META-PREDICTION MODEL FOR DISTILLATION-AWARE NAS ON UNSEEN DATASETS

Anonymous authors
Paper under double-blind review

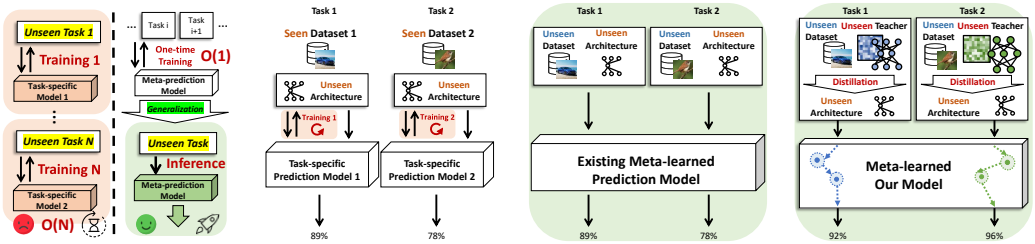
ABSTRACT

Distillation-aware Network Architecture Search (DaNAS) aims to search for an optimal student architecture that can obtain the best performance and/or efficiency when distilling the knowledge from a given teacher model. Previous DaNAS methods have mostly tackled the search for the network architecture for a fixed source/target tasks and the teacher, which are not generalized well on a new task, thus need to perform costly search for any new combination of the domains and the teachers. For standard NAS tasks without KD, meta-learning-based computationally efficient NAS methods have been proposed, which learn the generalized search process over multiple tasks and transfer the knowledge obtained over those tasks to a new task. However, since they assume learning from scratch without KD from a teacher, they might not be ideal for DaNAS scenarios, which could significantly affect the final accuracies of the architectures obtained from the search. To eliminate excessive computational cost of DaNAS methods and the sub-optimality of rapid NAS methods, we propose a distillation-aware meta accuracy prediction model which can predict a given architecture's final performances on a dataset when performing KD with a given teacher, without having to actually train it on the target task. The experimental results demonstrate that our proposed meta-prediction model successfully generalizes to multiple unseen datasets for DaNAS tasks, largely outperforming existing meta-NAS methods and rapid NAS baselines.

1 INTRODUCTION

Distillation-aware Neural Architecture Search (DaNAS) aims to search for an optimal student architecture which obtains the best performance and efficiency on a given dataset, when distilling the knowledge from the given teacher network to it (Liu et al., 2020; Gu & Tresp, 2020; Kim et al., 2022). For this task, we need to design a framework that considers the effect of Knowledge Distillation (KD), as a conventional NAS framework may be suboptimal for it. The reasons (Liu et al., 2020) are as follows: 1) For the same target dataset, an optimal student architecture for distilling the knowledge from the teacher and an optimal student architecture for learning from scratch with random initialization may be different. 2) For the same dataset, the optimal student architecture may depend on the choice of the teacher. To tackle such challenges, existing DaNAS methods guide the search process using the KD loss (Liu et al., 2020) or propose a proxy to evaluate distillation performance (Kim et al., 2022). However, such existing DaNAS methods do not generalize to multiple tasks and require training for any combination of dataset and teachers, and may result in excessive computational cost (e.g., 5 days with 200 TPUv2, for each task (Liu et al., 2020)). This hinders their applications to practical real-world scenarios, since optimal student architectures may vary depending on the type of datasets, teacher networks, and resource budgets. Therefore, we need a rapid and lightweight DaNAS method that can be generalized across different settings.

For standard NAS tasks without KD, there has been some progress in the development of rapid NAS methods that are computationally efficient, such as 1) meta-learning-based transferable NAS methods (Lee et al., 2021a;b; Jeong et al., 2021) and 2) zero-cost proxies (Mellor et al., 2021; Abdelfattah et al., 2021). The former, meta-NAS methods learn the generalized search process over multiple tasks, which allows it to adapt to a novel unseen task by transferring the knowledge obtained over those tasks to the new task, without having to train the NAS framework from scratch. To this end, meta-NAS methods generally utilize a task-adaptive prediction model, which rapidly adapts to novel datasets and devices. They outperform baseline NAS methods on multiple benchmark datasets (Lee



(a) Training Complexity (b) Task-specific Model (c) Meta-NAS (d) Ours

Figure 1: **Concept.** To search for a student architecture optimized for a distillation task, the prediction model should estimate the final accuracy of architecture differently depending on a dataset, a teacher network, a student architecture, and a distillation process. While existing meta-prediction models only support set-conditioned prediction, the proposed meta-prediction model performs the distillation-task-conditioned prediction.

et al., 2021a) and real-world datasets (Jeong et al., 2021), as well as with various devices (Lee et al., 2021b), significantly reducing the architecture search time to less than a few GPU seconds on an unseen setting. The latter, zero-cost NAS methods have proposed several proxies that can be obtained from the first mini-batch, without having to fully train the architecture on the target dataset.

However, despite the success of such rapid NAS methods on standard NAS tasks, they may be sub-optimal for DaNAS scenarios since they assume training from scratch without knowledge distillation from a teacher, which may have a large impact on the final accuracies of the architectures retrieved from the search. Therefore, to overcome the high search cost of DaNAS methods and the sub-optimality of rapid NAS methods, we propose a rapid distillation-aware meta-prediction model for DaNAS scenarios (Figure 1). Following the previous works on meta-NAS, we leverage meta-learning to learn a prediction model that can rapidly adapt to an unseen target task. However, our approach has two main differences in terms of the design of the prediction model and the meta-learning scheme, both of which are optimized for the distillation task. First, we propose a distillation-aware task encoding function that remaps parameters from the teacher to the student and considers the output from the embedding network of the remapped student as the embedding to estimate the teacher network's impact on the final accuracy of the distilled model. We use the encodings of the architectures as well as the tasks as the inputs to the meta-prediction model to estimate the final accuracy of the distilled model. Second, we use the accuracy of the teacher to guide the gradient-based adaptation of the meta-prediction model. This allows a more accurate and rapid estimation of the architecture's performance on a target task (dataset) with a specific teacher network.

We meta-learn the proposed distillation-aware prediction model on the subsets of TinyImageNet and neural architectures from the ResNet search space. Then we validate its prediction performance on multiple heterogeneous unseen datasets such as CUB, Stanford Cars, DTD, and STL10. The experimental results show that our meta-learned prediction model adapts to novel target tasks to estimate the final accuracy of an architecture distilled by an unseen teacher network within 3.45 (wall clock sec) on average without direct training on the target tasks. This is 1534.2$\times$ faster than the meta-prediction model with few-shot student architecture-accuracy samples which is time-consuming since it requires collecting time of few-shot samples for adaptation. Further, the DaNAS framework with the proposed distillation-aware meta-prediction model outperforms existing meta-NAS and zero-cost proxies on the same set of datasets.

