the configuration space (see Sect. 3.1). We adopt a leave-one-out evaluation protocol for all backbones and tasks (especially, leaving one dataset for testing while using the remaining for training). During the offline meta-learning phase, we freeze the feature extractor of models while only optimizing the performance predictor, task-similarity layer, visual encoder, and configuration encoder. Please refer to supplementary Sect. 7 for model architecture details. We also highlight that the training time is around 10 min. When compared to other HPO baselines (online phase) which usually take hours of training, the training cost of the performance predictor can be neglected.

Evaluation metric We use Average Precision@10 ($AP@10$) metric (see Sect. 3.1 and Zhu (2004)) to assess the ranking of configurations that HyperSTAR recommends. We first build a ground-truth list of hyperparameter configurations based on decreasing order of actual Top-1 accuracies or mAPs. We then compute $AP@10$ by comparing this list with the recommendation list generated based on the decreasing order of the predicted performance scores from HyperSTAR. Thus, $AP@10$ reflects the number and relative ranking of relevant configurations predicted by 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 (Feurer et al., 2015c; Bardenet et al., 2013) 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 SE-ResNeXt-50 and ShuffleNet-v2-x1 pretrained on ImageNet (Wong et al., 2018) for image classification, and YOLOv2 and Faster R-CNN pretrained on MS-COCO for object detection over all training images of a dataset. In comparison, our task representation is a batch-wise 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). We observe a similar trend for object detection where the respective performance gains are 12.54% and 7.08% for YOLOv2, and 12.37% and 8.22% for Faster R-CNN. This suggests that learning end-to-end visually inspired task representation enables 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 more effective than representing it as a point estimate using global mean.

Table 1 AP@10 comparison for SE-ResNeXt-50 for 10 public image classification datasets

Test dataset	BookCover30	Caltech256	DeepFashion	Food101	MIT Indoor	IP102 (pests)	Oxford-IIIT Pets	Places365	SUN397	Textures	Average
Baselines											
Feurer et al. (2015c)	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. (2015a)	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+Sim. (Similarity)	62.60	80.97	82.39	67.31	78.79	83.64	79.52	90.37	**94.47**	81.63	80.17
BM+Sim.+GAN	**68.27**	**86.72**	**91.51**	68.20	77.97	87.52	79.64	91.72	91.85	81.46	82.49
BM+Sim.+GAN+Balance	66.79	92.69	90.57	68.94	79.21	88.43	80.02	92.15	94.49	79.77	**83.31**

HyperSTAR outperforms all baselines by a large margin ($\geq$ 10.07%) with similarity & adversarial regularization and balanced meta sampling providing significant improvement on average

Table 2 AP@10 comparison for ShuffleNet-v2-x1 for 10 public image classification datasets

Test dataset	BookCover30	Caltech256	DeepFashion	Food101	MIT Indoor	IP102 (pests)	Oxford-IIIT Pets	Places365	SUN397	Textures	Average
Baselines											
Feurer et al. (2015c)	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. (2015a)	21.74	10.95	0.00	11.81	25.74	0.0	11.95	27.87	27.87	0.0	15.25
Task-Agnostic	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+Sim. (Similarity)	**80.23**	20.51	35.28	**87.10**	38.21	**57.73**	**54.69**	71.78	44.78	46.52	53.68
BM+Sim.+GAN	76.92	21.51	**40.10**	84.60	**47.74**	55.34	47.20	**75.25**	46.09	46.56	54.13
BM+Sim.+GAN+Balance	70.11	42.57	42.12	65.42	47.11	59.54	48.50	64.50	54.24	52.25	**54.64**

HyperSTAR outperforms all baselines by a large margin ($\geq$ 17.7%) with similarity & adversarial regularization and balanced meta sampling providing significant improvement on average

Performance Predictor Comparison We compare HyperSTAR with existing meta-learning-based warm-start HPO methods. We first compare HyperSTAR with (Feurer et al., 2015c) that uses random forest regression over a joint vector of meta-data and one-hot encoding of hyperparameters to predict the performance and use that to build a recommendation list of configurations. We also compare with (Feurer et al., 2015a) that finds the most similar training dataset for a given test dataset based on meta-data features and use the ground truth list of recommended configurations for the training dataset as the prediction for the test dataset. We further set up a task-agnostic baseline to compare the effectiveness of our task-aware recommendations. For this, we disregard the test dataset features and build a predicted list of recommended configurations by sorting the configurations in decreasing order of their average performance over the training datasets. From Tables 1, 2, 3 and 4, we can observe that HyperSTAR surpasses all the baselines with average $AP@10$ margin of at least 23% for SE-ResNeXt-50, 39% for ShuffleNet-v2-x1, 22% for YOLOv2 and 32% for Faster R-CNN. We also observe that similarity-based approaches (task-agnostic and Feurer et al. (2015a)) have a higher variance in $AP@10$ across datasets than task-representation-based approaches (HyperSTAR and Feurer et al. (2015c)).

