Zero-Shot AutoML with Pretrained Models

Ekrem Öztürk [*1] **Fabio Ferreira** [*1] **Hadi S. Jomaa** [*2] **Lars Schmidt-Thieme** [2] **Josif Grabocka** [1]
Frank Hutter [1 3]

Abstract

Given a new dataset D and a low compute budget, how should we choose a pre-trained model to fine-tune to D, and set the fine-tuning hyperparameters without risking overfitting, particularly if D is small? Here, we extend automated machine learning (AutoML) to best make these choices. Our domain-independent meta-learning approach learns a zero-shot surrogate model, which, at test time, allows to select the right deep learning (DL) pipeline (including the pre-trained model and fine-tuning hyperparameters) for a new dataset D given only trivial meta-features describing D, such as image resolution or the number of classes. To train this zero-shot model, we collect performance data for many DL pipelines on a large collection of datasets and meta-train on this data to minimize a pairwise ranking objective. We evaluate our approach under the strict time limit of the vision track of the ChaLearn AutoDL challenge benchmark, clearly outperforming all challenge contenders.

1. Introduction

A typical problem in deep learning (DL) applications is to find a good model for a given dataset D in a restrictive time budget. In the case of tabular data, a popular approach for solving this problem is automated machine learning (AutoML), as implemented, e.g., in Auto-sklearn (Feurer et al., 2015a) or Auto-Gluon (Erickson et al., 2020). However, in domains such as computer vision and natural language processing, a better solution, especially under low resource constraints, is typically to fine-tune an existing pre-trained model. This, at first glance, appears to render AutoML unnecessary for those domains. However, as we will demonstrate in this paper, AutoML and pre-trained models can be combined to yield much stronger performance than either of them alone.

A great advantage of fine-tuning pre-trained models is strong anytime performance: the use of pre-trained models often allows to obtain very strong performance orders of magnitude faster than when training a model from scratch. In many practical applications, this strong anytime performance is crucial, e.g., for DL-based recognition systems in manufacturing, or automated DL (AutoDL) web services. The clock starts ticking as soon as a new dataset is available, and it would be far too costly to train a new model from scratch, let alone optimize its hyperparameters. The recent ChaLearn AutoDL competition (Liu et al., 2021) mimicked these tight time constraints, rewarding performance found in an anytime fashion.

While fine-tuning pre-trained models enjoys strong anytime performance, it also introduces many additional degrees of freedom. Firstly, we need to select a pre-trained network to fine-tune. To obtain good anytime performance, we may even want to start by training a shallow model to obtain good results quickly, and at some point switch to fine-tuning a deeper model. There are many additional degrees of freedom in this fine-tuning phase, concerning learning rates, data augmentation, and regularization techniques. We refer to the combination of a pre-trained model and the fully specified fine-tuning phase, including its hyperparameters, as a *DL pipeline*. Which DL pipeline works best depends heavily on the dataset, for instance, datasets with high-resolution images may favor the use of more downsampling layers than the low-resolution images of the CIFAR dataset (Krizhevsky et al., 2009); likewise, datasets with few images may favor smaller learning rates. We, therefore, require an automated method for selecting the best DL pipeline based on the characteristics of the dataset at hand.

In this paper, we tackle this problem by meta-learning a model across datasets that enables zero-shot DL pipeline selection. Specifically, we create a meta-dataset holding the performance of many DL pipelines on a broad range of datasets. Using this meta-dataset, we then learn a function that selects the right DL pipeline based on the properties of the dataset (e.g., the image resolution and the number of images) in a zero-shot setting. To learn this selection function, we first formulate DL pipeline selection as a classical

[*]Equal contribution [1]University of Freiburg [2]University of Hildesheim [3]Bosch Center for Artificial Intelligence. Correspondence to: Fabio Ferreira <ferreira@cs.uni-freiburg.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

algorithm selection (AS) problem (Rice, 1976) and then improve upon this formulation by recognizing DL pipelines as points in a geometric space that allows information about the performance of some pipelines to inform performance on others. We then train a deep neural network with a pairwise ranking objective to emphasize the rank of the DL pipeline predicted to perform best in a manner that automatically normalizes across datasets. Note, that we use the *zero-shot* nomenclature not to refer to samples of unseen classes but to express that we cannot even afford a single exploratory evaluation of a pipeline but need to directly select a suitable one in a zero-shot manner.

Our contributions can be summarized as:

- We extend AutoML to best exploit pre-trained models by meta-learning to select the best DL pipeline conditional on dataset meta-features.
- We introduce a large meta-dataset with the performances of 525 DL pipelines across 35 image-classification datasets and 15 augmentations each. With 525*35*15 entries, it is, to our best knowledge, the first DL meta-dataset for image classification of this size, being over 1000 times larger than previous meta-datasets (Triantafillou et al., 2019).
- We go beyond a standard formulation as an algorithm selection problem by formulating the new problem of selecting a DL pipeline as a point in a geometric space to exploit similarities between DL pipelines.
- We introduce a novel zero-shot AutoDL method that addresses this pipeline selection problem with a pairwise ranking loss.
- In the setting of the recent ChaLearn AutoDL challenge (Liu et al., 2021), our zero-shot AutoDL approach dominates all competitors on a broad range of 35 image datasets, as well as in the challenge itself.

To foster reproducibility, we make our PyTorch (Paszke et al., 2019) code, models, and data publicly available under this URL.

2. Related Work

Algorithm selection Assume a set $\mathcal{P}$ of algorithms $\mathcal{A} \in \mathcal{P}$ (e.g., classifiers or neural network hyperparameter configurations), a set of *instances* $i \in \mathcal{I}$ (e.g., dataset features), and a *cost metric* $m : \mathcal{P} \times \mathcal{I} \to \mathbb{R}$. Specifying the loss of algorithm $\mathcal{A} \in \mathcal{P}$ on instance $i \in \mathcal{I}$ with $m(\mathcal{A}, i)$, the algorithm selection problem (Rice, 1976; Smith-Miles, 2009; Kotthoff et al., 2012; Bischl et al., 2016) is to find a mapping $s : \mathcal{I} \to \mathcal{P}$ that minimizes the cost metric $\sum_{i \in \mathcal{I}} m(s(i), i)$ across instances $\mathcal{I}$. Algorithm selection has been applied to achieve state-of-the-art results in many hard combinatorial problems, most prominently Boolean satisfiability solving (SAT), where SATzilla (Xu et al., 2008) won several competitions by learning to select the best SAT solver on a per-instance basis. There are many methods for solving algorithm selection, based on regression (Xu et al., 2008), k-nearest neighbours (Kadioglu et al., 2011), cost-sensitive classification (Xu et al., 2012), and clustering (Kadioglu et al., 2010; Malitsky et al., 2013). *AutoFolio* (Lindauer et al., 2015) is a state-of-the-art algorithm selection system that combines all of these approaches in one and chooses between them using algorithm configuration (Hutter et al., 2011). We will use AutoFolio as one of our methods for selecting DL pipelines based on dataset meta-features.

Hyperparameter optimization (HPO) HPO plays an integral role in fine-tuning any machine learning algorithm. Beyond simple strategies, such as random search (Bergstra & Bengio, 2012), conventional techniques typically involve fitting (probabilistic) surrogate models of the true response, e.g. Gaussian Process (Rasmussen & Williams, 2006), random forests (Hutter et al., 2011), neural networks (Springenberg et al., 2016), or some hybrid techniques (Snoek et al., 2015), and selecting configurations that optimize predefined acquisition functions (Wilson et al., 2018). Recently, approaches started taking into account the dissimilarity between pre-training and downstream domains (Li et al., 2020). HPO multi-fidelity methods further reduce the wall-clock time necessary to arrive at optimal configurations (Li et al., 2017; Falkner et al., 2018; Awad et al., 2021).

