DISTILLATION-AWARE NAS FOR UNSEEN DATASETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Distillation-aware network architecture search (DNAS) aims to search for an optimal student architecture to obtain the best performance and efficiency when distilling the knowledge from the given teacher model and dataset. However, NAS for knowledge distillation (KD) needs to consider the final accuracy of KD from teacher network which is difficult problem than non-KD NAS. Previous DNAS methods still have mostly tackled the optimization of searching for the network architecture for a single task, which does not generalize well with the unseen teacher or unseen dataset which is a more realistic scenario for DNAS. To overcome the computation costs and obtain transferable search for unseen dataset, we introduce a gradient-based rapid one-shot adaptive predictor by using the accuracy of the teacher model that allows removing the need to train architectures on a target task to obtain sample accuracies for adaptation. The experimental results demonstrate that our proposed meta-predictor that conducts one-shot adaptation with the accuracy of the teacher model successfully generalizes to multiple unseen datasets for DNAS tasks. Additionally, our model outperforms existing meta-NAS on DNAS tasks and shows the effectiveness of one-shot adaptation.

1 INTRODUCTION

Architecture-aware Knowledge Distillation (AKD) or Neural Architecture Search for Knowledge Distillation (NAS for KD) aim to search an optimal student architecture to obtain the best performance and efficiency when distilling the knowledge from the given teacher model and dataset (Liu et al., 2020). In general, we need carefully designed AKD frameworks instead of simply using NAS methods to find student architecture for a KD environment. The reasons are as follows: 1) An optimal student architecture for distilling knowledge from the teacher model and an optimal architecture for learning from scratch with random initialization are different (Liu et al., 2020). 2) Even for the same dataset if the teacher model is different, each teacher model has a different optimal student architecture (Liu et al., 2020). To meet the conditions, most AKD methods (Liu et al., 2020) have been guided through a search process using KD loss. Liu et al. (2020) guided the reinforcement-learning agent to search for an optimal student architecture to maximize the KD-based rewards. However, such existing AKD methods are too time-consuming that each search requires 5 days with 200 TPUv2 (Liu et al., 2020) and xxx, which is a critical limitation to be applied to many real-world scenarios. Since optimal student architectures are different depending on the type of datasets, teacher models, and resource budgets, accordingly, to support these many practical scenarios, we need a rapid and lightweight AKD method.

On the other hand, recently, in the NAS world for non-KD environments, rapid NAS methods have been getting a lot of attention from researchers: 1) meta-learning-based transferable NAS methods (meta-NAS framework) (Lee et al., 2021a;b; Jeong et al., 2021) 2) zero-cost proxies (Mellor et al., 2021; Abdelfattah et al., 2021). The former, meta-NAS frameworks learn the generalized search process over multiple tasks rather than a single task. This allows the frameworks to be rapidly applied to a new unseen task by transferring search knowledge without large time to train NAS frameworks from scratch on the unseen task. To this end, meta-NAS methods have proposed a task-adaptive performance predictor, which supports multiple unseen datasets with a single predictor. They outperformed baseline NAS methods on multiple benchmark datasets (Lee et al., 2021a), devices (Lee et al., 2021b), and real-world datasets (Jeong et al., 2021) and significantly reduced search time to less than a few GPU seconds on an unseen target task. The latter, zero-cost NAS methods (Mellor et al., 2021; Abdelfattah et al., 2021) have proposed several proxies that can be obtained from the first mini-batch so that do not require extensive training time.

However, while the benefit of the short search time of such rapid NAS methods is promising, they are sub-optimal to be used for AKD scenarios since they do not meet the aforementioned AKD conditions. For example, the existing all meta-predictors of MetaD2A (Lee et al., 2021a), TANS (Jeong et al., 2021), and HELP (Lee et al., 2021b) perform teacher-model-agnostic prediction that ignore the information of the teacher model. Furthermore, the accuracy predictors (Lee et al., 2021a; Jeong et al., 2021) are trained with simple amortized meta-learning, in which there is no gradient update-based adaptation to the target task. This enables fast prediction due to no need to collect architecture-accuracy data for further training on a target task, but there is a limit to the performance of the predictor. If we use zero-cost proxies on AKD scenarios, since zero-cost proxies predict the performance of student architectures based on random initialization, it might be sub-optimal on distilling knowledge from the teacher model to the student architecture.