Table 3 $AP@10$ comparison for YOLOv2 for 10 public object detection datasets across different methods

Test dataset	Chess Pieces	Deep Fashion	Flickr32	IP102 (pests)	KITTI	MOT17Det	Oktoberfest	SKU-110k	SVHN	VOC 2007	Average
Baselines											
Feurer et al. (2015c)	6.38	23.42	10.39	17.88	23.76	15.55	19.13	62.43	37.94	25.53	24.24
Feurer et al. (2015a)	26.25	10.58	65.39	46.31	47.46	33.22	33.22	65.39	47.46	46.31	42.16
Task-Agnostic	29.33	37.35	40.46	23.44	37.81	13.24	15.75	39.12	41.78	11.97	29.03
Meta-data	34.42	49.27	45.96	57.06	48.93	37.76	42.54	76.23	51.95	36.48	48.06
Global Mean	35.10	47.95	68.4	66.56	68.7	44.32	41.94	71.96	53.12	37.19	53.52
Ablations											
Batchwise Mean (BM)	32.46	49.44	67.22	72.79	68.34	54.42	44.40	78.25	54.45	42.54	56.43
BM+GAN	41.08	50.07	70.50	66.69	70.23	51.89	46.46	77.69	61.81	55.12	59.15
BM+Sim. (Similarity)	58.46	48.16	67.40	68.28	67.39	46.01	43.17	74.12	62.06	60.45	59.55
BM+Sim.+GAN	59.80	50.60	70.48	68.39	59.58	39.32	49.62	73.12	67.15	67.95	60.60
BM+Sim.+GAN+Bal. (Balance)	61.38	49.75	70.74	71.77	72.02	48.25	43.35	73.18	65.96	69.68	62.61
BM+Sim.+GAN+Bal.+Object Regions	**63.55**	49.10	**74.66**	70.89	71.56	44.49	44.32	77.76	**76.84**	**69.99**	**64.32**

HyperSTAR outperforms all baselines by a large margin (at least 10.8%) with similarity and adversarial regularization techniques along with balanced meta sampling. Object-specific features in task-representation adds a further average improvement of 1.71%

Table 4 $AP@10$ comparison for Faster R-CNN for 10 public object detection datasets across different methods

Test dataset	Chess Pieces	Deep Fashion	Flickr32	IP102 (Pests)	KITTI	MOT17Det	Oktoberfest	SKU-110k	SVHN	VOC 2007	Average
Baselines											
Feurer et al. (2015c)	11.26	31.51	10.52	42.25	21.07	10.91	20.22	2.64	5.60	0.67	15.67
Feurer et al. (2015a)	1.00	64.60	63.00	0.0	35.44	7.90	7.90	63.00	35.44	0.00	27.83
Task-Agnostic	7.46	17.86	42.67	2.11	23.30	8.15	20.13	41.88	35.30	0.00	19.89
Meta-data	15.71	27.73	53.70	55.94	41.28	22.15	50.89	63.11	62.90	9.52	40.29
Global Mean	14.30	21.11	57.55	68.32	47.81	23.76	55.51	78.04	67.33	10.65	44.44
Ablations											
Batchwise Mean (BM)	14.37	67.60	46.43	72.09	35.26	19.27	30.47	67.31	70.9	11.96	43.59
BM+GAN	14.84	66.82	52.50	68.55	41.21	15.31	48.01	66.10	63.45	7.87	44.47
BM+Sim. (Similarity)	16.05	65.91	55.24	59.56	41.35	19.90	48.00	78.00	63.11	12.22	45.93
BM+Sim.+GAN	27.80	**84.04**	69.46	65.48	43.96	33.56	55.80	61.93	55.42	29.10	52.66
BM+Sim.+GAN+Bal. (Balance)	24.75	48.00	**77.35**	53.52	53.96	26.56	49.76	**94.65**	78.90	**47.44**	55.49
BM+Sim.+GAN+Bal.+Region Prop	**27.42**	70.87	70.89	**73.59**	**55.85**	**30.56**	**64.94**	92.79	**86.46**	32.65	**60.60**

HyperSTAR outperforms all baselines by a large margin (at least 16.2%) with similarity and adversarial regularization techniques along with balanced meta sampling. Object-specific features in task-representation adds a further average improvement of 5.11%

4.2 Ablation Study

Regularization 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, 2, 3 and 4 that imposing the regularizations improves the $AP@10$ for 6 out of 10 datasets for SE-ResNeXt-50, 9 out of 10 datasets for ShuffleNet-v2-x1, 5 out of 10 datasets for YOLOv2 and all 10 out of 10 datasets for Faster R-CNN. This suggests that, on expectation, imposing regularization enables the task representations to learn more 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 recommend better hyperparameter configurations than a plain batchwise mean setting.