Transfer HPO Transfer learning can be used in HPO to leverage knowledge from previous experiments to yield a strong surrogate model with few observations on the target dataset. For example, Wistuba & Grabocka 2021 and Jomaa et al. 2021a both propose a meta-initialization strategy by optimizing a deep kernel Gaussian process surrogate model (Wilson et al., 2016) across meta-train datasets to allow for fast adaptation given a few observations. Similarly, Salinas et al. 2020 learns a Gaussian Copula (Wilson & Ghahramani, 2010) and addresses the heterogeneous scales of the responses across datasets, whereas Perrone et al. 2018 pre-trains a shared layer in a multi-task setting. Transfer HPO is also possible based on meta-features (Vanschoren, 2018), i.e. dataset characteristics which can be either engineered (Feurer et al., 2015b; Wistuba et al., 2016) or learned (Jomaa et al., 2021b) to warm-start HPO.

Zero-shot HPO The conventional setting of zero-shot learning aims to recognize samples whose instances may not have been seen during training (Xian et al., 2018; Verma et al., 2018; Radford et al., 2021). In the setting of zero-shot HPO, in contrast, the focus lies on improving sample efficiency for hyperparameter optimization. Contrary to techniques in previous sections that improve their sample efficiency by increasing the number of trials, zero-shot HPO

has emerged as a more efficient approach that does not require any observations of the response on the target dataset. Wistuba et al. 2015 design a sequential model-free approach that optimizes the ranks of hyperparameter configurations based on their average performance over a collection of datasets. Winkelmolen et al. 2020 propose a Bayesian optimization solution for zero-shot HPO, whereby a surrogate model is fit to the dataset and hyperparameters and optimized by minimizing a ranking meta-loss. We note that both these approaches return a fixed set of hyperparameter configurations without using meta-features, which is undesirable as the AutoDL setting used in this work only allows for running a single model. Related to our work is (Tornede et al., 2020), who also describe datasets and pipelines as joint feature vectors. They use these to assess the learning of zero-shot models with algorithm selection and ranking-based objectives in a benchmark of tabular datasets and shallow base models in a sparse cost-matrix setting. In this paper, we propose a novel zero-shot HPO solution inspired by the success of algorithm selection techniques that learns to select the best DL pipeline based on both parametric choices inside the DL pipeline and dataset meta-features of complex vision datasets, by optimizing a ranking objective jointly across datasets.

AutoDL Competition ChaLearn's AutoDL Challenge (Liu et al., 2021) evaluated competitors in an anytime setting with strict time limits, leading to the prominent use of pre-trained models by the participants. We focus on the challenge's image-track and summarize the winning approaches here and give more details in Section 5.3. In the 2019 AutoCV/CV2 competition, the winning approach (Baek et al., 2020) used a ResNet-18 (He et al., 2015) pre-trained on ImageNet (Krizhevsky et al., 2012) with Fast AutoAugment (Lim et al.). All image-track winning solutions used the AutoCV winner code as a skeleton and built their methods on top. Their modifications ranged from switching to a more stable ResNet-9 during training (DeepWisdom), ensembling predictions (DeepBlueAI) to data-adaptive pre-processing (PASA NJU).

Meta-learning Meta-learning (Finn et al., 2017) can be used to solve tasks where the training dataset is small. Sun et al. 2019 meta-learn to transfer large-scale pre-trained DNN weights to solve few-shot learning tasks. Verma et al. 2019 tackle Zero-Shot Learning by meta-learning a generative model for synthesizing examples from unseen classes conditioned on class attributes. Laadan et al. 2019 generate (shallow model) pipelines on diverse datasets and use dataset meta-features to rank the pipelines to create a meta-dataset of pipelines and their performance results.

Despite the abundance of meta-learning methods, and in contrast to the large benchmarks for tabular data (Pineda-Arango et al., 2021), few meta-learning benchmarks exist for image datasets. Zhai et al. 2019 introduced a set of 19 vision tasks and evaluated 18 representation learning methods. Triantafillou et al. 2019 also introduced a meta-dataset of 10 few-shot image tasks and a growing set of baselines, currently comprising 11 and 18 evaluations on two different settings. Dumoulin et al. 2021 combines these two benchmarks and compares Big Transfer (Kolesnikov et al., 2020) against the baselines of Triantafillou et al. 2019. As we will show, our DL meta-dataset for image tasks is far larger than all previous meta-learning benchmarks.

3. Zero-Shot AutoML with Pretained Models

3.1. Notation and Problem Definition

Let $\mathcal{X} := \{x_n\}_{n=1}^N$ denote a set of N distinct deep learning (DL) pipelines. Every DL pipeline $x_n := (M_n, \theta_n)$ comprises a pre-trained model $M_n \in \mathcal{M}$ and fine-tuning hyperparameters $\theta_n \in \Theta$ that are used to fine-tune M_n to a given dataset. Furthermore, let $\mathcal{D} = \{D_i\}_{i=1}^I$ denote a collection of I datasets, where each dataset $D_i \in \mathcal{D}$ is split into disjoint training, validation and testing subsets $D_i := D_i^{(\text{tr})} \cup D_i^{(\text{val})} \cup D_i^{(\text{test})}$. For each dataset, we are given a vector of K descriptive characteristics (a.k.a. meta-features), such as the number of data points and the image resolution, as $\phi_i \in \Phi \subseteq \mathbb{R}^K$ (see Section 4.1 for the full set of meta-features we used in our experiments). We denote by $x_n^{(\text{ft})} := \text{Tune}\left(x_n, D^{(\text{tr})}, D^{(\text{val})}\right)$ the model resulting from fine-tuning the pre-trained model M_n with hyperparameters θ_n on training data $D^{(\text{tr})}$ using validation data $D^{(\text{val})}$ for early stopping. Then, denoting the loss of a fine-tuned model $x_n^{(\text{ft})}$ on the test split of the same dataset D as $\mathcal{L}(x_n^{(\text{ft})}, D^{(\text{test})})$, the test cost of DL pipeline x_n on D is defined as:

$$C(x_n, D) = \mathcal{L}\left(\text{Tune}\left(x_n, D^{(\text{tr})}, D^{(\text{val})}\right), D^{(\text{test})}\right). \quad (1)$$

Definition 1. *Given a set of N DL pipelines $\mathcal{X} := \{x_n\}_{n=1}^N$ and a collection of I datasets $\mathcal{D} = \{D_i\}_{i=1}^I$ with meta-features ϕ_i for dataset $D_i \in \mathcal{D}$, and a $N \times I$ matrix of costs $C(x_n, D_i)$ representing the cost of pipeline x_n on dataset D_i, the problem of **zero-shot AutoML with pre-trained models (ZAP)** is to find a mapping $f : \Phi \to \mathcal{X}$ that yields minimal expected cost over $\mathcal{D}$:*

$$\text{argmin}_f \ \mathbb{E}_{i\sim\{1,\dots,I\}}\left[C(f(\phi_i), D_i)\right]. \quad (2)$$

3.2. ZAP via Algorithm Selection (ZAP-AS)

The problem of zero-shot AutoML with pre-trained models from Definition 1 can be directly formulated as an algorithm selection problem: the DL pipelines $\mathcal{X} := \{x_n\}_{n=1}^N$ are the algorithms $\mathcal{P}$, the datasets $\{D_i\}_{i=1}^I$ are the instances $\mathcal{I}$, and the test cost $C(x_n, D)$ of DL pipeline x_n on D defines the cost metric $m : \mathcal{P} \times \mathcal{I} \to \mathbb{R}$. We use the state-of-the-art

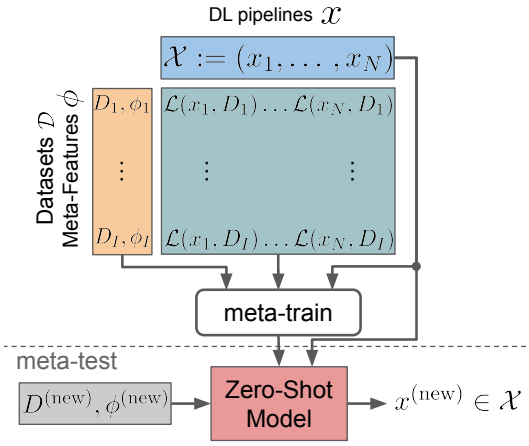

Figure 1. ZAP consists of two stages. In the meta-train stage, the cost matrix on the source tasks is leveraged to learn a joint response surface conditioned on the meta-features and pipelines. During the meta-test stage, ZAP assigns scores to the pipelines of the unseen datasets and the highest-scoring pipeline is selected.

algorithm selection system AutoFolio to learn a selector between our predefined DL pipelines, since it subsumes approaches based on regression, classification, clustering, and cost-sensitive classification, and selects the best one for the data at hand (Lindauer et al., 2015).