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

- We propose a novel meta-prediction model that generalizes across datasets, architectures, and teacher networks, which can accurately predict the performance of an architecture when distilling the knowledge of the given teacher network.
- We propose a novel distillation-aware task encoding, based on the parameter remapping from a specific teacher to a student and the functional embedding of the remapped student.
- We enable a rapid gradient-based one-shot adaptation of the meta-prediction model on a target task by guiding it with a teacher-accuracy pair.
- We experimentally validate that our meta-prediction model successfully estimates the final accuracies of the student architectures for a given teacher network on multiple unseen datasets, significantly outperforming existing methods both in the correlation between prediction and GT accuracies, and in end-to-end DaNAS performances.

2 RELATED WORK

Distillation-aware NAS Knowledge Distillation (KD) is referred to as an efficient model compression technique under the resource budget limitations, by transferring the knowledge from the large teacher network to the smaller student network. Since distilling performance depends on student architectures as verified in Liu et al. (2020), it is important to search for a Pareto optimal student architecture given the dataset and teacher network. Recently, while Neural Architecture Search (NAS) suggests to automate designing the optimal architecture (Zoph & Le, 2017; Baker et al., 2017; Zhong et al., 2018; Real et al., 2017; Liu et al., 2018; Elsken et al., 2019; Real et al., 2019; Liu et al., 2019) that aims to search for an optimal architecture that has high accuracy when trained with only the ground-truth label, it might be suboptimal for searching for an architecture optimized for distilling given the teacher network. Liu et al. (2020) verified the assumption with the RNN-based actor-critic RL agent, which KD-trains sampled architecture for 5 epochs with a specific teacher to obtain KD-guided accuracy proxy. However, as it takes 5 days with 200 TPUv2 on account of distilling knowledge for each task, it cannot be applied to the real-world KD scenarios which are under the resource budget limitations. Kim et al. (2022) suggested the trust region with Bayesian optimization search algorithm and new KD-guided score. However, it cannot be generalized to multiple tasks which makes rapid search impossible. Gu & Tresp (2020) applied a structured pruning method for a good student to learn better from the teacher. They specify scaling factors for all channels of a teacher network and optimize that with KD-aware loss function. By removing the channels with zero scaling factors, a pruned student architecture can be obtained. However, as the optimization process should be performed repeatedly for each dataset and teacher network to get the optimal student architecture, the total time is too large.

Meta-learning and Meta-prediction Model Meta-learning (learning to learn) (Thrun & Pratt, 1998) allows a model to quickly adapt to an unseen task, by learning to generalize over distribution of tasks and transferring the meta-knowledge to the unseen task. One of the main meta-learning approaches is a gradient-based meta-learning (Finn et al., 2017; Li et al., 2017), which learns an initialization parameter shared across multiple tasks to be optimal for any tasks within few gradient-based updates from the initial parameter. Recently, meta-learning-based NAS (Meta-NAS) frameworks (Lee et al., 2021a; Jeong et al., 2021; Lee et al., 2021b) have been proposed, which are rapid and transferable to large-scale unseen tasks after one-time meta-learning. MetaD2A (Lee et al., 2021a) and TANS (Jeong et al., 2021) designed the accuracy prediction models and Lee et al. (2021b) proposed the latency prediction model. MetaD2A and TANS proposed the set encoder for task-adaptive accuracy prediction models. However, none of them do consider the teacher network and distillation process to design an architecture, thereby an architecture obtained by them might be suboptimal when distilling knowledge from the teacher network to the architecture. Due to the high cost and time to collect architecture-accuracy pairs as samples for adaptation on unseen tasks, they do not leverage the advantage of the few-shot adaptation on a target task. On the other hand, we utilize a teacher-accuracy pair for adaptation, allowing the predictive model to do gradient-based adaptation to new tasks. The additional required time is the time for a forward pass to obtain the accuracy of the teacher network.

3 BACKGROUND

This section describes existing accuracy prediction models designed for NAS and their limitations in using the models rapidly across multiple tasks of searching for optimal student architectures to distill the knowledge from teacher networks. In Table 1, we briefly summarize differences between task-specific accuracy prediction models (Task-spec.), existing meta-accuracy prediction models (MetaD2A (Lee et al., 2021a), TANS (Jeong et al., 2021)), and the proposed meta-accuracy model for DaNAS (Ours). As the task-specific model needs to be trained on each task, the total training time increases as the number of tasks increases ($O(N)$). Since meta-prediction models including ours can be generalized to multiple unseen tasks (datasets) after one-time meta-training, the training time is constant even if the number of tasks increases ($O(1)$). However, while our model can be generalized for unseen teacher networks (Unseen

Table 1: Comparison between accuracy prediction models for NAS and DaNAS.

	Task-spec.	MetaD2A	TANS	**Ours**
Time Compl.	$O(N)$	$O(1)$	$O(1)$	$O(1)$
Meta-learning	X	O	O	**O**
Unseen Arch.	O	O	O	**O**
Unseen Data.	X	O	O	**O**
Unseen Teach.	X	X	X	**O**
Distil.-aware	X	X	X	**O**
Fast Adapt.	X	X	X	**O**

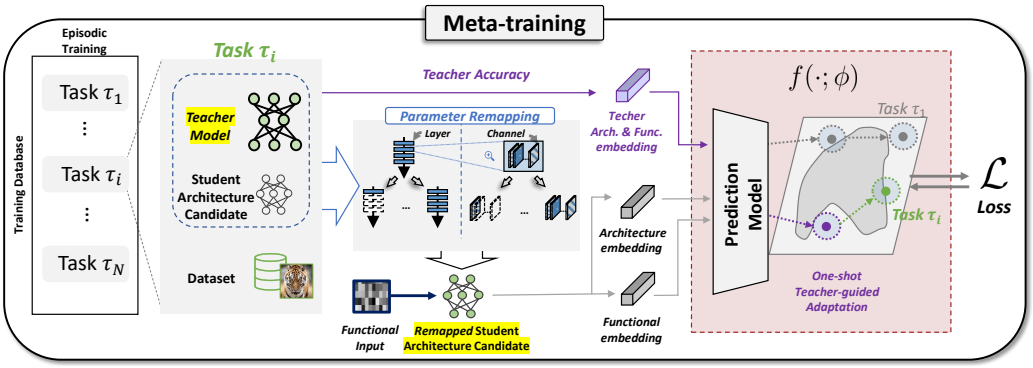


Figure 2: **Distillation-aware Meta-learning.** To search for the optimal student architecture to distill the knowledge from the given teacher network $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}})^{\tau}$ well, the sampled task should be encoded in a distillation-aware way. By taking a fixed input after remapping the trained parameters of teacher to the randomly sampled student candidate, we can encode the student candidate functionally in a teacher-adaptive way. Along with the functional embedding, by considering the architecture configuration of the student, we encode sampled task in a way appropriate for the distillation-aware NAS. Also, one-shot adaptation with teacher embedding which contains both architectural and functional information makes teacher-guided meta-learning becomes possible.

Teach.) and estimate distillation-aware accuracy (Distil.-aware), existing meta-prediction models have difficulty to do them. Furthermore, we empirically demonstrate the efficacy of the gradient-based one-shot fast adaptation with teacher-accuracy pair to improve the prediction performance of the accuracy prediction model on an unseen task in NAS (Fast Adapt.).