Therefore, to overcome the heavy search cost of AKD methods and the sub-optimality of rapid NAS methods, we propose a rapid teacher-model-adaptive predictor for AKD scenarios. Following the previous meta-NAS works, we keep the advantage of meta-learning the predictor that can rapidly transfer to an unseen target task. The main different things are as follows: 1) We encode the information of the teacher model to search for student architecture that is optimal for the given teacher model. 2) We introduce a gradient-based rapid one-shot adaptive predictor by using the accuracy of the teacher model. This allows rapid adaptation by removing the need to train architectures on a target task to obtain sample accuracies for adaptation. Furthermore, we empirically show that remapping parameters from the teacher model to student architecture improves the prediction performance of zero-cost proxies in AKD situations.

We meta-learn the proposed method on subsets of TinyImageNet and neural architectures from the ResNet search space. Then we validate it to search for neural architectures on multiple unseen datasets such as MNIST, SVHN, CIFAR-10, CIFAR-100, Aircraft, and Oxford-IIIT Pets. In this experiment, our meta-learned model obtains a neural architecture within XX GPU seconds on average without direct training on a target task. We show that our proposed teacher-model-adaptive meta-predictor outperforms existing set-based meta-predictor on AKD tasks and the effectiveness of one-shot adaptation with an accuracy of teacher model.

To summarize, our contributions to this work are as follows:

- To our knowledge, we are the first rapid method for AKD, by proposing teacher-model-adaptive encoding and learn a predictor via one-shot meta-learning with accuracy of teacher model.
- We validate that combining parameter remapping with zero-cost proxies significantly improves the prediction performance on AKD.
- Our meta-learned model successfully searches for optimal student architectures to distill knowledge from teacher model on multiple unseen datasets and outperforming rapid NAS baselines.

2 Related Work

Parameter Remapping (PR)

Architecture-aware Knowledge Distillation (AKD)

Meta-learning and Meta-NAS Meta-learning (learning to learn) Thrun & Pratt (1998) aims to learn a model that generalizes over a distribution of tasks rather than a single task, such that the model meta-learned over a large number of tasks rapidly adapts to an unseen task. The performance of existing methods (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Nichol et al., 2018; Lee et al., 2019) is usually evaluated on few-shot classification tasks, where the model classifies between instances of unseen classes given only a few training examples. Recently, several works have proposed to utilize meta-learning for NAS. Most of them focus on few-shot classification tasks to search for the architectures and parameters that can generalize well to a new task (Elsken et al., 2020; Lian et al., 2019; Shaw et al., 2019) with gradient-based meta-learning. However, they have limited practical applicability since the computational cost of meta-learning is prohibitively large, for NAS under a standard many-shot setting. A recently proposed meta-NAS framework with amortized

meta-learning, MetaD2A (Lee et al., 2021a), which utilizes a set encoder to encode a task and uses a graph decoder to generate a task-adaptive architecture, obtained state-of-the-art performance on unseen datasets, with minimal search cost. We also propose a similar amortized meta-learning framework, for hardware-adaptive NAS, based on a novel hardware device embedding. However, after obtaining the device-conditioned initialization parameters, we perform further inner gradient steps for device-specific adaptation, unlike MetaD2A (Lee et al., 2021a).

3 METHOD

3.1 PROBLEM DEFINITION

Many AKD methods (Liu et al., 2020) have been included initial KD loss or initial accuracy as a proxy to evaluate the performance of student architecture candidates during the architecture search process. Assume that we are given $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}}), \mathcal{D}$, and $\{\mathbf{s}(\cdot;\boldsymbol{\theta})\}$ where $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}})$ is a teacher network parameterized by $\tilde{\boldsymbol{\theta}}$, which is trained on a dataset $\mathcal{D}$ and $\{\mathbf{s}(\cdot;\boldsymbol{\theta})\}$ are a set of student architecture candidates. Each student architecture is randomly sampled from search space $\mathcal{S}$. In general, we obtain a proxy with initial accuracy for each student architecture candidate ($\mathbf{s}$) via distilling knowledge from the teacher network $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}})$ as follows:

$$\boldsymbol{\theta}_0 \leftarrow \text{random initialization} \tag{1}$$

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \alpha \nabla_{\boldsymbol{\theta}_{i-1}} \mathcal{L}(\boldsymbol{\theta}; \mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}}), \mathbf{s}(\cdot;\boldsymbol{\theta}_{i-1}), \mathcal{B}_i) \quad \text{for } i = 1, \dots, N, \tag{2}$$

where α is a learning rate and mini-batch $\mathcal{B}_i$ is randomly sampled from the dataset $\mathcal{D}$. We conduct N distillation steps. $\mathcal{L}$ could be any KD-loss, e.g., Mean Square Error (MSE) loss between the output features of the teacher network and those of the student networks. We use the accuracy or loss of the given student model $\mathbf{s}(\cdot;\boldsymbol{\theta}_N)$ on the validation set as the proxy for the final accuracy of the model. We can obtain the final model by fine-tuning $\mathbf{s}$.