While this formulation of zero-shot AutoML with pre-trained models as algorithm selection will turn out to already yield very strong performance, it has one limitation: algorithm selection abstracts away our DL pipelines as uncorrelated algorithms, losing all information about the pre-trained models they are based on, and which hyperparameters are being used for fine-tuning. This information, e.g., allows us to predict the cost of DL pipelines to other DL pipelines with similar settings without ever having run them. Thus, we next introduce a novel approach for exploiting this knowledge.

3.3. ZAP via Zero-Shot HPO (ZAP-HPO)

We now describe a variant of our formulation of zero-shot AutoML that exploits the fact that the DL pipelines between which we select are points in a geometric space, and that we can see the space of DL pipelines we consider as a search space for hyperparameter optimization (HPO), with a categorical value for the choice of pre-trained model and continuous fine-tuning hyperparameters; we can then use concepts from zero-shot HPO to tackle this problem.

We define M as a finite collection of N pre-trained models and represent each instance, M_n, as a one-hot encoded vector, and $\theta_n \in \Theta \subseteq \mathbb{R}^L$ as a vector of continuous variables defining its respective hyperparameters. For instance, Θ can represent the continuous space of learning rates and dropout values of a pre-trained neural network model in $\mathcal{M} \in \{0,1\}^{|N|}$. Consequently, the DL pipelines are projected to the geometric space defined by $\mathcal{X} \subseteq \mathcal{M} \times \Theta$ and can be viewed as a hyperparameter configuration space where pre-trained models are simply categorical variables.

Denote by f_ψ a parametric surrogate with parameters ψ that estimates the test cost observed by fine-tuning the DL pipeline x_j on dataset D_i with meta-features ϕ_i. The surrogate captures the fusion of (i) pipeline hyperparameters (i.e. x represented by the pre-trained model's one-hot-encoding indicator $\mathcal{M} \in \{1, \ldots, M\}$ and the fine-tuning hyperparameters $\theta \in \Theta$) with (ii) dataset meta-features ϕ, in order to estimate the cost after fine-tuning. Formally, that is:

$$f(\psi)_{i,j} := f(x_j, \phi_i; \psi) : \mathcal{M} \times \Theta \times \Phi \rightarrow \mathbb{R}_+ \quad (3)$$

A unique aspect of searching for efficient pipelines is that we are concerned with the *relative* cost of the pipelines, to find the best one. As such, we propose to utilize the surrogate model as a proxy function for the rank of configurations, and learn the pairwise cost ranking of pairs of pipelines. In this perspective, pairwise ranking strategies use the relative ordering between pairs of configurations to optimize the probability that the rank of the j-th pipeline is lower than the k-th pipeline on the i-th dataset. Therefore, using given pre-computed cost $C_{i,j} = C(x_j, D_i)$ we define the set of triples $\mathcal{E} := \{(i,j,k) \mid C(x_j, D_i) < C(x_k, D_i)\}$. Every triple (i,j,k) denotes a pair (x_j, x_k), where the cost of x_j is smaller (better pipeline) than x_k on the i-th dataset. Correspondingly, we want our surrogate to predict $f(\psi)_{i,j}$ to be lower than $f(\psi)_{i,k}$; we thus meta-learn our surrogate with a ranking loss as:

$$\arg\min_{\psi} \sum_{(i,j,k) \in \mathcal{E}} \log\left(\sigma\left(f(\psi)_{i,j} - f(\psi)_{i,k}\right)\right), \quad (4)$$

with $\sigma(\cdot)$ as the sigmoid function which prevents the difference from exploding to negative infinity as we minimize the loss. Equation 4 maximizes the gap between the surrogate scores, by *decreasing the surrogate score for the good DL pipelines with low costs*, while at the same time increasing the surrogate score of bad pipelines with high costs. As a result, the score of the best DL pipeline with the lowest cost will be maximally decreased. Furthermore, Figure 2 presents a visual description of our proposed ranking loss.

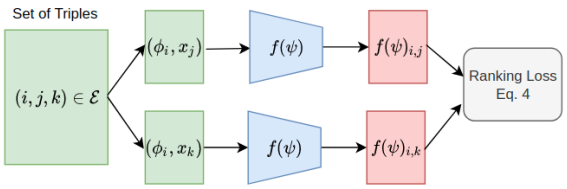

Figure 2. Overview of our pairwise ranking objective

Once we meta-learn the surrogate, we can transfer it to a new dataset $D^{(\text{new})}$ with meta-features $\phi^{(\text{new})}$ in a **zero-shot HPO** mechanism using Equation 5. The full meta-learn and meta-test procedure is depicted in Figure 1.

$$x^{(\text{new})} := \underset{x_n, n \in \{1,\ldots,N\}}{\arg\min} f\left(x_n^{(\text{ft})}, \phi^{(\text{new})}; \psi\right) \tag{5}$$

For an empirical motivation on the benefits of learning surrogates with pairwise ranking losses, we compare to the same surrogate model optimized with a least-squares loss:

$$\underset{\psi}{\arg\min} \sum_{i=1}^{I} \sum_{n=1}^{N} \left(f\left(\psi\right)_{i,n} - C\left(x_n, D_i\right)\right)^2 \tag{6}$$

As a sanity check, we also compare the performance of randomly selecting a pipeline. In this experiment, we evaluate the performance of the pipeline having the largest estimated value by the surrogate (Equation 5) across all the I-many source datasets. The results of Figure 3 demonstrate that the surrogate trained with Equation 4 is significantly better than the regression-based variant of Equation 6 in terms of identifying the best pipeline. Further details about the evaluation protocol are found in Section 5.1.

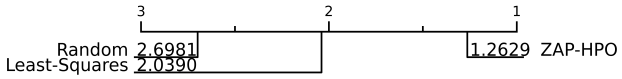

Figure 3. Critical difference diagram comparing loss functions using the Wilcoxon-Holm signed-rank (5% significance level).

4. ZAP Meta-Dataset Design

In this section, we introduce a novel meta-dataset (Pineda-Arango et al., 2021), that will ultimately allow us to perform zero-shot AutoML with pre-trained models (ZAP). The meta-data required for the ZAP problem includes a set of datasets with meta-features, a set of DL pipelines, and the test costs for these pipelines on these datasets. Correspondingly, we describe how we curated a set of 35 image datasets, with 15 augmentations each (4.1); define a space of DL pipelines (4.2); and find strong instantiations in it for each of the datasets, each of which we evaluate on all datasets to obtain a 525×525 matrix of test costs (4.3).

4.1. A Set of Image Datasets for ZAP

The set of datasets should be chosen to be representative of the datasets that will eventually be tackled by the ZAP system building on them. While all our pre-trained networks are pre-trained on ImageNet (Deng et al., 2009), during the fine-tuning stage also smaller and specialized datasets are to be expected. Consequently, we chose both small and large, as well as diverse datasets that cover a wide range of domains (objects, medical, aerial, drawings, etc.) with varying formats, i.e. colored and black-and-white images and datasets with varying image resolutions or the number of classes. With this preference in mind, we retrieved 35 *core* datasets provided by the TensorFlow (Abadi et al., 2015) Datasets (TFDS) utility library (Google, 2021) and applied a dataset augmentation process (Stoll, 2020) that takes a TFDS core dataset as input and outputs a subset of that differs in the number of classes and the number of train/test samples per class. Note that this dataset augmentation process does not perform augmentations on a sample level. We repeat this subset retrieval 15 times for each dataset, resulting in 525 datasets $\mathcal{D}$. Further details about the augmentation process are found in Appendix A.2.

To represent a dataset, we use only extremely cheap and readily available dataset-dependent meta-features (Hutter et al., 2020) ϕ: number of training images, number of image channels, image resolution, and number of classes.