Task-specific Accuracy Prediction Model During search process, training each architecture candidate on a task until convergence to obtain its final performance, which is a single-scalar value, is wasteful. Thus, many NAS methods (Luo et al., 2018; Wang et al., 2020; Cai et al., 2020) leverage a task-specific accuracy prediction model as a proxy to reduce the excessive search cost spent on training the models. Let us assume a task specification $\tau = \{\mathbf{D}^{\tau}, \mathbf{S}, \mathbf{Y}^{\tau}\}$ where $\mathbf{D}^{\tau} \subset \mathcal{D}$ is a dataset, $\mathbf{S} \subset \mathcal{S}$ is a set of neural architectures, and $\mathbf{Y}^{\tau} \subset \mathcal{Y}$ is a set of accuracies obtained by training each architecture $s \in \mathbf{S}$ on the dataset $\mathcal{D}^{\tau}$. The task-specific accuracy prediction model $f^{\tau}(s;\phi) : \mathcal{S} \rightarrow \mathbb{R}$ parameterized by ϕ is trained for each task τ that estimates the accuracy $y^{\tau} \in \mathbf{Y}^{\tau}$ of a neural architecture $s \in \mathbf{S}$ for a given dataset $\mathbf{D}^{\tau}$ by minimizing empirical loss, $\mathcal{L}$ (e.g. mean squared error) defined on the predicted values $f^{\tau}(\mathbf{s};\phi)$ and actual measurements y^{τ} as follows:

$$\min_{\phi} \mathcal{L}\big(f^{\tau}\big(\mathbf{S};\phi\big), \mathbf{Y}^{\tau}\big) \tag{1}$$

However, the task-specific accuracy prediction model is highly limited for its practical application to real-world setting. Since it will not generalize to a new task (dataset), we need to learn a task-specific accuracy prediction model on the new target dataset. However, this is extremely time-consuming since we need to collect a large number of architecture-accuracy pairs (at least thousands samples), by training sampled architectures on the new dataset.

Meta-accuracy Prediction Model To address such inefficiency in task-specific NAS and enable transferrable NAS on multiple tasks, Lee et al. (2021a) and Jeong et al. (2021) proposed generalizable accuracy prediction models that can rapidly adapt to unseen datasets **within a few GPU seconds** after a single round of meta-learning. Specifically, such rapid search is made possible by meta-learning the prediction model conditioned on the given dataset: $f(s, \mathbf{D}^{\tau};\phi) : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{R}$. This enables it to support **multiple** datasets with a **single** prediction model since the predicted accuracy $y \leftarrow (s, \mathbf{D}^{\tau})$ is dependent on both the dataset type $\mathbf{D}^{\tau}$ and the architecture s. Then, they train the prediction model to generalize over the task distribution $p(\tau)$ by sampling task τ per iteration via amortized meta-learning as follows:

$$\min_{\phi} \sum_{\tau \sim p(\tau)} \mathcal{L}\big(f\big(\mathbf{S}, \mathbf{D}^{\tau};\phi\big), \mathbf{Y}^{\tau}\big) \tag{2}$$

After this one-time meta-learning, the set-conditioned accuracy prediction model $f(\cdot;\phi^*)$ with meta-learned parameters ϕ^* can be transferred to an unseen task $\hat{\tau}$ to search for an architecture optimized for $\hat{\tau}$. Note that the meta-learned prediction model works well on a unseen dataset $\hat{\mathbf{D}}^{\hat{\tau}}$ which has

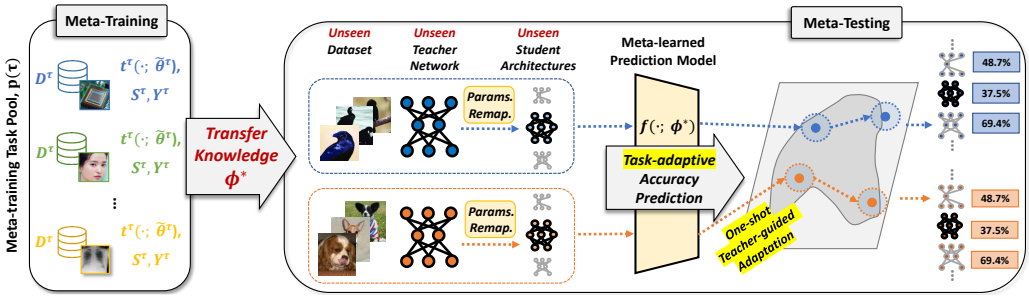


Figure 3: **Transfer to Unseen Task.** With transferred meta-knowledge ϕ^* learned from meta training, our meta-predictor $f(\cdot;\phi^*)$ can adapt to an unseen task $\hat{\tau}$. Especially, inner updating our meta-prediction model through a teacher network-accuracy pair, which is referred as the teacher-guide, the adaptation can be conducted quickly with almost no cost. Then, by remapping parameters of teacher network trained on $\mathbf{D}^\tau$ to the given student candidate, teacher-adaptive embedding of the student candidate can be obtained. Utilizing this teacher-aware embedding scheme, our meta-prediction model can rapidly output estimated distillation performance of the given student architecture on the unseen task.

no overlap with datasets $\{\mathbf{D}^\tau\}$ used in the meta-training phase. This reduces the meta-prediction model's complexity from $O(N)$ to $O(1)$ on N tasks.

Limitations While the meta-accuracy prediction models are promising as they enable rapid search on multiple tasks, directly using the existing meta-accuracy prediction model for distillation-aware NAS problems is suboptimal. This is because they do not consider the teacher and may be inaccurate for DaNAS scenarios due to following reasons:

1. **Distillation-independent accuracy prediction.** DaNAS is different from the existing NAS problems in that a task τ in distillation-aware NAS domain includes a teacher network $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}})^\tau$ which is parameterized by $\tilde{\boldsymbol{\theta}}$ and is trained on a dataset $\mathbf{D}^\tau$. Thus, for the same dataset, the optimal student architectures can be different if the teacher networks are different. However, existing meta-prediction model does not encode the teacher network and thus will search for the same student architecture for different teacher networks, which might be sub-optimal. Thus, we need a teacher-conditioned accuracy prediction model for DaNAS.
2. **No gradient-guided task adaptation.** Existing Meta-NAS frameworks aims to learn a generalized accuracy prediction for multiple tasks with amortized meta-learning conditioned on the task embeddings, omitting gradient-based adaptation steps (Finn et al., 2017; Li et al., 2017). This was mainly done to expedite the meta-training step and reduce the adaptation time on a new task, since gradient-based adaptation requires obtaining few-shot architecture-accuracy samples of the target dataset which takes at least a few GPU min to hours and renders rapid NAS impossible.

To address the above issues, we aim to design a teacher-conditioned accuracy prediction model and the accuracy of teacher-guided meta-learning for rapid DaNAS tasks. In our work, we reveal that the teacher-conditioned accuracy prediction is more accurate than set-conditioned accuracy prediction to predict the final performance of student architectures for DaNAS scnearios. In addition, we show that it is possible to perform gradient-guided task adaptation using the accuracy of teacher network. Please refer to concept Figure 2 which illustrates our distillation-aware accuracy prediction model.