Balanced Meta Sampling We observe in Tables 1, 2, 3 and 4 that incorporating balanced meta sampling improves $AP@10$, on average, by 0.82%, 0.51%, 2.01% and 2.83% for SE-ResNeXt-50, ShuffleNet-v2-x1, YOLOv2 and Faster R-CNN respectively. The impact of balancing is more significant in object detection than image classification. This is because compared to image classification, the distribution

of hyperparameter configurations is more skewed for object detection, leading balanced meta sampling to result in a bigger improvement for object detection.

Object-specific Local Features We study the effect of including object-specific features in the task representation for object detection. As shown in Tables 3 and 4, we observe that incorporating ground truth based object regions for YOLOv2 and region proposals for Faster R-CNN leads to significant improvements (on average) in the $AP@10$ by 1.71% and 5.11%, respectively. We believe that the higher improvement for Faster R-CNN can be attributed to the availability of large number of region proposals that provide a stronger discriminative signal for the task information.

4.3 Warm-Starting with Recommendations

We evaluate the ranking order of configurations recommended by HyperSTAR and baseline methods by plotting a curve showing the best Top-1 accuracy or mAP achieved after k configurations for $k = 1 \ldots H$ at the budget of 1600 epochs as shown in Fig. 8a. We observe that HyperSTAR-based recommendation order 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 recommend configurations which is negligible compared to multiple GPU hours required to evaluate a single configuration.

4.4 Task-Aware Hyperband

We warm-start Hyperband (HB) (Li et al., 2017) using the task-aware hyperparameter recommendation from HyperSTAR and compare it with the vanilla Hyperband (Li et al., 2017) and BOHB (Falkner et al., 2018). 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. 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 and datasets. 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, 250 and 100 epochs ($\eta = 2$, low-budget settings). Since the Detectron-based Faster R-CNN does not define an epoch, the budget is defined in terms of iterations per one full trial with number of iterations remaining the same for a given dataset across all hyperparameters. For instance, a budget of 1600% denotes training equivalent to running 16 configurations. Figures 4 and 5 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. Figure 8b 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. We see a similar trend for object detection in Figs. 6, 7 and 8b. HyperSTAR outperforms vanilla HB and BOHB in the low-budget settings for all datasets achieving around 11.8% and 8.0% higher best $mAP@0.5$ on average for the smallest budget setting on YOLOv2 and Faster R-CNN respectively.

The difference in the Top-1 accuracy or mAP achieved by all three methods gradually diminish with increasing time budget and eventually becomes negligible for the largest budget setting. The accuracy is also on par with the best Top-1 accuracy and mAP achievable for the given hyperparameter space. This happens for vanilla HB and BOHB because over time, they explore the hyperparameter space sufficiently enough to discover the best possible configuration. Although HyperSTAR-based Hyperband aims to improve HPO efficiency for low-budget settings, being able to achieve the best possible performance suggests that it is also sound for large-budget settings. Given sufficient budget, our method can achieve on par (if not better) performance compared to other HPO methods. Our plots also show BOHB being comparable to vanilla HB for low-budget settings while being better than vanilla HB in the large-budget settings. 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 (Xue et al., 2019). 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 or mAP achieved over the unseen dataset using the best configuration for the most similar training dataset. 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 (1100% of total iterations/training for Faster R-CNN). As shown in Figs. 4, 5,