4.2. DL Pipeline Design Space for ZAP on Image Data

The DL pipelines we employ should be chosen to be diverse and achieve high performance on the aforementioned datasets since the optimum we can hope for is to choose the best of these pipelines on a per-dataset basis. To obtain strong pipelines, we started from the code base of the winner of the AutoCV competition (Baek et al., 2020), which fine-tuned a pre-trained ResNet-18 model. We then built a highly-parameterized space of DL pipelines around this by exposing a wide range of degrees of freedom. These included fine-tuning hyperparameters, such as learning rate, percentage of frozen parameters, weight decay, and batch size. Additionally, we exposed hyperparameters for the online execution that were previously hard-coded and that control, e.g., the number of samples used or when to evaluate progress with the validation dataset. To span a more powerful space with diverse pipelines, we also added additional architectural, optimization, as well as fine-tuning choices, including:

- A binary choice between an EfficientNet (Tan & Le, 2019) pre-trained on ImageNet (Russakovsky et al., 2015) or the previously-used ResNet-18;
- The proportion of weights frozen when fine-tuning;
- Additional stochastic optimizers (Adam (Kingma & Ba, 2015), AdamW (Loshchilov & Hutter, 2018), Nesterov accelerated gradient (Nesterov, 1983)) and learning rate schedules (plateau, cosine (Loshchilov & Hutter, 2017));
- A choice of using a simple classifier (either a SVM,

random forest or logistic regression) that can be trained and used within the first 90 seconds of run-time in order to improve anytime performance.

Overall, our DL pipeline space $\mathcal{X}$ is comprised of 26 hyperparameters of the types real and integer-valued, categorical, and conditional. A condensed version is presented in Table 6.

Table 1. **The search space of our DL pipelines** consisting of general DL hyperparameters, training-strategy hyperparameters and fine-tuning strategy hyperparameters. A more detailed version can be found in Appendix A.1.

Name	Type, Scale	Range
Batch size	int, log	$[16, 64]$
Learning rate	float, log	$[10^{-5}, 10^{-1}]$
Weight decay	float, log	$[10^{-5}, 10^{-2}]$
Momentum	float	$[0.01, 0.99]$
Optimizer	cat	{SGD, Adam, AdamW}
Scheduler	cat	{plateau, cosine}
Architecture	cat	{ResNet18, EffNet-b0 EffNet-b1, EffNet-b2}
Steps per epoch	int, log	$[5, 250]$
Early epoch	int	$[1, 3]$
CV ratio	float	$[0.05, 0.2]$
Max valid count	int, log	$[128, 512]$
Skip valid thresh.	float	$[0.7, 0.95]$
Test freq.	int	$[1, 3]$
Max inner loop	float	$[0.1, 0.3]$
# init samples	int,log	$[128, 512]$
Max input size	int	$[5, 7]$
1st simple model	cat	{true, false}
Simple model	cat	{SVC, NuSVC, RF, LR}

4.3. Selection and Evaluation of DL Pipelines

With the 525 datasets and our 26-dimensional DL pipeline space at our disposal, we now explain how we generated the DL pipeline candidates that we evaluated on the datasets. Instead of uniformly or randomly sampling the 26-dimensional DL pipeline space, to focus on DL pipelines that are strong at least on one dataset, we ran an optimization process to find a (near-)optimal DL pipeline for one dataset at a time. Specifically, we used the hyperparameter optimization method BOHB (Falkner et al., 2018), which supports high-dimensional and categorical hyperparameter spaces, to find a (near)-optimal instantiation of our DL pipeline space for each dataset. We optimized the anytime Area under the Learning Curve (ALC) score (introduced in the AutoDL challenge (Liu et al., 2021) and described in more detail in Section 5.1) via BOHB, with a budget of five minutes for evaluating one DL pipeline on one dataset. We repeated each of these runs three times and used the mean to handle the substantial noise in these evaluations. This process resulted in one optimized DL pipeline per dataset; we thus have $N = D = 525$ DL pipelines that comprise the set $\mathcal{X}$ of DL pipelines in our ZAP formulation.

Given this set of 525 DL pipelines $\mathcal{X}$, and the set of our 525 datasets $\mathcal{D}$, let us now explain the evaluation procedure. We ran each pipeline $x \in \mathcal{X}$ on each dataset $D \in \mathcal{D}$, computing the ALC score the pipelines reached within 10 minutes, and again computing the mean of three runs to reduce noise. While the AutoDL competition used a budget of 20 minutes, we used a shorter time of 10 minutes here (and 5 minutes for the runs of BOHB above) for two reasons: First, to limit the substantial computational overhead for carrying out these $525 \cdot 525 = 275,625$ evaluations of (DL pipeline, dataset) pairs; still, it required 2,871 GPU days to collect this data. Second, due to the typically monotonically increasing anytime ALC score, performance after 5 and 10 minutes can be expected to be a good proxy for the full 20 minutes.

Finally, we record every pairs' average-of-three ALC score in the cost matrix $C \in \mathbb{R}^{NxI}$ (in our case with $N = I = 525$ since we found one DL pipeline per dataset). This cost matrix is visualized in Figure 4. From the cost matrix, we directly see that there are easy datasets (at the top, where all pipelines achieve high scores) and hard ones at the bottom (where only very few pipelines reach high scores). Likewise, there are overall strong pipelines (to the left, with good scores on most datasets) and poor ones (on the right, with good scores on only a few datasets). The most interesting pattern for ZAP is that there exists substantial horizontal and vertical striping, indicating that different datasets are hard for different pipelines. This points to the usefulness of selecting pipelines in a dataset-dependent manner in ZAP.

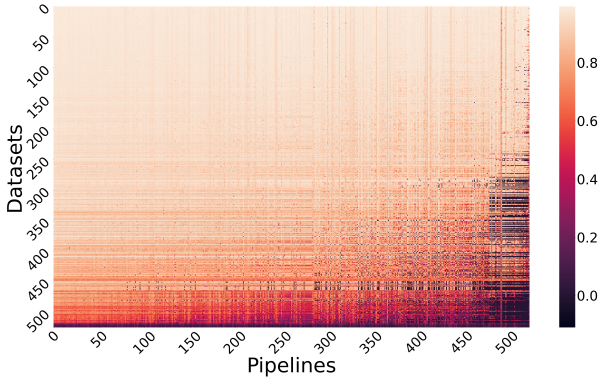

Figure 4. **Cost matrix C as a heatmap** Color indicates the ALC score (higher is better). We observe that some datasets (dark rows) are more complex and some pipelines (dark columns) generalize worse across datasets systematically than others.

5. Experiments

Our experiments are designed to evaluate the performance of an AutoDL system based on ZAP. We use the anytime

evaluation protocol of the ChaLearn AutoDL Challenge (Liu et al., 2021) and demonstrate that our ZAP methods perform substantially better than the winners of that competition, both in our ZAP evaluation benchmark grounded on our ZAP meta-dataset as well as in the original AutoDL challenge benchmark. We also carry out a range of ablations to better understand the root of our gains.

5.1. Evaluation Protocol

Let us first describe the restrictions of the evaluation protocol under which we evaluate different AutoDL methods (our ZAP approach and various baselines). The main restriction is the anytime learning metric to score participants with the Area under the Learning Curve (ALC): at each time step, an AutoDL system can update its predictions on test data, and in a post hoc evaluation phase, the accuracy of these predictions is averaged over the entire learning curve. A second core aspect of the challenge is the limited time budget of 20 minutes for training models and making predictions on test data. In light of the large training times of conventional deep learning models, this short time window encourages the use of efficient techniques, particularly the use of pretrained models. The performance measurement starts with the presentation of the training data (and the inputs of the test split), and the AutoDL system can train in increments and interleave test predictions; however, the time for making predictions also counts as part of the run-time budget. Consequently, AutoDL systems need to decide when to make predictions to improve performance.

For a formal description of the metric, as well as an example of a learning curve plot under the competition metric, please see Appendix A.3.

5.2. Benchmarks and Training Protocol

Overall, we evaluate our ZAP methods under two benchmarks: one based on the ZAP meta-dataset which we refer to as *ZAP benchmark* and the original AutoDL benchmark. For the AutoDL benchmark, we submit our ZAP methods trained on the ZAP meta-dataset. The following describes how we train and evaluate our methods on the ZAP benchmark. We evaluate in a "leave-one-core-dataset-out protocol" that avoids any possibility of an information leak across augmented datasets. Specifically, we meta-train our methods on 34 out of the original 35 datasets, plus their augmented versions, and test on the held-out original dataset. We average the resulting performance over 35 outer loop iterations holding out a different core dataset each time. We further apply 5-fold inner cross-validation to optimally identify the best stopping epoch while monitoring validation performance. We evaluate each method (including the baselines) 10 times with different seeds and report averages, standard deviations, boxplots, and statistical tests over these 10 results.