4 METHOD

Distillation-aware Accuracy Prediction Model For each task $\tau = \{\mathbf{D}^\tau, \mathbf{S}, \mathbf{Y}^\tau, \mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau)\}$ from a distillation-aware NAS domain, we propose a distillation-aware accuracy prediction model $f(\cdot;\boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$ as follows:

$$f(s, \mathbf{D}^\tau, \mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}) : \mathcal{S} \times \mathcal{D} \times \mathcal{T} \rightarrow \mathbb{R} \tag{3}$$

that can estimate an accuracy of a student architecture s differently depending on the student architecture s, a dataset $\mathbf{D}^\tau$ and a teacher network $\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau)$. We encode a task τ by encoding the target dataset, teacher network, and the student architecture by simulating the distillation process. To this end, we encode each sampled student architecture candidate $s(\cdot;\boldsymbol{\theta})$ in a teacher-adaptive manner

by leveraging a parameter remapping $\tilde{\boldsymbol{\theta}}_s^\tau = g(\tilde{\boldsymbol{\theta}}^\tau)$ with the remapping function g from the parameters $\tilde{\boldsymbol{\theta}}^\tau$ of the teacher network trained on the dataset $\mathbf{D}^\tau$ to the parameters $\tilde{\boldsymbol{\theta}}_s^\tau$ of each student architecture $s(\cdot;\tilde{\boldsymbol{\theta}}_s^\tau)$, which can be done at almost no cost. Next, we embed each student candidate $s(\cdot;\tilde{\boldsymbol{\theta}}_s^\tau)$ except its final prediction layer using a functional embedding $\boldsymbol{h}_z^\tau$ by feeding in a fixed Gaussian random noise $\mathbf{z}$ (Jeong et al., 2021) as input as follows:

$$s(\mathbf{z};\tilde{\boldsymbol{\theta}}_s^\tau) = \boldsymbol{h}_z^\tau \tag{4}$$

In our search space, the depth and channel widths of the sampled student architecture s is different with those of the teacher network $\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau)$. We assume that in the distillation process, the teacher network has the same or larger size compared with a given student architecture, for each layer in each stage. Therefore, while there are several remapping techniques, we can simply copy and paste the parameters of the teacher network at the same position to those of the student architecture. Specifically, we consider a network is a composition of M stages, where each i-th stage corresponds to $\mathcal{B}_i$. By doing so, we represent the teacher network and a randomly sampled student architecture as:

$$\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau) = \tilde{\mathcal{B}}_M \circ \cdots \circ \tilde{\mathcal{B}}_1(\cdot;\tilde{\boldsymbol{\theta}}_1^\tau), \quad \mathbf{s}^\tau(\cdot;\theta) = \mathcal{B}_M \circ \cdots \circ \mathcal{B}_1(\cdot;\theta_1) \tag{5}$$

If the sampled student architecture has d layers (depth) and $k_1, k_2, ..., k_d$ channel width for each layer at stage m, remapping process is conducted hierarchically for each stage m, by first performing depth-level remapping followed by subsequent width-level remapping with remapping function(g), $\tilde{\boldsymbol{\theta}}_s^\tau = g(\tilde{\boldsymbol{\theta}}^\tau)$, as follows:

$$\tilde{\boldsymbol{\theta}}_{s,m,o}^\tau = \tilde{\boldsymbol{\theta}}_{m,o}^\tau \quad \text{for } o = 1, \ldots, d. \tag{6}$$

$$\tilde{\boldsymbol{\theta}}_{s,m,o,k}^\tau = \tilde{\boldsymbol{\theta}}_{m,o,k}^\tau \quad \text{for } k = 1, \ldots, k_o. \tag{7}$$

In addition, as suggested in Cai et al. (2020), the architecture configuration such as depth and channel widths of s at each stage can be represented as a one-hot embedding vector. Concatenating one-hot embedding vectors for all stages, we can obtain the embedding for the whole network architecture, $\boldsymbol{h}_s$. The accuracy prediction model takes two embeddings $\boldsymbol{h}_z^\tau$ and $\boldsymbol{h}_s$ as inputs and outputs the estimated accuracy, with additional nonlinear projection layers parameterized by $\boldsymbol{\phi}$. For the detailed descriptions, please see the supplementary document.

Adaptation Guided with Teacher-Accuracy Pair To obtain meta-learned $f(\cdot;\boldsymbol{\phi}^*)$, we meta-learn the proposed accuracy prediction model over task samples from the task distribution $p(\tau)$ as follows:

$$\min_{\boldsymbol{\phi}} \sum_{\tau\sim p(\tau)} \mathcal{L}\big(f\big(\mathbf{S}, \mathbf{D}^\tau, \mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}\big), \mathbf{Y}^\tau\big) \tag{8}$$

where $\mathcal{L}$ is the Mean Square Error loss. Differently from the previous meta-accuracy prediction models (Lee et al., 2021a; Jeong et al., 2021) based on the amortized meta-learning, we further perform a guided adaptation on the target task with the teacher network $\boldsymbol{t}^\tau$ and its accuracy $\tilde{y}^\tau$ evaluated on the target dataset, by conducting the inner gradient updates with them as follows:

$$\boldsymbol{\phi}_{(i+1)}^\tau = \boldsymbol{\phi}_{(i)}^\tau - \alpha \nabla_{\boldsymbol{\phi}_{(i)}} \mathcal{L}(f(\mathbf{t}^\tau, \mathbf{D}^\tau, \mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}_{(i)}), \tilde{y}^\tau) \quad \text{for } i = 1, \ldots, I \tag{9}$$

where i denotes the i-th inner gradient step, I is the total number of inner gradient steps, and α is the multi-dimensional inner learning rate vector (Li et al., 2017). This meta-learning scheme enables the accuracy prediction model to start from the knowledge learned from the meta-training tasks and rapidly adapt to the new task with the guidance from the teacher-accuracy pair. Note that while there are gradient-based meta-learning methods (Finn et al., 2017; Li et al., 2017; Lee et al., 2021b), our work is the first to train the meta-accuracy prediction model for distillation-aware NAS that can generalize to unseen datasets via gradient-based guided adaptation. Furthermore, by introducing the simple yet effective approach of using the teacher network and its accuracy pair as few-shot training instances for task adaptation, we significantly reduce the time-consuming collection of the student architecture-accuracy pairs for each task, which allows efficient adaptation to each task. The final meta-learning objective reflecting Equation (8) and (9) is given as follows:

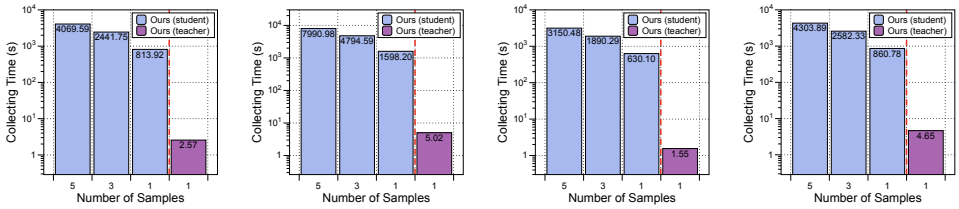
$$\min_{\boldsymbol{\phi}} \sum_{\tau\sim p(\tau)} \mathcal{L}\big(f\big(\mathbf{S}, \mathbf{D}^\tau, \mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}_{(I+1)}^\tau\big), \mathbf{Y}^\tau\big) \tag{10}$$

Table 2: **Efficacy of the Proposed Meta-prediction Model for DaNAS**

Method	Model Components				Unseen Datasets				
	Arch.	Set.	PR.	Gui.	CUB	SFC	DTD	STL	Mean
MAML (Finn et al., 2017)	✓			✓	0.57	0.57	0.20	0.20	0.34
Meta-SGD (Li et al., 2017)	✓			✓	0.57	0.57	0.22	0.21	0.39
MetaD2A (Lee et al., 2021a)	✓	✓			0.44	0.44	0.13	0.09	0.27
TANS (Jeong et al., 2021)	✓	✓			0.65	0.58	0.18	0.11	0.38
Ours (non-guided)	✓		✓		0.68	0.60	0.27	0.16	0.42
Ours (guided)	✓		✓	✓	**0.78**	**0.70**	**0.48**	**0.48**	**0.56**

Table 3: **Time and Sample Efficiency of the Proposed Meta-prediction Model for DaNAS**

Target Dataset	Method	Corr.	Training			
			Sample Type	# of Samples	Collecting Time	Speed Up
STL10	Few-shot meta-prediction model	0.297	Student	5	4303.89 (s)	1.000×
		0.267	Student	3	2582.33 (s)	1.667×
		0.219	Student	1	860.777 (s)	5.000×
	Ours	0.485	Teacher	1	4.654 (s)	924.771×



(a) CUB (b) SFC (c) DTD (d) STL

Figure 4: **Time and Sample Efficiency**

Adapting the Meta-prediction Model on Unseen Tasks (Meta-test) In Figure 3, we utilize the meta-learned prediction model $f(\cdot;\boldsymbol{\phi}^*)$ on a novel distillation task $\hat{\tau}$ that was unseen during meta-training. Given a task $\hat{\tau} = \{\mathbf{D}^{\hat{\tau}}, \mathbf{S}, \mathbf{Y}^{\hat{\tau}}, \mathbf{t}^{\hat{\tau}}(\cdot;\tilde{\boldsymbol{\theta}}^{\hat{\tau}})\}$, we first compute the accuracy $\tilde{y}^{\hat{\tau}} = \mathbf{t}^{\hat{\tau}}(\mathbf{D}^{\hat{\tau}};\tilde{\boldsymbol{\theta}}^{\hat{\tau}})$ of the teacher network through the forward process on the validation. Then, we update the meta-learned parameters $\boldsymbol{\phi}^{\hat{\tau}}_{(1)} = \boldsymbol{\phi}^*$ by adapting it to the unseen task by taking inner gradient steps, using the Equation (9) to obtain $\boldsymbol{\phi}^{\hat{\tau}}_{(I+1)}$ with $\{\mathbf{D}^{\hat{\tau}}, \mathbf{t}^{\hat{\tau}}(\cdot;\tilde{\boldsymbol{\theta}}^{\hat{\tau}}), \tilde{y}^{\hat{\tau}}\}$. Then, for each student architecture candidate s, we remap parameters and compute $\boldsymbol{h}^{\hat{\tau}}_z$ with Equation (4) and obtain $\boldsymbol{h}^{\hat{\tau}}_s$. Finally, we concatenate $\boldsymbol{h}^{\hat{\tau}}_z$ and $\boldsymbol{h}^{\hat{\tau}}_s$ and compute the estimated accuracy of s on the task $\hat{\tau}$ with $f(\boldsymbol{h}^{\hat{\tau}}_z, \boldsymbol{h}^{\hat{\tau}}_s; \boldsymbol{\phi}^{\hat{\tau}}_{(I+1)})$.

5 Experiment

We first validate and analyze our prediction model trained via meta-learning on predicting the distillation performance of unseen student architectures on an unseen dataset, with an unseen teacher in Section 5.1. Second, we report the time efficiency of the proposed prediction model compared with few-shot meta-prediction models in Section 5.2. Lastly, we demonstrate the performance of our model on end-to-end DaNAS tasks compared against existing rapid NAS methods in Section 5.3.

Search Space We consider the ResNet search space (He et al., 2016) consisting of 4 stages. Similarity to previous works (Cai et al., 2020; Lu et al., 2020), we perform depth-wise search to select the number of layers of each stage and channel-wise search to select the channel shrink ratio for each layer, allowing the search process to choose among 10^{11}. We use the largest architecture as teacher architectures for all experiments except for Section 5.4, where we report the results with different teacher architectures. For more descriptions, see the supplemental document.

Meta-Training Database Following Ryu et al. (2020), we use TinyImageNet (Le & Yang, 2015) as the source dataset for meta-training.We generate a heterogeneous set of tasks by class-wisely splitting the dataset into 10 splits. We use subsets consisting of 8 and 2 splits for meta-training and meta-validation, respectively. For each sub-dataset, we train a teacher architecture on it for 180 epochs. We randomly sample 2000 student architectures and, for each dataset-teacher network pair, collect 200 student architecture-accuracy pairs after distilling knowledge from the teacher

Table 4: **Comparison with Rapid Proxies for the Unseen DaNAS Tasks**

	Method	Unseen Datasets					Search Time (GPU sec)				
		CUB	SFC	DTD	STL	Mean	CUB	SFC	DTD	STL	Mean
Zero-cost proxies	grad norm (Abdelfattah et al., 2021)	-0.07	0.09	0.19	0.22	0.10	0.98	1.00	1.11	2.11	1.30
	snip (Abdelfattah et al., 2021)	0.11	0.27	0.24	0.40	0.25	1.05	1.03	0.97	2.14	1.30
	grasp (Abdelfattah et al., 2021)	-0.15	-0.06	-0.19	-0.10	-0.12	1.37	1.34	1.34	2.56	1.65
	fisher (Abdelfattah et al., 2021)	0.03	0.08	0.20	0.27	0.14	0.97	1.04	1.05	2.10	1.29
	plain (Abdelfattah et al., 2021)	0.12	0.22	0.26	0.34	0.23	0.90	1.01	1.03	2.10	1.26
	synflow (Abdelfattah et al., 2021)	-0.10	0.13	0.17	**0.50**	0.17	0.85	0.98	0.97	2.16	1.24
	NASWOT (Mellor et al., 2021)	-0.45	0.15	0.32	0.29	0.07	0.92	1.00	1.02	2.03	1.24
Meta-learning	MetaD2A (Lee et al., 2021a)	0.44	0.44	0.13	0.09	0.27	0.01	0.01	0.01	0.01	0.01
	TANS (Jeong et al., 2021)	0.65	0.58	0.18	0.11	0.38	0.01	0.01	0.01	0.01	0.01
	Ours	**0.78**	**0.70**	**0.48**	0.48	**0.56**	0.05	0.05	0.05	0.05	0.05