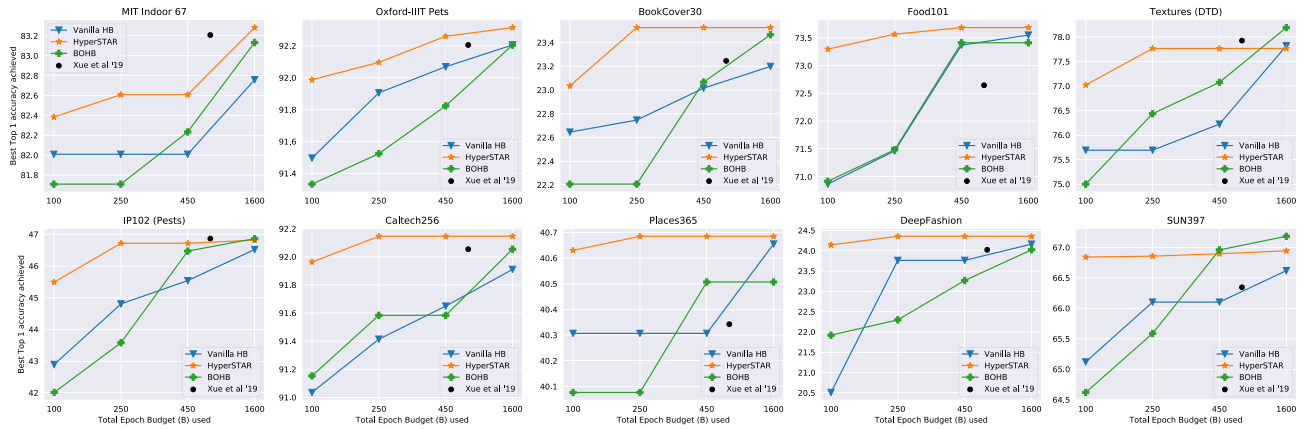

Fig. 4 End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for SE-ResNeXt-50. HyperSTAR outperforms other methods when performing on low epoch budgets (100, 250, 450)

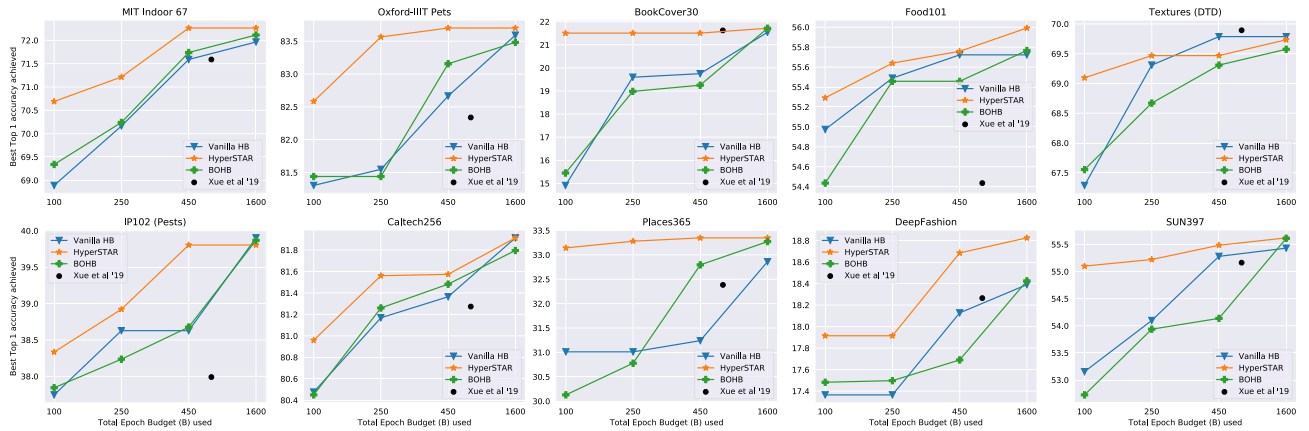

Fig. 5 End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for ShuffleNet-v2-x1. HyperSTAR outperforms other methods when performing on low epoch budgets (100, 250, 450)