For evaluation under the AutoDL benchmark, we upload our ZAP methods trained on the ZAP meta-dataset as well as the baselines to the submission board, made available to us by the challenge organizers. We report the performances on the five undisclosed final datasets of different domains (objects, aerial, people, medical, handwriting recognition) across 10 submissions. Here, we used the same hyperparameters from the ZAP benchmark.

5.3. Baselines

To assess the performance of our proposed method, we compare it against multiple baselines, which we describe here. Aside from a random selection of one of our 525 carefully designed DL pipelines, and the single best pipeline on average across the datasets, we chose the top-3 winners of the 2019 ChaLearn AutoDL Challenge: DeepWisdom, DeepBlueAI, and PASA NJU.

Given a novel dataset and our 525 selected DL pipelines, **random selection** uniformly samples one of these pipelines and **single-best** picks the one which performs best on average over $\mathcal{D}$. We average the random selection baseline over three random selections.

Table 2. **Summary of winner techniques** All contenders use ImageNet pre-trained networks and FAA denotes Fast AutoAugment.

Solution	Augmentation	ML technique
DeepWisdom	FAA	ResNet18/9 Meta-trained solution agents
DeepBlueAI	FAA	ResNet18 Adaptive ensemble learning
PASA NJU	Simple	ResNet18/SeResnext50 Data adaptive preprocessing

The challenge winner baselines build their methods on top of AutoCLINT (Baek et al., 2020), with the following modifications (summarized in Table 2):

- **DeepWisdom** initially caches mini-batches with a pre-trained ResNet-18 model and quickly switches to a pre-trained ResNet-9 by inputting cached batches first. They initialize the networks with ImageNet pre-trained parameters except for the batch normalization and bias layers. After an initial optimization phase, they turn on Fast AutoAugment (Lim et al.).
- **DeepBlueAI** initializes a pre-trained ResNet-18 network and adapts some of the model hyperparameters (image size, steps per epoch, epoch after which starting validating and fusing results, etc.) to the target dataset. They also ensemble the latest n predictions to stabilize them. Later in the procedure, they augment the dataset by Fast AutoAugment for a limited budget.
- **PASA NJU** pre-processes the data with a data-adaptive strategy by first sampling images to analyze. Then

they crop images to a standard shape and apply image flip augmentations. They start the training with a pre-trained ResNet-18 and switch to SeResNext-50 (Hu et al., 2017) when no further improvement is expected on the validation score.

5.4. Results on the ZAP Benchmark

In Figure 5, we depict the performance of our ZAP methods ZAP-AS and ZAP-HPO compared to the winner baselines of the AutoDL challenge. Our algorithm-selection-based ZAP-AS method already outperforms the competition winners, and our geometry-aware ranking-based model, ZAP-HPO, performs even significantly better.

In Table 3, we report the rank of the scores identifying the winners across the two main metrics. Our proposed method wins in both ALC score, i.e., the metric for which it was optimized, but also in terms of the normalized area under the curve (Normalized AUC).

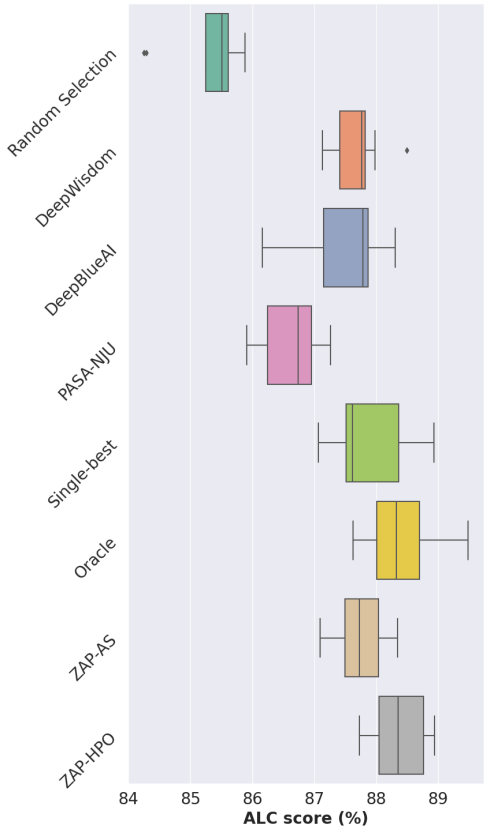

Figure 5. **ALC scores of our approach vs. winner baselines** over 525 datasets and 10 repetitions. Our ZAP methods clearly improve over the challenge winners (higher is better), by almost 1 point. Our geometry-aware zero-shot HPO version of ZAP with its binary pairwise ranking objective works best.

Table 3. **Ranking our approach vs. winner baselines on the ZAP benchmark.** We rank the solutions per test dataset and report average ranks over the 525 datasets (averages over 10 repetitions). ZAP clearly performs best (lower is better), both in terms of ALC (which it was optimized for) and also in terms of Normalized AUC.

Solution	Rank (ALC)	Rank(Normalized AUC)
DeepWisdom	2.53 ± 0.06	2.63 ± 0.03
DeepBlueAI	2.64 ± 0.04	2.73 ± 0.03
PASA NJU	2.76 ± 0.03	2.56 ± 0.03
ZAP-HPO	$\mathbf{2.07 \pm 0.05}$	$\mathbf{2.08 \pm 0.03}$

5.5. Results for a Sparsely-filled Cost Matrix

To further investigate the source of the gains in our model, we propose a more realistic setting, where the cost matrix includes missing values. While algorithm selection methods, such as ZAP-AS, require the dense cost matrix, our geometry-aware rank-based ZAP-HPO method handled missing values gracefully. To evaluate this quantitatively, next to ZAP-AS and ZAP-HPO with the full cost matrix, we also evaluate ZAP-HPO based on cost matrices that only have 75%, 50%, and 25% of the entries (dropped at random) remaining. As Figure 6 shows, ZAP-HPO's performance loss due to missing entries is quite small, and even with only 25% remaining entries, it still performs similarly to ZAP-AS. We believe that ZAP-HPO's low sensitivity to missing values stems from the capacity of the model to capture the correlation across the pipelines in the geometric space. It can hereby generalize well and "impute" missing values of the cost matrix. Furthermore, as shown in Table 4, even with missing values in the cost matrix, ZAP-HPO clearly outperforms the winners of the AutoDL competition.

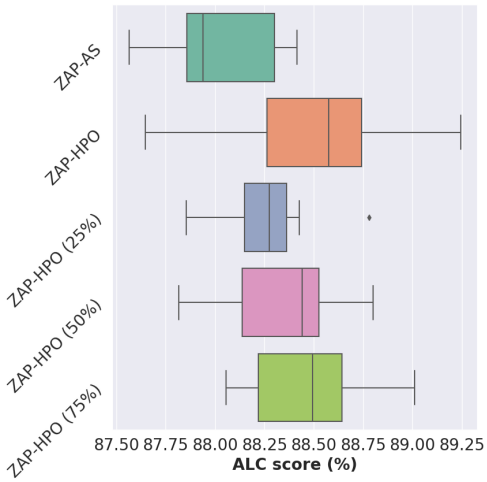

Figure 6. **ALC scores of ZAP-HPO when meta-trained on a full (dense) vs. sparse cost matrix** over 525 datasets and 10 repetitions. The density of the matrices are denoted as 75%, 50%, and 25%.

Table 4. **The ranking of ZAP-HPO when optimized with a sparse cost matrix** shows we still outperform the AutoDL challenge winner solutions.