1. **Limitation of Existing AKD.** However, in general, ensuring to obtain accurate accuracy proxy requires a lot of distillation steps and time for each student architecture candidate. Such distilling to a large number of candidates is highly ineffective and not practical. Especially, when we consider the AKD scenarios with many tasks, the total search cost is very enormous. To shorten the multiple distillation steps for accurate accuracy, we empirically show that applying parameter remapping (PR) from teacher model to student candidate before the distillation phase can significantly shorten distillation steps (Section 3.2 and Experiment **??**), and accordingly we can reduce the required time/data for AKD to work as an accurate proxy, which is largely practical in real-world. Note that while a few NAS works (Fang et al., 2020a;b; Cai et al., 2018) use PR as an element for non-AKD, we focus on the effect of PR for AKD which needs to map the parameter from teacher model to student candidates. In adddition, our goal is to propose the PR method that can be applied to existing KDs orthogonally.
2. **Limitation of Existing PR.** While in Section 3.2, we show that applying PR before distillation steps reduces the search time effectively, it has a limitation that remaps parameters heuristically, ignoring task information such as dataset, parameters and architecture of a teacher, and architecture of a student which is critic for KD scenarios. Thus, in Section 3.3, we introduce to task-adaptively remap parameter by considering the dataset, teacher model, and student architecture. Such learning to remap parameters is novel compared with existing heuristic PR methods (Fang et al., 2020a;b; Cai et al., 2018). We allow this by designing meta-learning framework for AKD to learn to generalize over task distribution and transfer to unseen tasks. Furthermore, in Experiment **??**, we show that the proposed transferable Meta-NAS method outperforms existing Meta-NAS method (transferable accuracy predictor), MetaD2A (Lee et al., 2021a), which do not consider KD-objective for AKD.

3.2 REVISITING PARAMETER REMAPPING FOR AKD

We can define search space types roughly as a graph-based type (Liu et al., 2019; Ying et al., 2019; Dong & Yang, 2020) that selects one among predefined operation sets for each node and a stack

type (Lu et al., 2021; Cai et al., 2020; **?**), which designs each block differently by selecting the number of layers, and the number of channels of convolutional layers. In this work, we focus on the latter, stack type, because it usually used for edge-device setting on a classification task or a backbone of object detection. In other words, we only consider the weights of convolutional layers.

We remap the trained parameters $\tilde{\boldsymbol{\theta}}$ of the teacher model $\mathbf{t}$ to initial parameters $\boldsymbol{\theta}_0$ of a candidate student architecture $\mathbf{s} \sim \mathcal{S}$ before distillation phase. In other words, we replace Equation (1) as follows:

$$\boldsymbol{\theta}_0 = \boldsymbol{w} \circ \tilde{\boldsymbol{\theta}} \tag{3}$$

where $\boldsymbol{w}$ is a set of weighted values, that the sum of values are 1. In general, we can set the values of $\boldsymbol{w}$ heuristically. As the number of convolutional layers in neural architecture and the dimension of channel for each layer between the teacher model and candidate student architecture can be different, we design channel chunk-wise PR to be applied for stack-type search space generally. We assume that the architecture size of the teacher model is larger than that of student architecture. We divide channels as chunks of predefined size such as parameters of trained teacher model $\tilde{\boldsymbol{\theta}} = \{\tilde{\theta}^1, ..., \tilde{\theta}^{\tilde{C}}\}$ with the number of chunks $\tilde{C}$ and initial parameters of student architecture $\boldsymbol{\theta}_0 = \{\theta^1, ..., \theta^C\}$ with the number of chunks C. For example, let the teacher model has 3 convolutional layers with the dimension of channels $\{32, 64, 128\}$, student architecture has 2 convolutional layers with the dimension of channels $\{16, 32\}$ and the chunk size is 16. Then the number of parameter chunks with the size 16 of the teacher model $\tilde{C} = 2 + 4 + 8 = 14$ and the student architecture $C = 1 + 2 = 3$. We can represent Equation (3) as the weighted sum of chunks $\tilde{\boldsymbol{\theta}} = \{\tilde{\theta}^j\}_{j=1}^{\tilde{C}}$ of the teacher model and map it to a chunk of student architecture θ_0^k. as follows:

$$\theta_0^k = \sum_j^{\tilde{C}} w^{jk} * \tilde{\theta}^j \quad \text{where} \quad \sum_j^{\tilde{C}} w^{jk} = 1 \tag{4}$$