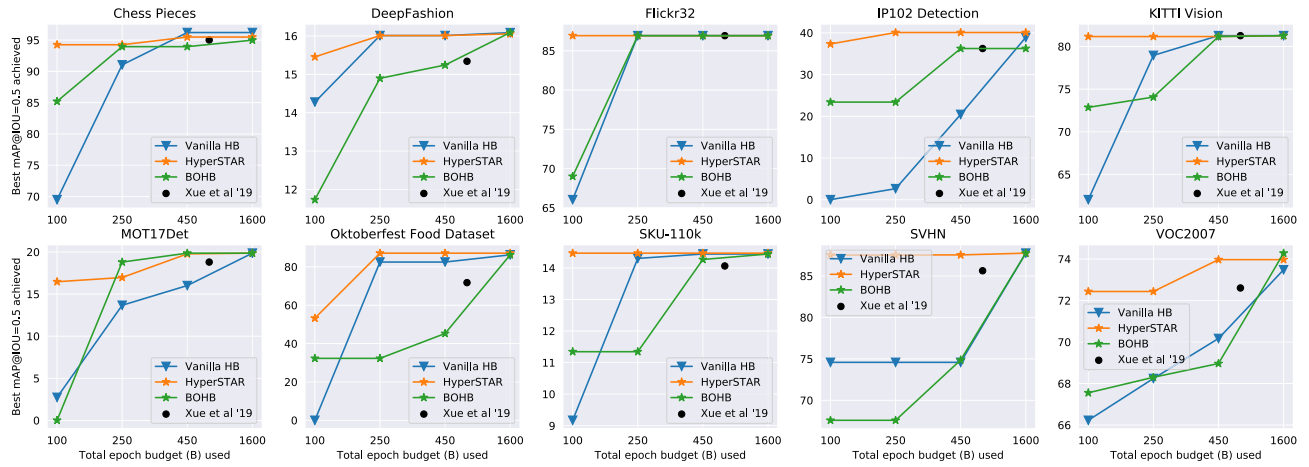

Fig. 6 End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for Yolo V2. HyperSTAR outperforms other methods when performing on low epoch budgets (100, 250, 450)

6, 7 and 8b, we observe that, on expectation, our task-aware HB achieves the same performance in as little as 10 times less budget than that of Tr-AutoML. This reinforces that learning a dataset embedding from raw pixels reduces the time required to predict the optimal hyperparameters compared to Tr-AutoML.

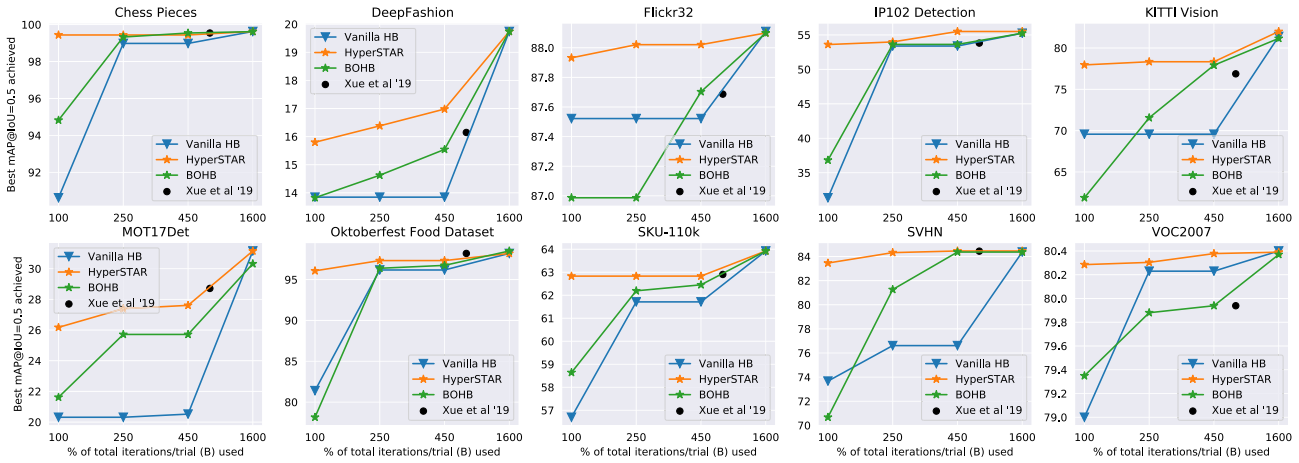

Fig. 7 End-to-End performance comparison of task-aware HyperSTAR based Hyperband with existing methods for Detectron-based Faster RCNN (FPN-R50-X1). HyperSTAR outperforms other methods when performing on low budgets (100%, 250%, 450%)