Solution	75% filled	50% filled	25% filled
DeepWisdom	2.52 ± 0.06	2.52 ± 0.05	2.52 ± 0.05
DeepBlueAI	2.64 ± 0.05	2.64 ± 0.04	2.63 ± 0.04
PASA NJU	2.75 ± 0.03	2.75 ± 0.04	2.75 ± 0.04
ZAP-HPO (sparse)	$\mathbf{2.09 \pm 0.06}$	$\mathbf{2.09 \pm 0.05}$	$\mathbf{2.10 \pm 0.04}$

5.6. Results on the AutoDL Benchmark

In Table 5 we report the average rank performances for the AutoDL benchmark on the five undisclosed datasets across 10 submissions. We highlight that, unlike the winner baselines, we did not use the challenge feedback datasets to optimize the base model and zero-shot model hyperparameters for the final submission but reused (only) the ones from the ZAP benchmark. Due to a difference in distributions between these benchmarks, it cannot be taken for granted that our method generalizes to the AutoDL competition datasets. However, as Table 5 shows, ZAP-HPO clearly outperforms the winner baselines on this AutoDL setting. It also clearly outperforms the single-best and random baselines (which have ranks $2.7(\pm 0.1)$ and $3.42(\pm 0.68)$, respectively). In this setting of generalizing out of distribution, the more conservative ZAP-AS method performs even slightly better than ZAP-HPO, with average ranks of $1.81(\pm 0.3)$ vs. $2.16(\pm 0.15)$.

Table 5. **Ranking our approach vs. winner baselines on the AutoDL benchmark.** We rank the solutions per test dataset and report average ranks over the five AutoDL benchmark final datasets (averages over 10 submissions).

Solution	Rank (ALC)
DeepWisdom	2.46 ± 0.13
DeepBlueAI	2.76 ± 0.08
PASA NJU	2.62 ± 0.11
ZAP-HPO	$\mathbf{2.16 \pm 0.15}$

6. Conclusion

In this paper we extend the realm of AutoML to address the common problem of fine-tuning pre-trained Deep Learning (DL) models on new image classification datasets. Concretely, we focus on deciding which pre-trained model to use and how to set the many hyperparameters of the fine-tuning procedure, in a regime where strong anytime performance is essential. We formalize the problem as Zero-shot AutoML with Pre-trained Models (ZAP), which transfers knowledge from a meta-dataset of a number of DL pipelines evaluated on a set of image datasets. In that context, we open-source the largest meta-dataset of evaluations for fine-tuning DL pipelines with 275K evaluated pipelines on 35 popular image datasets (2871 GPU days of compute). Furthermore, we propose two approaches for tackling ZAP: (i) formulating it as an instance of the algorithm selection (AS) problem and using AS methods, and (ii) a novel zero-shot hyperparameter optimization method trained with a ranking objective. Our methods clearly achieve the new state of the art in terms of anytime Automated Deep Learning (AutoDL) performance and significantly outperform all the solutions of the 2019 ChaLearn AutoDL Challenge.

7. Limitations

As mentioned before, computing the cost matrix is expensive. In particular, when training deep models, early optimization phases are often noisy and the final performance of a model is difficult to predict. Consequently, the accuracy of the cost matrix is directly coupled to the duration of the deep model training. Applying early-stopping methods to reduce the expenses of determining a cost matrix is thus challenging. Another limitation we observed is the sensitivity to the zero shot model's hyperparameters. For example, we noticed that using a least-squares or a triplet margin objective performed significantly worse than our ZAP-HPO objective. Lastly, it is not clear a-priori which attributes best describe the DL pipelines in order to achieve the best performance and selecting a sub-optimal set may lead to a deterioration of performance.

Acknowledgements

This paper builds on and extends our original submission to the AutoDL challenge, and we are indebted to everyone who helped with that submission; in particular, we would like to thank Danny Stoll for his help on the initial hyperparameter configuration space design and data augmentation as well as Marius Lindauer for his help with AutoFolio. We also thank Dipti Sengupta for her help in reviewing both code and the paper. Moreover, we acknowledge funding by Robert Bosch GmbH, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number 417962828, by the state of Baden-Württemberg through bwHPC, and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG, by TAILOR, a project funded by the EU Horizon 2020 research, and innovation programme under GA No 952215, and by European Research Council (ERC) Consolidator Grant "Deep Learning 2.0" (grant no. 101045765). Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the ERC can be held responsible for them.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/.

Awad, N., Mallik, N., and Hutter, F. DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. In *Proc. of IJCAI'21*, pp. 2147–2153, 2021.

Baek, W., Kim, I., Kim, S., and Lim, S. AutoCLINT: The Winning Method in AutoCV Challenge 2019. volume abs/2005.04373, 2020.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. 13:281–305, 2012.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Frechétte, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., and Vanschoren, J. ASlib: A benchmark library for algorithm selection. 237:41–58, 2016.

Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature, 2018.

Cohen, G., Afshar, S., Tapson, J., and Schaik, A. V. Emnist: Extending mnist to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. doi: 10.1109/ijcnn.2017.7966217.

Das, N., Reddy, J. M., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., and Basu, D. K. A statistical-topological feature combination for recognition of handwritten numerals. *Appl. Soft Comput.*, 12(8):2486–2495, August 2012a. ISSN 1568-4946. URL http://dx.doi.org/10.1016/j.asoc.2012.03.039.

Das, N., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., and Basu, D. K. A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application. *Appl. Soft Comput.*, 12(5):1592–1606, May 2012b. URL http://dx.doi.org/10.1016/j.asoc.2011.11.030.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of CVPR'09*, pp. 248–255, 2009.

Dumoulin, V., Houlsby, N., Evci, U., Zhai, X., Goroshin, R., Gelly, S., and Larochelle, H. Comparing transfer and meta learning approaches on a unified few-shot classification benchmark. *CoRR*, abs/2104.02638, 2021. URL https://arxiv.org/abs/2104.02638.

Elson, J., Douceur, J. J., Howell, J., and Saul, J. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., October 2007.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv:2003.06505 [stat.ML]*, 2020.

Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proc. of ICML'18*, pp. 1437–1446, 2018.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and robust automated machine learning. In *Proc. of NeurIPS'15*, pp. 2962–2970, 2015a.

Feurer, M., Springenberg, J., and Hutter, F. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proc. of AAAI'15*, pp. 1128–1135, 2015b.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017.

Google. TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv:1512.03385 [cs.CV]*, 2015.

Helber, P., Bischke, B., Dengel, A., and Borth, D. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, 2017.

Howard, J. Imagenette. URL https://github.com/fastai/imagenette/.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

Hutter, F., Hoos, H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION'11*, pp. 507–523, 2011.

Hutter, F., Kotthoff, L., and Vanschoren, J. (eds.). *Meta-Learning*, pp. 35–61. Springer International Publishing, 2020.

Jomaa, H. S., Arango, S. P., Schmidt-Thieme, L., and Grabocka, J. Transfer learning for bayesian hpo with end-to-end landmark meta-features. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021a.

Jomaa, H. S., Schmidt-Thieme, L., and Grabocka, J. Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, pp. 964–985, 2021b.

Kadioglu, S., Malitsky, Y., Sellmann, M., and Tierney, K. ISAC - instance-specific algorithm configuration. In *Proc. of ECAI'10*, pp. 751–756, 2010.

Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. Algorithm selection and scheduling. In *Proc. of CP'11*, pp. 454–469, 2011.

Kather, J. N., Weis, C.-A., Bianconi, F., Melchers, S. M., Schad, L. R., Gaiser, T., Marx, A., and Z"ollner, F. G. Multi-class texture analysis in colorectal cancer histology. *Scientific reports*, 6:27988, 2016.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, CVPR'11*, 2011.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *Proc. of ICLR'15*, 2015.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. In *Proc. of ECCV'20*, pp. 491–507, 2020.

Kotthoff, L., Gent, I., and Miguel, I. An evaluation of machine learning in algorithm selection for search problems. 25(3):257–270, 2012.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet classification with deep convolutional neural networks. In *Proc. of NeurIPS'12*, pp. 1097–1105, 2012.

Laadan, D., Vainshtein, R., Curiel, Y., Katz, G., and Rokach, L. Rankml: A meta learning-based approach for pre-ranking machine learning pipelines. *arXiv preprint arXiv:1911.00108*, 2019.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. Rethinking the hyperparameters for fine-tuning. In *Proc. of ICLR'20*, 2020.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *Proc. of ICLR'17*, 2017.

Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. Fast AutoAugment.

Lindauer, M., Hoos, H., Hutter, F., and Schaub, T. Autofolio: Algorithm configuration for algorithm selection. In *Proc. of Workshops at AAAI'15*, 2015.