Algorithm 1 Meta-training

Input: $(\mathcal{D}_1, \tilde{\theta}_1), \ldots, (\mathcal{D}_T, \tilde{\theta}_T)$
Input: Architecture Set $\mathcal{S}$
Input: Learning rate α, β
Output: ϕ^*
Randomly initialize ϕ
while not converged **do**
  for $t = 1$ to T **do**
    Sample $s \sim \mathcal{S}$.
    Sample batches $\{\mathcal{B}_t\} \sim \mathcal{D}_t$
    Compute $w = f(\tilde{\theta}_t, s; \phi)$.
    Remap $\theta_t = w * \tilde{\theta}_t$.
    Compute $\mathcal{L}(\mathcal{B}_t; s, \theta_t, \phi)$.
  end for
  $\phi \leftarrow \phi - \beta \nabla_\phi \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}(\mathcal{B}_t; s, \theta_t, \phi)$
end while

Algorithm 2 Meta-test

Input: $\mathcal{D}, \mathcal{D}^{te} \tilde{\theta}, \phi^*$
Input: Architecture Set $\mathcal{S}$
Input: Learning rate α
Output: θ^*
while not converged **do**
  Sample $s \sim \mathcal{S}$.
  Sample batches $\{\mathcal{B}\} \sim \mathcal{D}$
  Compute $w = f(\tilde{\theta}, s; \phi^*)$.
  Remap $\theta = w * \tilde{\theta}$.
end while
Evaluate the test examples in $\mathcal{D}^{te}$ with the parameter θ^* and architecture s.

where $\boldsymbol{w}^k = \{w^{jk}\}_{j=1}^{\tilde{C}}$ is weighted values to remap θ_0^k. We apply Equation (4) to all chunks $\boldsymbol{\theta}_0 = \{\theta_0^k\}_{k=1}^{C}$ of the student architecture. Unlike KD-only performance proxy, we observe that including PR before KD phase improves the performance and efficiency of the proxy in Experiment **??**.

3.3 Learning to Remap Parameters for KD

In Section3.2, parameters are remapped heuristically, e.g., averaging parameters of chunks, selecting the former or the latter or uniformly parameters, which might not be optimal, and ignoring the task information such as parameters of teacher model $\tilde{\theta}$, architecture of student $\mathbf{s}$, and dataset $\mathcal{D}$. To overcome such limitations, we propose a task-conditioned weighted value predictor $f(;\boldsymbol{\phi})$ parameterized with $\boldsymbol{\phi}$ as follows:

$$\boldsymbol{w} = f(\tilde{\boldsymbol{\theta}}, \mathbf{s}, \mathcal{D}; \boldsymbol{\phi}) \tag{5}$$

Meta-training We learn ϕ to generalize across parameters of teacher models and student architectures to apply it to new AKD tasks. We achieve this goal with a meta-learning framework that can transfer knowledge obtained from the task pool $p(\tau)$ to an unseen task τ^{υ}. During the meta-training phase, we leverage the episodic training strategy that samples tasks randomly from the task pool $p(\tau)$, where each task τ consists of $\{\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau}), \mathcal{B}^{\tau}, \mathbf{s}^{\tau}\}$ where $\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau})$ is a teacher network parameterized by $\tilde{\boldsymbol{\theta}}^{\tau}$, which is trained on a dataset $\mathcal{D}^{\tau}$, $\mathcal{B}^{\tau}$ is randomly sampled from $\mathcal{D}^{\tau}$, and an student architecture $\mathbf{s}^{\tau}$ is randomly sampled from search space $\mathcal{S}$. We meta-train the weighted value predictor to minimize the KD loss $\mathcal{L}$ by optimizing the following task-conditioned meta-learning objective to obtain meta-learned $f(\cdot;\phi^{*})$ as follows:

$$\min_{\phi} \sum_{\tau \sim p(\tau)} \mathcal{L}(\tilde{\boldsymbol{\theta}}^{\tau}, \mathbf{s}^{\tau}, \mathcal{B}^{\tau}; \phi) \tag{6}$$

Task-conditioned Encoding Inspired by existing meta-nas frameworks (Lee et al., 2021a; Jeong et al., 2021; Lee et al., 2021b), we encode a given task as a single representation. However, note that our work is the first to encode the parameters of the teacher model, and while existing meta-nas frameworks have not been considered AKD environment. To encode the parameters of the teacher model, student architecture, and dataset, we leverage [TO BE UPDATED], respectively.