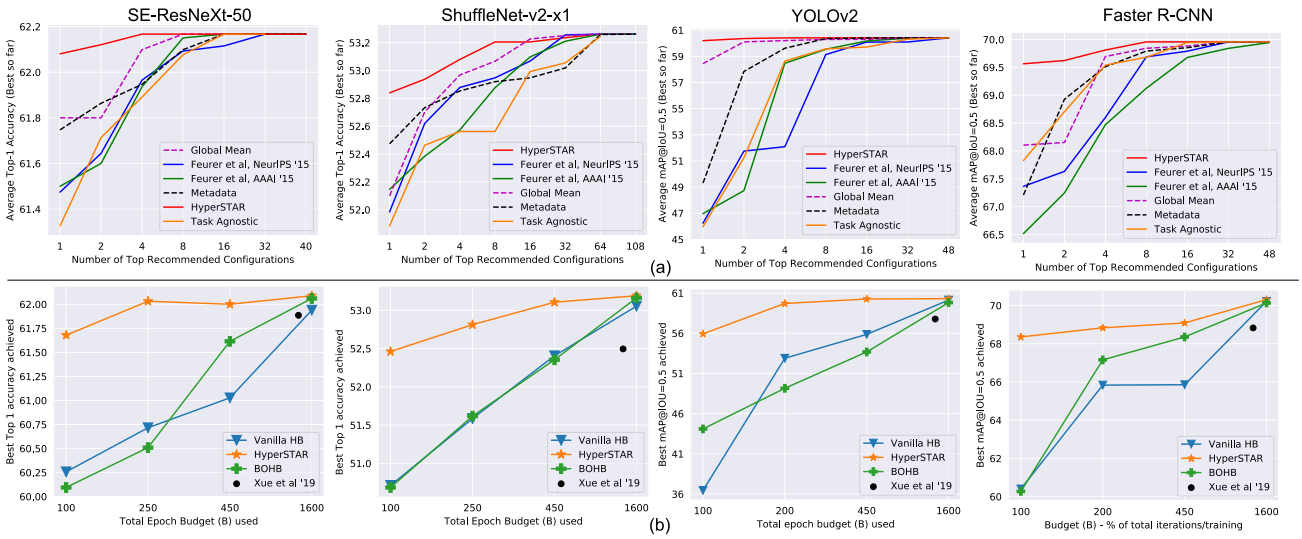

Fig. 8 **a** Comparison of evaluating configurations recommended by HyperSTAR with baseline methods in ranked order. Compared to baselines, HyperSTAR achieves the best performance by evaluating 50% less configurations. **b** Comparison of warm-starting Hyperband (HB) with HyperSTAR vs. baseline approaches across different epoch budgets. HyperSTAR achieves optimal accuracy in 25% of the budget required by other methods. For the smallest budget setting, it also achieves 1.5% higher 'Best Top-1 accuracy on average' on both image classification models, and 11.8% and 8.0% higher 'Best mAP@0.5 on average' on YOLOv2 and Faster-RCNN respectively for object detection

4.5 Recommending Model Pruning Parameters

This extension shows that HyperSTAR can recommend model pruning hyperparameters besides model training parameters, such as, learning rate, number of layers to fine tune, and optimizer. The experiment demonstrates pruning channels from a YOLOv2 model. After training the YOLOv2 model, we perform channel pruning on all but the last convolutional layer using the channel importance estimation method from (Molchanov et al., 2019). At every pruning iteration, all channels in the selected convolutional layers are ranked based on an importance estimation; then the K least important channels are pruned followed by P epochs of fine-tuning. We set the compression ratio to 50%, i.e., only half of the original channels will remain after pruning. K and P are our pruning hyperparameters that HyperSTAR can optimize. Figure 9 shows a comparison of hyperparameter recommendation by HyperSTAR with other baselines. We can observe that HyperSTAR achieves close to optimal accuracy in 25% or less of the budget required by other methods (i.e. 2 configurations vs. 8 or more configurations).

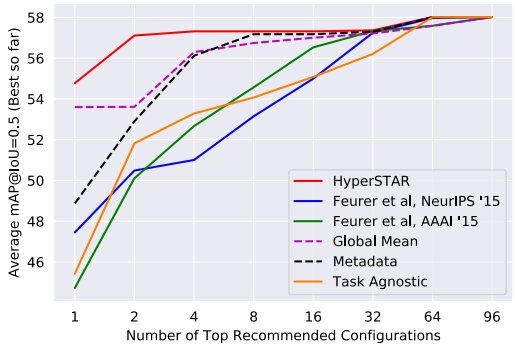

Fig. 9 Comparison of evaluating configurations recommended by HyperSTAR with baseline methods in ranked order for model pruning on YOLOv2. HyperSTAR achieves optimal accuracy in 25% or less of the budget required by other methods

5 Conclusion

We present HyperSTAR, the first efficient task-aware warm-start algorithm for hyperparameter optimization (HPO) on visual recognition tasks. HyperSTAR operates by learning directly from raw images an end-to-end task representation and a performance predictor that produces a ranking of hyperparameter configurations. We show that this ranking accelerates HPO algorithms such as Hyperband (HB). We also present a meta-balance method that reduces the bias of dominant configurations in the meta-dataset, yielding a more effective hyperparameter recommendation. Our experiments on real-world image classification and object detection 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. Also, our experiments show that HyperSTAR combined with Hyperband achieves the optimal performance in only 25% of the budget required by other HB variants for image classification and object detection tasks. Our experiments show that HyperSTAR is beneficial when the computational time budget is tight as it accelerates the performance of HPO methods; and that it matches the performance of other HPO methods when the computational time budget is large. Lastly, our experiments show that the considered hyperparameters to optimize can range from model training (e.g., learning rate) to those of model compression (e.g., number of channels to prune). We conclude that HyperSTAR can be a crucial component for services that automatically train models by recommending effective, task-aware hyperparameters to reduce cost. In future work, we could explore making the hyperparameter configurations into a continuous space where the generalization to unseen configurations can be feasible.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11263-023-01961-0.