Liu, Z., Pavao, A., Xu, Z., Escalera, S., Ferreira, F., Guyon, I., Hong, S., Hutter, F., Ji, R., Junior, J. C. S. J., Li, G., Lindauer, M., et al. Winning solutions and post-challenge analyses of the chalearn autodl challenge 2019. In *TPAMI'21*, pp. 3108–3125, 2021.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *Proc. of ICLR'17*, 2017.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam. In *Proc. of ICLR'18*, 2018.

Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *Proc. of IJCAI'13*, pp. 608–614, 2013.

Moroney, L. Horses or humans dataset, feb 2019a. URL http://laurencemoroney.com/horses-or-humans-dataset.

Moroney, L. Rock, paper, scissors dataset, feb 2019b. URL http://laurencemoroney.com/rock-paper-scissors-dataset.

Mwebaze, E., Gebru, T., Frome, A., Nsumba, S., and Tusubira, J. icassava 2019fine-grained visual categorization challenge, 2019.

Nene, S. A., Nayar, S. K., Murase, H., et al. Columbia object image library (coil-20). 1996.

Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Paszke, A., Gross, S., Massa, F., Lerer, A., et al. PyTorch: An imperative style, high-performance deep learning library. In *Proc. of NeurIPS'19*, pp. 8024–8035, 2019.

Perrone, V., Jenatton, R., Seeger, M., and Archambeau, C. Scalable hyperparameter transfer learning. In *Proc. of NeurIPS'18*, pp. 6845–6855, 2018.

Pineda-Arango, S., Jomaa, H. S., Wistuba, M., and Grabocka, J. HPO-B: A large-scale reproducible benchmark for black-box HPO based on openml. volume abs/2106.06257, 2021.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *Proc. of ICML'21*, pp. 8748–8763, 2021.

Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Jaeger, S., and Thoma, G. R. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568, 2018.

Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Rice, J. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Salinas, D., Shen, H., and Perrone, V. A quantile-based approach for hyperparameter transfer learning. In *Proc. of ICML'20*, pp. 8438–8448, 2020.

Smith-Miles, K. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):6:1–6:25, January 2009.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, and Adams, R. Scalable Bayesian optimization using deep neural networks. In *Proc. of ICML'15*, pp. 2171–2180, 2015.

Springenberg, J., Klein, A., Falkner, S., and Hutter, F. Bayesian optimization with robust Bayesian neural networks. In Lee, D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Proc. of NeurIPS'16*, 2016.

Stoll, D. Icgen, 2020. URL https://github.com/automl/ICGen.

Sun, Q., Liu, Y., Chua, T.-S., and Schiele, B. Meta-transfer learning for few-shot learning. In *Proc. of CVPR'19*, pp. 403–412, 2019.

Tan, M. and Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proc. of ICML'19*, pp. 6105–6114, 2019.

Team, T. T. Flowers, jan 2019. URL http://download.tensorflow.org/example_images/flower_photos.tgz.

Tornede, A., Wever, M., and Hüllermeier, E. Extreme algorithm selection with dyadic feature representation. In *Proc. of DS'20*, pp. 309–324, 2020.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Xu, K., Goroshin, R., Swersky, C. G. K., Manzagol, P., and Larochelle, H. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019.

Vanschoren, J. Meta-learning: A survey. *CoRR*, abs/1810.03548, 2018.

Verma, V. K., Arora, G., Mishra, A., and Rai, P. Generalized zero-shot learning via synthesized examples. In *Proc. of CVPR'18*, pp. 4281–4289, 2018.

Verma, V. K., Brahma, D., and Rai, P. A meta-learning framework for generalized zero-shot learning. In *Proc. of AAAI'19*, pp. 6062–6069, 2019.

Wilson, A. G. and Ghahramani, Z. Copula Processes. In *Proc. of NeurIPS'10*, pp. 2460–2468, 2010.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Proc. of AISTATS'16*, pp. 370–378, 2016.

Wilson, J., Hutter, F., and Deisenroth, M. Maximizing acquisition functions for Bayesian optimization. In *Proc. of NeurIPS'18*, pp. 741–749, 2018.

Winkelmolen, F., Ivkin, N., Bozkurt, H. F., and Karnin, Z. S. Practical and sample efficient zero-shot HPO. volume abs/2007.13382, 2020.

Wistuba, M. and Grabocka, J. Few-shot bayesian optimization with deep kernel surrogates. In *Proc. of ICLR'21*, 2021.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. Sequential Model-free Hyperparameter Tuning. In *Proc. of ICDM '15*, pp. 1033–1038, 2015.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Proc. of ECML/PKDD'16*, pp. 199–214, 2016.

Xian, Y., Lorenz, T., Schiele, B., and Akata, Z. Feature generating networks for zero-shot learning. In *Proc. of CVPR'18*, pp. 5542–5551, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017.

Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K. SATzilla: Portfolio-based algorithm selection for SAT. 32:565–606, 2008.

Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K. Evaluating component solver contributions to portfolio-based algorithm selectors. In *Proc. of SAT'12*, pp. 228–241, 2012.

Yang, Y. and Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, 2010.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., et al. The Visual Task Adaptation Benchmark. *CoRR*, abs/1910.04867, 2019.

Zhu, J., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

A. Appendix

A.1. Search Space of DL Pipelines

We list the search space of the DL pipelines clustered into two groups. The first group are general DL hyperparameters including the architecture and fine-tuning strategy. The second group defines the early stopping, validation and test strategies during the 20 minutes of training, including set sizes and evaluation timings.

Table 6. **The search space of our DL pipelines** clustered into two groups. The first group are general DL hyperparameters including the architecture and fine-tuning strategy. The second group defines the early stopping, validation and test strategies during the 20 minutes of training, including set sizes and evaluation timings.

Name	Type, Scale	Range
Batch size	int, log	$[16, 64]$
Learning rate	float, log	$[10^{-5}, 10^{-1}]$
Min learn. rate	float, log	$[10^{-8}, 10^{-5}]$
Weight decay	float, log	$[10^{-5}, 10^{-2}]$
Momentum	float	$[0.01, 0.99]$
Optimizer	cat	{SGD, Adam, AdamW}
Nesterov	cat	{true, false}
Amsgrad	cat	{true, false}
Scheduler	cat	{plateau, cosine}
Freeze portion	cat	$\{0.0, 0.1, \ldots, 0.5\}$
Warm-up mult.	cat	$\{1.0, 1.5, \ldots, 3.0\}$
Warm-up epoch	int	$[3, 6]$
Architecture	cat	{ResNet18, EffNet-b0 EffNet-b1, EffNet-b2}
Steps per epoch	int, log	$[5, 250]$
Early epoch	int	$[1, 3]$
CV ratio	float	$[0.05, 0.2]$
Max valid count	int, log	$[128, 512]$
Skip valid thresh.	float	$[0.7, 0.95]$
Test freq.	int	$[1, 3]$
Test freq. max	int	$[60, 120]$
Test freq. step	int	$[2, 10]$
Max inner loop	float	$[0.1, 0.3]$
# init samples	int,log	$[128, 512]$
Max input size	int	$[5, 7]$
1^{st} simple model	cat	{true, false}
Simple model	cat	{SVC, NuSVC, RF, LR}

A.2. Dataset Augmentation

In the following Table 7 we list all TFDS datasets we used for creating our 525 datasets with their respective domains. For each dataset $\mathcal{D}_{original}$ we create 15 subsets by sampling number of classes from range $[2, 100]$ and min/max number of samples per class from range $[20, 10^5]$ such that $\mathcal{D}_i \subseteq \mathcal{D}_{original}, \forall i \in \{1, \ldots, 15\}$ (Figure 7). Remark that there is no procedure on sample level, meaning that subsets inherit image resolutions and channels from the original dataset as given in the Tables 8 and 9. Figure 8 contains the number of sample and class distributions of these subsets along with the AutoDL Challenge benchmark datasets.