Teacher Network Embedding To encode a trained teacher network $\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau})$, we consider both its relative positions of parameter chunks and the model parameters trained on the dataset $\mathcal{D}^{\tau}$ for each task. Thus, we propose to generate a teacher network embedding with two encoding functions: 1) positional encoding and 2) functional encoding. Following (Vaswani et al., 2017), we obtain a positional encoding $\boldsymbol{v}_{t,p}^{\tau}$ using the number of total parameter chunks and the relative position of each chunk in the teacher network. Then our next goal is to encode the trained teacher parameter chunks for the given task, to consider the information contained in each chunk. Inspired by (Jeong et al., 2021), by feeding a fixed Gaussian random noise into the teacher network, we can get parameter chunk-output feature map pairs. Through this operation, information contained in each parameter chunk, which can be represented as a functional encoding vector, $\boldsymbol{v}_{t,f}^{\tau}$, can be obtained. With the two encoding vectors and corresponding embedding functions, the proposed teacher network embedding, $\boldsymbol{K}^{\tau}$ can be obtained by summing the positional embedding and functional embedding.

$$\boldsymbol{K}^{\tau} = \boldsymbol{f}_1(\boldsymbol{v}_{t,p}^{\tau}) + \boldsymbol{f}_2(\boldsymbol{v}_{t,f}^{\tau}) \tag{7}$$

Student Network Embedding When we consider a randomly sampled student network, we only know the number of total student parameter chunks and the corresponding architecture information. Therefore, we can get a student network embedding with following two encoding functions: 1) positional encoding and 2) architecture encoding. The former encoding method is the same as that of a teacher network. Following (Cai et al., 2020), we can obtain a architecture embedding $\boldsymbol{v}_{s,a}^{\tau}$ with auxiliary information about the architecture topology, such as the number of layers, and channel widths. [TO BE UPDATED]

$$\boldsymbol{Q}^{\tau} = \boldsymbol{g}_1(\boldsymbol{v}_{s,p}^{\tau}) + \boldsymbol{g}_2(\boldsymbol{v}_{s,a}^{\tau}) \tag{8}$$

Multi-head Attention Score between Teacher and Student [TO BE UPDATED]

$$\boldsymbol{K}^{\tau}\boldsymbol{W}^{K} = \text{Concat}(\boldsymbol{K}^{\tau}\boldsymbol{W}_1^{K}, \boldsymbol{K}^{\tau}\boldsymbol{W}_2^{K}, ..., \boldsymbol{K}^{\tau}\boldsymbol{W}_h^{K}) \tag{9}$$

$$\boldsymbol{Q}^{\tau}\boldsymbol{W}^{Q} = \text{Concat}(\boldsymbol{Q}^{\tau}\boldsymbol{W}_1^{Q}, \boldsymbol{Q}^{\tau}\boldsymbol{W}_2^{Q}, ..., \boldsymbol{Q}^{\tau}\boldsymbol{W}_h^{Q}) \tag{10}$$

$$\boldsymbol{e}_{\boldsymbol{i}}^{\tau} = \frac{Q^{\tau}W_i^{Q}(K^{\tau}W_i^{K})^{\top}}{\sqrt{d_k}} \tag{11}$$

$$\text{Attention}(\boldsymbol{Q}^{\tau}, \boldsymbol{K}^{\boldsymbol{\tau}}) = \text{softmax}(\sum_{i}^{h} z_i * {e_i}^{\tau}) \tag{12}$$

Student Network Encoding

Following Ryu et al. (2020), we practically consider T datasets during meta-training phase. Please refer to Algorithm (1) for meta-training task-conditioned weighted value predictor and supplementary material for the implementation details.