Table 5: **End-to-end Distillation-aware NAS**

	Method										Teacher
	grad norm	snip	grasp	fisher	plain	synflow	NASWOT	MetaD2A	TANS	**Ours**	
CUB	17.26	17.26	19.86	17.26	21.89	16.72	23.33	22.59	22.83	**23.85**	23.30
SFC	29.89	29.89	11.19	29.89	29.89	23.16	34.30	30.53	27.60	**34.79**	62.70
STL	41.34	41.34	36.16	49.86	41.34	36.44	44.59	43.25	47.23	**50.00**	70.77
DTD	16.16	16.16	16.16	12.82	16.54	14.98	14.33	17.89	12.82	**19.23**	21.44

network. For each task, we consider the dataset-teacher pair, while randomly selecting distilled student architecture-accuracy pair to train meta-prediction models. Note that the collection of the meta-training database and meta-training is done **only once** for each search space, as the meta-trained prediction model can be transferred to any DaNAS tasks with different unseen datasets and teacher networks at almost no cost (forward cost to get teacher network accuracy). The time to meta-train our prediction model, which takes only once, is 2 hours with a single NVIDIA's RTX 2080 Ti GPU.

Generalization to Unseen Datasets We transfer the meta-knowledge of the prediction model trained over meta-training tasks to meta-test tasks. Each meta-test task consists of an unseen dataset and an unseen teacher network trained on the unseen dataset and multiple unseen student architecture candidates. We consider four unseen datasets such as CUB (Wah et al., 2011), Stanford Cars (SFC) (Krause et al., 2013), DTD (Cimpoi et al., 2014), STL10 (STL) (Coates et al., 2011) in all experiments. We use fixed 30 student architectures and distill knowledge from the teacher network on each unseen dataset to compute Spearman's rank correlation coefficient (the higher the better) between the estimated and true accuracies to report the performance of the prediction models in Section 5.1, 5.2, and 5.3. For end-to-end DaNAS experiments in Section 5.3, we randomly sample 10,000 student architectures and select the architecture with the highest predicted performance using prediction models, and distill knowledge to the selected architecture with the teacher network and dataset. For more descriptions of **training details** and **experimental setups**, see the suppl. document.

5.1 Efficacy of the Proposed Meta-prediction Model on Unseen Datasets

We validate whether our prediction model and meta-learning strategy help improve the performance of the accuracy prediction model. In Table 2, ours (guided) outperforms all rapid meta-prediction models (Finn et al., 2017; Li et al., 2017; Lee et al., 2021a; Jeong et al., 2021) on four unseen datasets. MAML and Meta-SGD that learn accuracy prediction models by taking only the architecture configuration of a student candidate as an input show poor performance, as they predict the distillation performance without consideration of a target dataset and teacher network. Set-conditioned prediction models (MetaD2A and TANS) also achieve significantly lower performance than ours since the search is done in a teacher-agnostic manner, which has a large impact on the performance of the student trained with distillation. On the other hand, by utilizing the proposed teacher-parameter-remapping-based embedding, ours (guided) significantly outperform all baselines on all 4 unseen tasks. Furthermore, we observe that adaptation to target task by guiding our meta-prediction model with the accuracy of the teacher network on the dataset largely improves the prediction performance over the non-guided search, ours (non-guided). In sum, the results show the clear advantage of both the design of our prediction model and the teacher-guided meta-learning scheme.

5.2 Efficiency of Teacher Guided Accuracy Prediction

In general, guided search such as few-shot adaptation on the target task requires collection of few-shot architecture-accuracy pairs. To obtain even a single pair of architecture-accuracy, we should train the architecture until it fully converges, which prevents rapid search at meta-test time. However, the

DaNAS environment is different from the conventional NAS, in that we already have trained model, which is the teacher network. There is no need to wait to get the accuracy-architecture pair(s) for adaptation and since we can get the accuracy of the teacher with a single forward pass. In Table 3 and Figure 4, we report the time efficiency of meta-prediction model, compared against that of the baselines, when adapting to unseen tasks with the student architecture-accuracy pair(s). Our model only needs 3.45(s), on average on the four target datasets, which significantly reduces the adaptation time. Interestingly, the prediction performance is the best when we use the teacher network-accuracy pair rather than the student architecture-accuracy pair(s). As the number of few-shot samples for an adaptation increases, the Spearman's rank correlation coefficient result gets better, but even increasing the number of adaptation samples until 5, our adaptation strategy outperforms other baselines.

5.3 Comparison with Rapid NAS Methods

In Table 4, we validate our meta-prediction model with other rapid performance prediction methods. Zero-cost proxies calculate the scores of architecture candidates by feeding a few batches of target dataset into randomly initialized architectures. Therefore, they are efficient, but cannot predict the distillation performance well consistently across all unseen datasets. Ours outperforms all rapid NAS baseline models, including six zero-cost proxies (Abdelfattah et al., 2021; Mellor et al., 2021) and four meta-accuracy prediction models (Finn et al., 2017; Li et al., 2017; Lee et al., 2021a; Jeong et al., 2021) on four unseen datasets and shows competitive results compared to synflow on STL10.

We further validate the proposed meta-prediction models by conducting end-to-end DaNAS tasks on four unseen datasets. For a fair comparison, we randomly sample 10,000 student architectures and select the top-1 student architecture whose estimated performance by each method is the highest, and then distill the knowledge from the same teacher network trained on the target dataset to the selected student architecture with the same number of epochs. In Table 5, the student architecture obtained by our model consistently obtains better distillation performance than baselines on unseen datasets. The average search time is 0.05 GPU seconds, which is competitive with other rapid NAS methods.

5.4 Different teacher networks

We further experiment to show that the optimal student architectures for the same dataset vary across the specific teacher model we distill the knowledge from. We consider two teacher networks trained on the same dataset, STL10 but with different architectures and different number of parameters, 2.75M and 5.89M respectively. We search for the optimal student architecture for each teacher network by our meta-learned model with the same process as described in Section 5.3. We distill knowledge from both teacher networks to each obtained students and finetune the students. In Table 6, for Teacher 2 (T2), performance of the student (71.04 (%)) retrieved for Teacher 1 (T1) is lower than the student (72.19 (%)) retrieved for Teacher 2 (T2) and vice versa. This shows the clear necessity of the teacher-conditioned architecture search in the DaNAS scenarios.