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

References

Achille, A., Lam, M., & Tewari, R. et al. (2019). Task2vec: Task embedding for meta-learning. In *Proceeding of IEEE ICCV*.

Bardenet, R., Brendel, M., & Kégl, B., et al. (2013). Collaborative hyperparameter tuning. In *Proceedings of ICML*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(Feb), 281–305.

Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101-mining discriminative components with random forests. In *Proceedings of ECCV*.

Chen, W., Liu, T. Y., & Lan, Y., et al. (2009). Ranking measures and loss functions in learning to rank. In *Proceeding of NeurIPS*.

Cimpoi, M., Maji, S., & Kokkinos, I., et al. (2014). Describing textures in the wild. In *Proceeding of IEEE CVPR*.

Deng, J., Dong, W., & Socher, R., et al. (2009). Imagenet: A large-scale hierarchical image database. In *Proceeding of IEEE CVPR*.

Donahue, J., Jia, Y., & Vinyals, O., et al. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceeding of ICML*.

Everingham, M., Van Gool, L., Williams, C. K., et al. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision, 88*(2), 303–338.

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceeding of ICML*.

Feurer, M., Klein, A., & Eggensperger, K., et al. (2015a). Efficient and robust automated machine learning. In *Proceeding of NeurIPS*.

Feurer, M., Klein, A., & Eggensperger, K., et al. (2015b). Efficient and robust automated machine learning. In *Proceeding of NeurIPS*.

Feurer, M., Springenberg, J., & Hutter, F. (2015c). Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceeding of the AAAI Conf. on AI*.

Franceschi, L., Donini, M., & Frasconi, P., et al. (2017). Forward and reverse gradient-based hyperparameter optimization. In *Proceeding of ICML*.

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceeding of ICML*.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceeding of IEEE CVPR*.

Girshick, R. (2015). Fast r-cnn. In *Proceeding of IEEE ICCV*.

Goldman, E., Herzig, R., & Eisenschtat, A. et al. (2019). Precise detection in densely packed scenes. In *Proceeding of IEEE CVPR*.

Griffin, G., Holub, A., & Perona, P. (2007). Caltech-256 object category dataset.

He, K., Gkioxari, G., & Dollár, P., et al. (2017). Mask r-cnn. In *Proceeding of IEEE ICCV*.

He, K., Zhang, X., & Ren, S. et al. (2016). Deep residual learning for image recognition. In *Proceeding of IEEE CVPR*.

Hoffman, J., Tzeng, E., & Park, T. et al. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *Proceeding of ICML*.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceeding of IEEE CVPR*.

Huang, G., Liu, Z., & Van Der Maaten, L. et al. (2017). Densely connected convolutional networks. In *Proceeding of IEEE CVPR*.

Hutter, F., Kotthoff, L., & Vanschoren, J. (eds) (2018). *Automatic machine learning: Methods, systems, challenges*. Springer (in press), available at http://automl.org/book.

Iwana, B. K., Raza Rizvi, S. T., & Ahmed, S., et al. (2016). Judging a book by its cover. arXiv:1610.09204.