Meta-test We now describe how to use our meta-learned weighted value prediction model $f(\cdot;\boldsymbol{\phi}^*)$ to remap parameters of an architecture on a novel task τ^{υ} that is unseen during meta-training. Given a task $\tau^{\upsilon} = \{\mathbf{t}^{\tau^{\upsilon}}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau^{\upsilon}}), \mathcal{B}^{\tau^{\upsilon}}, \mathbf{s}^{\tau^{\upsilon}}\}$, we first obtain its task embedding. Then we use them as an input of our model to obtain the teacher parameters and student architecture (chunk) optimized weighted value. We sample a student architecture, remap its parameter from the target teacher and, estimate each student architecture based on the obtained performance from the remapped student architecture during AKD. Please refer to Algorithm (2).

4 Experiment

We first validate and analyze our predictor trained via meta-learning on predicting the accuracy of unseen student architectures when an unseen teacher and unseen dataset are given in Section 4.1. Second, we show the sample and search-time efficiency of the proposed predictor compared with existing task-specific predictors and few-shot meta-predictors in Section 4.2. Lastly, we demonstrate the performance of our model on AKD tasks by comparing our model with other rapid NAS methods which do not consider AKD environment in Section 4.3.

Search Space We consider ResNet search space (He et al., 2016) which is one of representative convolutional neural networks. Following (Cai et al., 2020; Lu et al., 2020), we perform a layer-wise search that selects the number of stages (blocks), the number of convolutional neural layers for each stage, and the channel expansion ratio for each layer as design choices. Our search space consists of 5 stages, and in each stage, the number of layers ranges across $\{1, 2, 3, 4\}$, and the channel expansion ratio should be chosen from $\{xx, xx, xx, xx\}$. This leads the search process to a choice out of $(5^1 + 5^2 + 5^3 + 5^4)^5 \approx xx^{xx}$. We use the largest architecture designed with 5 stages, 4 layers for each stage, and xx channel expansion for each layer as the teacher architectures for all tasks in the experiments.

Meta-Training Database Following Ryu et al. (2020), we use *TinyImageNet* (Le & Yang, 2015) as the source dataset for meta-training. It consists of 200 classes and 500 training images for each class. We generate a heterogeneous task samples by class-wisely splitting the dataset into 10 splits. We use 8 and 2 split datasets for meta-training and meta-validation, respectively. For each split dataset, we train a teacher architecture on the dataset with 180 epochs. We randomly sample 2000 student architectures and for each dataset and teacher model pair, collect 2000 student architecture-accuracy pairs after distilling knowledge from the teacher model. For each task, we consider the dataset and teacher pair, and randomly pick distilled student architecture-accuracy pair to train the proposed meta-predictor. Note that while the collecting meta-training database and meta-training is only once done for each search space, the meta-trained predictor is transferred to many AKD tasks with different datasets and teacher models with almost zero-cost (forward cost).

Transfer to Meta-Test Tasks: Unseen teacher model, unseen dataset, and unseen student architectures We transfer meta-learned predictor to meta-test tasks. Each meta-test task consists of an unseen dataset and an unseen teacher model trained on the unseen dataset and multiple unseen student architecture candidates. we consider four unseen datasets such as XX, XX, XX, and XX in all experiments. We use fixed 30 student architectures and distilling knowledge from the teacher model on each unseen dataset to compute Spearmanr ranking correlation experiments to demonstrate the accuracy predicting performance of prediction models in Section 4.1, 4.2, and 4.3. For end-to-end AKD problems in Section 4.3, we randomly sample XXX student architectures and pick one that the predicted performance is the highest depending on the prediction models and distill knowledge to the picked architecture from the target teacher model and dataset.

4.1 Efficacy of the Proposed Meta-predictor for AKD

Efficacy of Teacher Encoding MAML (Finn et al., 2017) and Meta-SGD (Li et al., 2017) are one of representative gradient based meta-learning models to learn to generalize to few-shot sample adaptation regression tasks. MetaD2A (Lee et al., 2021a) and TANS (Jeong et al., 2021) are the most closely related with our work, which are meta-learning based NAS and AutoML frameworks. Both frameworks have the accuracy prediction model and transfer them to unseen datasets after