Table 6: **The Effect of Teacher-aware NAS**

	Teacher 1 (T1)	Teacher 2 (T2)
Student for T1	65.76	71.04
Student for T2	65.69	72.19

6 Conclusion

We proposed a novel distillation-aware meta-accuracy prediction model, that can estimate the final accuracy of an unseen student architecture trained on an unseen dataset with knowledge distillation from an unseen teacher network. Existing task-independent DaNAS models require collecting a large number of architecture-accuracy samples as the training set, for each dataset-teacher-architecture triplet. The task-conditioned meta-prediction models, on the other hand, can generalize to new datasets, but they are suboptimal in that they do not consider the teacher. Our distillation-aware meta-prediction model, on the other hand, performs rapid gradient-based adaptation to a new teacher and a dataset using the teacher-accuracy pairs within a few GPU seconds. Furthermore, we proposed a novel distillation-aware task encoding that first remaps parameters from a specific teacher network to a student and then considers the embedding of the remapped student as the task encoding. Our prediction model can estimate the final accuracy of the student by considering the impact of distillation with the specific teacher network. The experimental results demonstrated that the proposed model outperforms existing rapid NAS methods such as meta-NAS methods and zero-cost proxies for the accuracy estimation and end-to-end DaNAS on the heterogeneous unseen tasks, respectively.

Ethics Statement This work has no concerns about the ethical implications.

Reproducibility Statement First, we attached the code referred to as *codes.zip* of the proposed method to help with reproducibility as a supplemental material file. Second, in the appendix document (after references), we enable the experimental results in the main paper to be reproduced by providing a detailed description of the materials and elaborating on the experiment setup, which is organized as follows:

- **Section A** - We describe the *implementation details* of functional embedding, architecture embedding and distillation-aware student architecture embedding as the final input of our model, and meta-surrogate performance prediction model.
- **Section B** - We provide the details of the *search space* that we used in the all experiments in the main document.
- **Section C** - We provide the detailed descriptions of *baselines* compared against our mode in the experiments of the main document.
- **Section D** - We elaborate on the detailed *training details*, such as meta-training the proposed prediction model and distilling knowledge from given teacher network to the student architecture, corresponding to the experiments introduced in the main document.
- **Section E** - We provide additional *experimental setup* about time measurements and conditions to sample student architecture in the search space.

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.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

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.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *International Conference on Learning Representations (ICLR)*, 2019.

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.

Jindong Gu and Volker Tresp. Search for better students to learn distilled knowledge. In *ECAI*, 2020.

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.

Taehyeon Kim, Heesoo Myeong, and Se-Young Yun. Revisiting architecture-aware knowledge distillation: Smaller models and faster search. *arXiv preprint arXiv:2206.13130*, 2022.

Animesh Koratana, Daniel Kang, Peter Bailis, and Matei Zaharia. LIT: Learned intermediate representation training for model compression. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3509–3518. PMLR, 09–15 Jun 2019.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

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.

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.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations (ICLR)*, 2018.

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.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems (NeurIPS)*, 2018.

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.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning (ICML)*, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence (AAAI)*, 2019.

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.

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

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.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sks9_ajex.

Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2423–2432, 2018.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017.

A IMPLMENTATION DETAILS

A.1 FUNCTIONAL EMBEDDING

We consider a network is a composition of M stages, where each i-th stage corresponds to $\mathcal{B}_i$. By doing so, we can represent the teacher network and a randomly sampled student architecture as:

$$\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau}) = \tilde{\mathcal{B}}_M \circ \cdots \circ \tilde{\mathcal{B}}_1(\cdot;\tilde{\boldsymbol{\theta}}_1^{\tau}), \quad \mathbf{s}^{\tau}(\cdot;\theta) = \mathcal{B}_M \circ \cdots \circ \mathcal{B}_1(\cdot;\theta_1) \tag{11}$$

In our search space, the depth and channel widths of the sampled student architecture s is different with those of the teacher network $\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau})$. Also, we assume that in the distillation process, the depth and channel widths for each layer in each stage of given teacher network are always greater than or equal to the student. For simulating the distillation process, we encode each sampled student architecture in a teacher-adaptive manner with remapping function, g.

Since the corresponding architecture configuration of teacher network and sampled student architecture is different, remapping process is conducted hierarchically for each stage m, by first performing depth-level remapping followed by subsequent width-level remapping. For the sampled student architecture, which has d depth and $k_1, k_2, ..., k_d$ channel width for each depth at stage m, remapped student parameters, $\tilde{\boldsymbol{\theta}}_s^{\tau} = g(\tilde{\boldsymbol{\theta}}^{\tau})$, is as follows:

$$\tilde{\boldsymbol{\theta}}_{s,m,o}^{\tau} = \tilde{\boldsymbol{\theta}}_{m,o}^{\tau} \quad \text{for } \ o = 1, \ldots, d. \tag{12}$$

$$\tilde{\boldsymbol{\theta}}_{s,m,o,k}^{\tau} = \tilde{\boldsymbol{\theta}}_{m,o,k}^{\tau} \quad \text{for } \ k = 1, \ldots, k_o. \tag{13}$$

With remapped student candidate, $s(\cdot;\tilde{\boldsymbol{\theta}}_s^{\tau})$, it is also important to get a functional embedding which has task-agnostic shape. The spatial dimension of final output logit for the given network varies across tasks and the depth and channel width within the same stage are also different for each sampled student architecture s. However, the output dimension of feature map at each stage is the same across for all tasks from the task pool $p(\tau)$. By feeding a fixed Gaussian random input tensor into the remapped student candidate, we can obtain teacher-conditioned network output feature map (attention map), $T \in \mathbb{R}^{C\times H\times W}$ at the last stage of student, $\mathcal{B}_M$ while satisfying the condition of getting a task-agnostic shape. By applying a mapping function, $\mathcal{F} : \mathbb{R}^{C\times H\times W} \rightarrow \mathbb{R}^{H\times W}$ to the feature map, the output can be used as the teacher-adaptive functional representation of the corresponding student candidate with respect to the fixed input tensor. Inspired by Zagoruyko & Komodakis (2017), we sum up absolute values across the channel dimension as follows: $\boldsymbol{F}(T) = \sum_{i=1}^{C} |T_i|$, then flatten and L_2 normalize it to obtain the functional embedding vector, $\boldsymbol{m}$, followed by functional projection layer, q, which outputs the projected 64-dimensional functional embedding vector, $q(\boldsymbol{m})$. Since we leverage the parameters of given teacher network, $q(\boldsymbol{m})$ can be teacher-adaptive.

A.2 ARCHITECTURE EMBEDDING

Our prediction model takes an architecture configuration of sampled student architecture which contains information such as depth and channel widths. ResNet search space is composed of 4 stages and 5 layers for each stage, the total number of layers is 20. For each layer, the channel shrink ratio should be chosen from $\{1, 2, 3, 4, 5, 6, 7, 8\}$. Similar to one-hot encoding scheme used in (Cai et al., 2020), by concatenating 1-dimensional vector which contains the information of image size, which is also referred as a resolution, we can construct one-hot architectural encoding, $\boldsymbol{a}$, in a 161-dimensional vector. Then, by passing the one-hot architecture encoding vector to architectural projection layer p, we can get an 64-dimensional architecture embedding vector, $p(\boldsymbol{a})$

A.3 DISTILLATION-AWARE STUDENT ARCHITECTURE EMBEDDING

With $p(\boldsymbol{a})$ and $q(\boldsymbol{m})$, which represents architecture embedding and teacher-adaptive functional embedding of sampled student architecture respectively, we concatenate those two embedding vectors and learn another non-linear projection layer, σ, which outputs distillation-aware network embedding with 64 dimensions.