Jamieson, K., & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Proceeding of AISTATS*.
Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceeding of ACM KDD*.
Kandasamy, K., Dasarathy, G., & Schneider, J., et al. (2017). Multi-fidelity bayesian optimisation with continuous approximations. In *Proceeding of ICML*.
Kim, J., Kim, S., & Choi, S. (2017). Learning to warm-start bayesian hyperparameter optimization. arXiv:1710.06219.
Klein, A., Falkner, S., & Bartels, S. et al. (2017). Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Proceeding of AISTATS*.
Klein, A., Falkner, S., & Springenberg, J. T., et al. (2016). Learning curve prediction with Bayesian neural networks. *ICLR*.
Kokiopoulou, E., Hauth, A., & Sbaiz, L., et al. (2019). Fast task-aware architecture inference. arXiv:1902.05781
Kozerawski, J., Fragoso, V., & Karianakis, N. et al. (2020). BLT: Balancing long-tailed datasets with adversarially-perturbed images. In *Proceeding of ACCV*.
Li, H., Fowlkes, C., & Yang, H., et al. (2023). Guided recommendation for model fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3633–3642).
Li, L., Jamieson, K., & DeSalvo, G., et al. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*.
Lindauer, M., & Hutter, F. (2018). Warmstarting of model-based algorithm configuration. In *Proceeding of the AAAI Conference on AI*.
Liu, Z., Luo, P., & Qiu, S., et al. (2016). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceeding of IEEE CVPR*.
Ma, N., Zhang, X., Zheng, H. T., et al. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceeding of ECCV*.
Maclaurin, D., Duvenaud, D., & Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In *Proceeding of ICML*.
Milan, A., Leal-Taixé, L., & Reid, I., et al. (2016). Mot16: A benchmark for multi-object tracking. arXiv:1603.00831
Mittal, G., Liu, C., & Karianakis, N. et al. (2020). Hyperstar: Task-aware hyperparameters for deep networks. In *Proceeding of IEEE/CVF CVPR*.
Molchanov, P., Mallya, A., & Tyree, S., et al. (2019). Importance estimation for neural network pruning. In *Proceeding of IEEE CVPR*.
Netzer, Y., Wang, T., & Coates, A. et al. (2011). Reading digits in natural images with unsupervised feature learning.
Parkhi, O. M., Vedaldi, A., & Zisserman, A. et al. (2012). Cats and dogs. In *Proceeding of IEEE CVPR*.
Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *Proceeding of ICML*.
Perrone, V., Jenatton, R., & Seeger, M. W. et al. (2018). Scalable hyperparameter transfer learning. In *Proceeding of NeurIPS*.
Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *Proceeding of IEEE CVPR*.
Redmon, J., & Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceeding of IEEE CVPR*.
Ren, S., He, K., & Girshick, R. et al. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceeding of NeurIPS*.
Romberg, S., Pueyo, L. G., & Lienhart, R. et al. (2011). Scalable logo recognition in real-world images. In *Proceeding of ACM ICMR*.
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. CoRR https://arxiv.org/abs/1409.1556v6
Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceeding of NeurIPS*.
Snoek, J., Rippel, O., & Swersky, K. et al. (2015). Scalable bayesian optimization using deep neural networks. In: *Proceeding of ICML*.
Swersky, K., Snoek, J., & Adams, R. P. (2013). Multi-task bayesian optimization. In *Proceeding of NeurIPS*, (pp. 2004–2012).
Swersky, K., Snoek, J., & Prescott Adams, R. (2014). Freeze–thaw Bayesian optimization. arXiv:1406.3896
Tzeng, E., Hoffman, J., & Saenko, K., et al. (2017). Adversarial discriminative domain adaptation. In *Proceeding of IEEE CVPR*.
Vaswani, A., Shazeer, N., & Parmar, N., et al. (2017). Attention is all you need. In *Proceeding of NeurIPS*.
Wang, Y. X., Ramanan, D., & Hebert, M. (2017). Learning to model the tail. In *Proceeding of NeurIPS*.
Wong, C., Houlsby, N., & Lu, Y., et al. (2018). Transfer learning with neural automl. In *Proceeding of NeurIPS*.
Wu, X., Zhan, C., & Lai, Y. K., et al. (2019). Ip102: A large-scale benchmark dataset for insect pest recognition. In *Proceeding of IEEE CVPR*.
Xiao, J., Hays, J., & Ehinger, K. A. et al. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Proceeding of IEEE CVPR*.
Xiao, Y., Xing, E. P., & Neiswanger, W. (2021). Amortized auto-tuning: Cost-efficient bayesian transfer optimization for hyperparameter recommendation. arXiv:2106.09179
Xu, H., Kang, N., & Zhang, G. et al. (2021). Nasoa: Towards faster task-oriented online fine-tuning with a zoo of models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (pp. 5097–5106).
Xue, C., Yan, J., & Yan, R., et al. (2019). Transferable automl by model sharing over grouped datasets. In *Proceeding of IEEE CVPR*.
Yan, C., Zhang, Y., & Zhang, Q., et al. (2022). Privacy-preserving online automl for domain-specific face detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (pp. 4134–4144).
Yang, D., Myronenko, A., & Wang, X. et al. (2021). T-automl: Automated machine learning for lesion segmentation using transformers in 3d medical imaging. In *Proceedings of the IEEE/CVF international conference on computer vision*, (pp. 3962–3974).
Yogatama, D., & Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Proceeding of AISTATS*.
Zhou, K., Hong, L., & Hu, S. et al. (2021). Dha: End-to-end joint optimization of data augmentation policy, hyper-parameter and architecture. arXiv:2109.05765
Zhou, B., Lapedriza, A., & Khosla, A., et al. (2017). Places: A 10 million image database for scene recognition. *T-PAMI*.
Zhu, M. (2004). Recall, precision and average precision. *Dept of Statistics and Actuarial Science, Univ of Waterloo, 2*(30), 6.
Ziller, A., Hansjakob, J., & Rusinov, V. et al. (2019). Oktoberfest food dataset. arXiv:1912.05007