Table 1: **Efficacy of the Proposed Meta-predictor for AKD.**

Method	Predictor Component				Unseen Dataset						
	Arch.	Data	Teacher	Update	CUB	Vgg Flower	GTRSB	Oxford-IIIT Pet	CIFAR 100	Tiny ImageNet (Valid)	Mean
MAML (Finn et al., 2017)	✓				0.494	0.459	0.041	0.255	x.xx	x.xx	0.312
Meta-SGD (Li et al., 2017)	✓				0.505	0.476	0.054	0.266	x.xx	x.xx	0.325
Ours (update 0)	✓	✓			0.691	0.772	0.342	0.258	x.xx	x.xx	0.516
MetaD2A (Lee et al., 2021a)	✓	✓			x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
TANS (Jeong et al., 2021)	✓	✓			x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
Ours (update 0)	✓	✓	✓		0.708	0.809	0.427	0.396	x.xx	x.xx	0.585
Ours (update 1)	✓	✓	✓	✓	**0.709**	**0.823**	**0.453**	**0.398**	**x.xx**	**x.xx**	**0.596**
Ours (update 2)	✓	✓	✓	✓	**0.734**	**0.845**	**0.458**	**0.425**	**x.xx**	**x.xx**	**0.616**

Table 2: **Time Saving of the Proposed Meta-predictor for AKD**

Target Dataset	Method	Corr.	Training			
			Sample Type	# of Samples	Collecting Time	Speed Up
Vgg Flower	Task-specific predictor	x.xxx	Student	xxx	xxx (s)	xxx×
		x.xxx	Student	xxx	xxx (s)	xxx×
	Few-shot meta-predictor	x.xxx	Student	5	xxx (s)	xxx×
		x.xxx	Student	3	xxx (s)	xxx×
		x.xxx	Student	1	xxx (s)	xxx×
	Ours	x.xxx	Teacher	1	xxx (s)	xxx×
CUB	Task-specific predictor	x.xxx	Student	xxx	xxx (s)	xxx×
		x.xxx	Student	xxx	xxx (s)	xxx×
	Few-shot meta-predictor	x.xxx	Student	5	xxx (s)	xxx×
		x.xxx	Student	3	xxx (s)	xxx×
		x.xxx	Student	1	xxx (s)	xxx×
	Ours	x.xxx	Teacher	1	xxx (s)	xxx×
GTRSB	Task-specific predictor	x.xxx	Student	xxx	xxx (s)	xxx×
		x.xxx	Student	xxx	xxx (s)	xxx×
	Few-shot meta-predictor	x.xxx	Student	5	xxx (s)	xxx×
		x.xxx	Student	3	xxx (s)	xxx×
		x.xxx	Student	1	xxx (s)	xxx×
	Ours	x.xxx	Teacher	1	xxx (s)	xxx×
Oxford-IIIT Pet	Task-specific predictor	x.xxx	Student	xxx	xxx (s)	xxx×
		x.xxx	Student	xxx	xxx (s)	xxx×
	Few-shot meta-predictor	x.xxx	Student	5	xxx (s)	xxx×
		x.xxx	Student	3	xxx (s)	xxx×
		x.xxx	Student	1	xxx (s)	xxx×
	Ours	x.xxx	Teacher	1	xxx (s)	xxx×

meta-training. The both accuracy models have set encoding scheme to perform set-adaptive accuracy prediction, yet, they do not have teacher encoding. As shown in Table 1, the meta-learning accuracy prediction models without considering a target dataset and teacher model such as MAML and Meta-SGD show poor performance on the unseen dataset and teacher models. While their results are better compared with MAML and Meta-SGD models since predictors of MetaD2A and TANS enable set-adaptive accuracy prediction, their prediction performances are sub-optimal compared with ours. Ours (update 0) with teacher encoding outperforms the baseline meta-learning models with the average correlation performance by XX. Furthermore, using gradient based inner update (Ours - update 1) on the target dataset and teacher models improve the prediction performance. This demonstrates that conducting inner update is better than amortized meta-learning (no inner update) of MetaD2A and TANS for accuracy prediction in AKD.