A.3. AutoDL: Area Under the Learning Curve (ALC) Metric

In the 2019 ChaLearn AutoDL challenge and also in all our experiments, the main performance metric is the Area under the Learning Curve (ALC). Formally, we test the currently trained model on a test set p_i at time t_i by calculating a scalar score,

Table 7. **Domains of the original datasets**

Domain	Datasets
Objects	Cifar100 (Krizhevsky et al., 2009), Cifar10, Horses or Humans (Moroney, 2019a), CycleGAN Horse2zebra (Zhu et al., 2017), CycleGAN Facades, CycleGAN Apple2orange, Imagenette (Howard), Coil100 (Nene et al., 1996), Stanford Dogs (Khosla et al., 2011), Rock, Paper and Scissors (Moroney, 2019b), TF Flowers (Team, 2019), Cassava (Mwebaze et al., 2019), Fashion MNIST (Xiao et al., 2017), Cars196 (Krause et al., 2013), Cats vs Dogs (Elson et al., 2007), ImageNet Resized 32x32 (Chrabaszcz et al., 2017)
Characters	Cmaterdb Devanagari (Das et al., 2012a;b), Cmaterdb Bangla, MNIST(LeCun et al., 2010), KMNIST (Clanuwat et al., 2018), EMNIST Byclass (Cohen et al., 2017), EMNIST MNIST, Cmaterdb Telugu, EMNIST Balanced, Omniglot (Lake et al., 2015), SVHN Cropped (Netzer et al., 2011)
Medical	Colorectal Histology (Kather et al., 2016), Malaria (Rajaraman et al., 2018)
Aerial	Uc Merced (Yang & Newsam, 2010), CycleGAN Maps, Eurosat RGB (Helber et al., 2017)
Drawings/Pictures	CycleGAN Vangogh2photo, CycleGAN Ukiyoe2photo

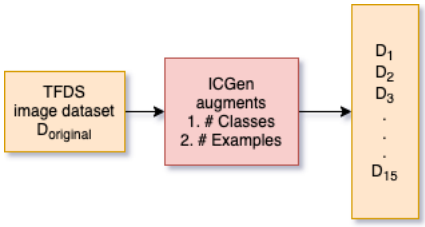

Figure 7. **Dataset augmentation flow**

the Normalized Area Under ROC Curve (AUC):

$$s_i = 2 * AUC(\vec{p_i}) - 1 \tag{7}$$

We then convert this score to a time-sensitive step function

$$s(t) = step\_fn(\vec{s}) \tag{8}$$

and we also transform the time non-linearly between $[0, 1]$ such that the performance on the first minute is weighted roughly the same as the last 10 minutes of the budget:

$$\tilde{t}(t) = \frac{\log(1 + t/t_0)}{\log(1 + T/t_0)} \tag{9}$$

where $T = 1200$ is the total default training budget and $t_0 = 60$ is the default reference time.

Finally, we measure the ALC by:

$$ALC = \int_0^1 s(t) d\tilde{t}(t) \tag{10}$$

For more details on the metric, we refer the reader to (Liu et al., 2021). We depict an example of an ALC plot in Figure 9.

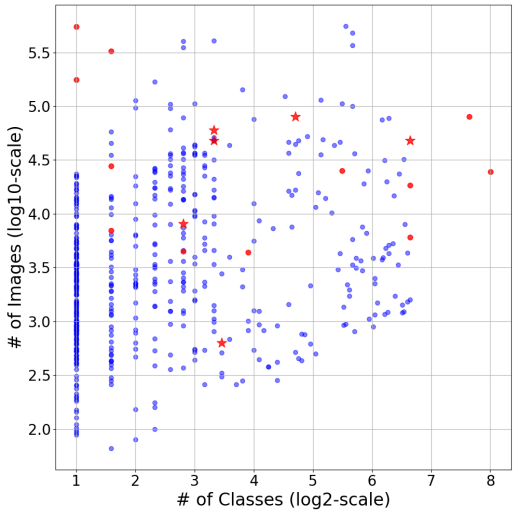

Figure 8. **Distribution of the meta-features** where each point corresponds to a dataset. Blue points come from our meta-dataset, whilst red ones are the datasets provided by AutoDL challenge. Star and point markers are public AutoDL datasets and private AutoDL datasets (from feedback and final phases), respectively.

A.4. Hardware Setup

Due to the anytime performance measurements of the AutoDL challenge's training and evaluation protocol and the resulting importance on wall clock time, we ensured that all experiments in this work were run on machines with the same hardware setup. The specification of our machines is the following: AMD EPYC 7502 32-Core Processor, NVIDIA GeForce RTX 2080 Ti, 500GB RAM, CUDA version 11.5, Ubuntu 20.04.3 LTS.

Statement of Contributions

Ekrem Öztürk: Methodology, Software (extending the AutoDL 2019 challenge submission base code from Fabio with more datasets), Validation (running all ZAP-AS experiments after the challenge deadline starting June 2020 and also the ZAP-HPO rebuttal experiments), Investigation, Data Curation, Writing: Original Draft, Writing: Review and Editing, Visualization

Fabio Ferreira: Conceptualization, Methodology, Software (providing the base code for the AutoDL challenge 2019 submission), Validation (and running all ZAP-AS experiments up to the challenge deadline on March 14 2020), Investigation,

Table 8. **Resolution Distribution** of datasets

Resolution	# of Datasets
28×28	90
32×32	105
64×64	15
105×105	15
128×128	15
150×150	15
256×256	90
300×300	30
600×600	15
Varying	135

Table 9. **Image channels** distribution of datasets

Channel	# of Datasets
B&W	90
RGB	435

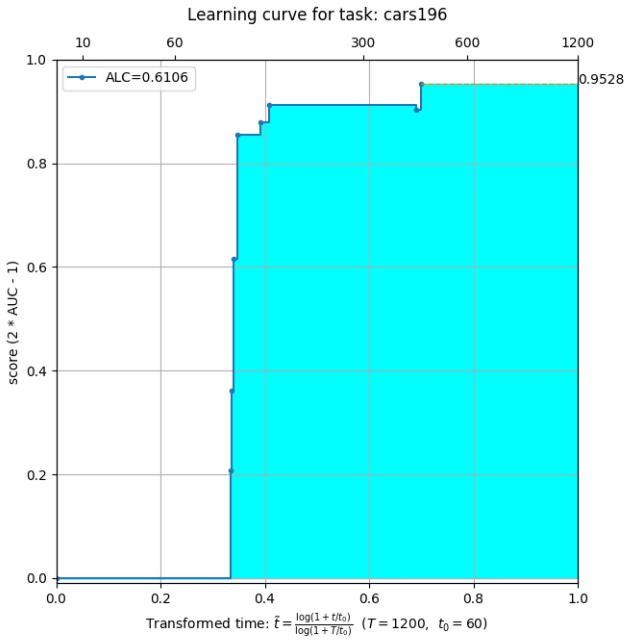

Figure 9. **Learning curve of a task** is the step function of normalized AUC (NAUC) scores received during the 20 minutes and the light blue area underneath is the area under the learning curve(ALC). Every dark blue point (steps) corresponds to a set of predictions made and y-axis to its NAUC score. When the procedure stops early, the final NAUC is interpolated to the end with a horizontal line. Top x-axis is the time limited to 1200 seconds and the bottom axis is the transformed version between $[0, 1]$. Visualization clearly shows that the first 60 seconds contributes to more than 20% of the total score and has the same weight as the last 600 seconds. Remark that total budget $T = 1200$ and reference time constant $t_0 = 60$ here, the same values as in our experiments.

Data Curation, Writing: Original Draft, Writing: Review and Editing, Visualization, Supervision of Ekrem Öztürk

Hadi S. Jomaa: (**todo**) Methodology, Software (providing the ZAP-HPO code), Validation (running the ZAP-HPO experiments for the initial paper submission), Writing: Original Draft, Visualization

Lars Schmidt-Thieme: Writing: Review and Editing

Josif Grabocka: Conceptualization, Methodology, Writing: Original Draft, Writing: Review and Editing, Supervision of Hadi S. Jomaa, Project management

Frank Hutter: Conceptualization (proposing the original ideas for tackling the problem specification of the AutoDL challenge), Methodology, Resources, Writing: Original Draft, Writing: Review and Editing, Supervision of Fabio Ferreira, Project management, Funding acquisition

Signatures

Ekrem Öztürk	Date
Fabio Ferreira	Date
Hadi S. Jomaa	Date
Lars Schmidt-Thieme	Date
Josif Grabocka	Date
Frank Hutter	Date