Efficacy of One-shot Inner Update with Teacher-Accuracy Pair

4.2 Time Saving of the Proposed Meta-predictor for AKD

Comparison with Task-specific Predictor

Comparison with Few-shot Adaptation of Meta-learning

Table 3: **Comparison with Rapid NAS Methods. / End-to-end AKD**

	Method	Correlation				End-to-end AKD			
		CUB	Vgg Flower	GTSRB	Oxford-IIIT Pet	Oxford 102 Flower	CUB	GTSRB	Oxford-IIIT Pet
	Params	-0.076	0.047	-0.196	0.096	x.xx	x.xx	x.xx	x.xx
	FLOPs	0.101	0.023	-0.211	0.143	x.xx	x.xx	x.xx	x.xx
Zero-cost Proxies	grad norm (Abdelfattah et al., 2021)	-0.091	-0.236	-0.242	0.156	x.xx	x.xx	x.xx	x.xx
	snip (Abdelfattah et al., 2021)	0.001	-0.163	-0.140	0.182	x.xx	x.xx	x.xx	x.xx
	grasp (Abdelfattah et al., 2021)	0.049	0.228	0.033	-0.003	x.xx	x.xx	x.xx	x.xx
	fisher (Abdelfattah et al., 2021)	0.071	-0.148	-0.245	0.090	x.xx	x.xx	x.xx	x.xx
	jacob cov (Abdelfattah et al., 2021)	-0.366	-0.176	-0.374	-0.007	x.xx	x.xx	x.xx	x.xx
	plain (Abdelfattah et al., 2021)	0.092	-0.176	0.090	0.119	x.xx	x.xx	x.xx	x.xx
	synflow bn (Abdelfattah et al., 2021)	-0.286	-0.436	-0.446	-0.127	x.xx	x.xx	x.xx	x.xx
	synflow (Abdelfattah et al., 2021)	-0.190	-0.290	-0.219	0.083	x.xx	x.xx	x.xx	x.xx
	NASWOT (Mellor et al., 2021)	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
Meta-learning	TANS (Jeong et al., 2021)	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
	MetaD2A (Lee et al., 2021a)	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
Ours (KD for 200 batches)	grad norm + PR	0.084	0.140	0.131	0.257	x.xx	x.xx	x.xx	x.xx
	snip + PR	0.236	0.243	0.155	0.292	x.xx	x.xx	x.xx	x.xx
	grasp + PR	0.332	0.150	-0.154	0.325	x.xx	x.xx	x.xx	x.xx
	fisher + PR	-0.337	-0.271	0.011	0.103	x.xx	x.xx	x.xx	x.xx
	jacob cov + PR	0.212	0.342	0.161	-0.077	x.xx	x.xx	x.xx	x.xx
	plain + PR	-0.220	-0.561	-0.173	-0.181	x.xx	x.xx	x.xx	x.xx
	synflow bn + PR	-0.332	-0.631	-0.271	-0.283	x.xx	x.xx	x.xx	x.xx
	synflow + PR	-0.185	-0.310	-0.189	0.066	x.xx	x.xx	x.xx	x.xx
	NASWOT + PR	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx	x.xx
	Meta-learning	**0.845**	**0.734**	**0.458**	**0.425**	**x.xx**	**x.xx**	**x.xx**	**x.xx**

4.3 Comparison with Rapid NAS Methods

4.4 End-to-end AKD

4.5 Efficacy of PR + Zero-cost Proxies for AKD

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *International Conference on Learning Representations (ICLR)*, 2021.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020.

Jiemin Fang, Yuzhu Sun, Kangjian Peng, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Fast neural network adaptation via parameter remapping and architecture search. In *International Conference on Learning Representations*, 2020a.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Kangjian Peng, Yuan Li, Wenyu Liu, and Xinggang Wang. Fna++: Fast network adaptation via parameter remapping and architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):2990–3004, 2020b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Wonyong Jeong, Hayeon Lee, Geon Park, Eunyoung Hyung, Jinheon Baek, and Sung Ju Hwang. Task-adaptive neural network search with meta-contrastive learning. *Advances in Neural Information Processing Systems*, 34:21310–21324, 2021.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Hayeon Lee, Eunyoung Hyung, and Sung Ju Hwang. Rapid neural architecture search by learning to generate graphs from datasets. In *International Conference on Learning Representations*, 2021a.

Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. Hardware-adaptive efficient latency prediction for nas via meta-learning. *Advances in Neural Information Processing Systems*, 34: 27016–27028, 2021b.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *In International Conference on Learning Representations (ICLR)*, 2019.

Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. Search to distill: Pearls are everywhere but not the eyes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7539–7548, 2020.

Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*, pp. 35–51. Springer, 2020.

Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):2971–2989, 2021.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning (ICML)*, pp. 7588–7598. PMLR, 2021.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Jeong Un Ryu, Jaewoong Shin, Hae Beom Lee, and Sung Ju Hwang. Metaperturb: Transferable regularizer for heterogeneous tasks and architectures. *Advances in Neural Information Processing Systems*, 33:11501–11512, 2020.

Albert Shaw, Wei Wei, Weiyang Liu, Le Song, and Bo Dai. Meta architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems (NIPS)*, 2017.

Sebastian Thrun and Lorien Pratt (eds.). *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-8047-9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems (NIPS)*, 2016.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning (ICML)*, pp. 7105–7114, 2019.