B SEARCH SPACE

We consider the ResNet search space (He et al., 2016). Our ResNet search space consists of 4 stages, and for each stage, the base channel width is $\{32, 64, 128, 256\}$. Similarly to (Cai et al., 2020; Lu et al., 2020), we perform depth-wise search to select the number of convolutional neural layers for each stage, and then perform channel-wise search to select the channel shrink ratio for each layer. The number of layers ranges across $\{1, 2, 3, 4, 5\}$, and the channel shrink ratio for each layer should be chosen from $\{1, 2, 3, 4, 5, 6, 7, 8\}$. For example, in stage 1 where the base channel width is 32, if the selected channel shrink ratio is 4, the channel width of the corresponding layer is $32/4 = 8$. For all networks in the search space, the channel shrink ratio of the last layer of each stage is fixed to 1 in order to make the dimension of output feature the same for each stage. This leads the search process to a choice out of $(8^1 + 8^2 + 8^3 + 8^4)^4 \approx 10^{11}$. We use the largest architecture with 4 stages, each of which contains 5 layers, taking the corresponding base channel width for all layers as teacher architectures, whose number of parameters is 5.89M, for all experiments except for the experiments in Section 5.4, where we report the results with different teacher architectures.

C BASELINES

In this section, we describe the rapid NAS baselines which we compared with our meta-prediction model. The baseline methods can be divided into two categories, 1) Zero-cost proxies, 2) Meta-prediction model.

C.1 ZERO-COST PROXIES

Abdelfattah et al. (2021) proposed some metrics based on pruning techniques that can quickly find architectures by leveraging a few batches of target dataset with random state architectures. A metric that measures how a neural network's activations correlate when it is exposed to various inputs over a few batches of the target dataset was also proposed by Mellor et al. (2021). Unlike other methods that require training an sampled architecture at least a few epochs as a proxy task, they have received a lot of attention as very rapid NAS becomes possible.

C.2 META-PREDICTION MODEL

MAML (Finn et al., 2017) and Meta-SGD (Li et al., 2017) are representative gradient based meta-learning models to learn to generalize to few-shot regression tasks. These baselines perform set- and distillation-independent prediction on a each task. MetaD2A (Lee et al., 2021a) and TANS (Jeong et al., 2021), which are the most closely related to our work, learn dataset-conditioned accuracy prediction models that can generalize to unseen datasets.

D TRAINING DETAILS

In this section, we describe the details of two main training training schemes used in our experiments as follows: 1) Meta-training the proposed prediction model and 2) Knowledge distillation from given teacher network to the student architecture.

D.1 META-TRAINING THE PROPOSED PREDICTION MODEL

We learn our prediction model over task distribution via episodic training Vinyals et al. (2016). For each episode, the number of meta-batch is 8, in other words, we sample randomly sample 8 tasks from the training database. Each task consists of a specific teacher network trained on a specific dataset, the accuracy of teacher network measured on the dataset, 30 student architecture candidates, and their final accuracy obtained by distilling the knowledge from the teacher network to each student architecture. The total number of episode is 500 and we used Adam optimizer with the learning rate 1e-4. As mentioned in Equation (9), we utilize the teacher network-accuracy pair for one-shot adaptation. When conducting the inner gradient updates with the teacher-accuracy pair, the total number of inner gradient step is set to one. The initial value of each element in the multi-dimensional inner learning rate vector for our prediction model and Meta-SGD (Li et al., 2017) is set to 1e-2. After the one-shot teacher-guided adaptation, the outer loss is calculated with randomly sampled 30 student architecture-distillation performance pairs for the given teacher network and dataset. During

meta-training, we select the prediction model with the smallest meta-validation loss as our final model.

D.2 KNOWLEDGE DISTILLATION

As mentioned before, our ResNet search space is composed of M stages which is set to 4 in our experiment setting. In the search space, before the first stage (stage 1), there is a convolutional layer, which referred as a stem, and there is a fully connected layer, which is referred as a tail after the last stage (stage M).

We just copied the stem and tail of given teacher network to the student architecture because all networks in the search space has the same dimension in the stem and tail. To distill the knowledge of teacher network to the sampled student architecture except for stem and tail, we utilized KD-aware loss function following Koratana et al. (2019). The total loss function is composed of KD loss term and Intermediate Representation (IR) loss term as follows:

$$\mathcal{L}_{total}(\mathbf{s}^\tau, \mathbf{t}^\tau; \mathbf{D}^\tau, y) = \beta \cdot \mathcal{L}_{KD,\alpha}(\mathbf{s}^\tau, \mathbf{t}^\tau; \mathbf{D}^\tau, y) + (1-\beta) \cdot \mathcal{L}_{IR}(\mathbf{s}^\tau, \mathbf{t}^\tau; \mathbf{D}^\tau) \tag{14}$$

with $\alpha, \beta \in$ [0, 1], which corresponds to an interpolation parameter in KD loss and total loss. We set those hyper-parameter values as 0.95, 0.75 respectively. $\mathbf{D}^\tau$ is the target dataset and y is the ground-truth label.

To distill the knowledge of the teacher network to the sampled student architecture, we need to minimize the KD-divergence between the probabilities of the teacher and student network. The probabilities of teacher and student network is as follows:

$$p^T_{t^\tau,i} = \frac{\exp(z_{t^\tau,i}/T)}{\sum_j \exp(z_{t^\tau,j}/T)}, \quad p^T_{s^\tau,i} = \frac{\exp(z_{s^\tau,i}/T)}{\sum_j \exp(z_{s^\tau,j}/T)} \tag{15}$$

where $z_{t^\tau,i}$ and $z_{s^\tau,i}$ are ouputs before the last softmax layer, which is referred as logits, and T is temperature hyper-parameter, which means the degree to soften logits. We use T as 10 for all experiments in ours. With H, which is cross-entropy loss, the KD loss $\mathcal{L}_{KD,\alpha}$ is as follows:

$$\mathcal{L}_{KD,\alpha}(p^T_{s^\tau,i}, p^T_{t^\tau,i}; \mathbf{D}^\tau, y) = \alpha \cdot H(y, p^T_{s^\tau,i}) + (1-\alpha) \cdot H(p^T_{s^\tau,i}, p^T_{t^\tau,i}) \tag{16}$$

As mention in Koratana et al. (2019), if given student and teacher network can be divided into several stages such that the dimension of input and output to be the same at each stage, knowledge distillation can be conducted utilizing L_2 loss between the output feature maps of student and teacher at each stage. With those output feature maps, the full IR loss is as follows:

$$\mathcal{L}_{IR}(x;\, \mathbf{s}^\tau, \mathbf{t}^\tau, \mathbf{D}^\tau) = L_2(\mathcal{B}_1(x),\, \tilde{\mathcal{B}}_1(x)) + \sum_{m=2}^{M} L_2(\mathcal{B}_m(\tilde{\mathcal{B}}_{m-1} \circ \cdots \circ \tilde{\mathcal{B}}_1(x)),\, \tilde{\mathcal{B}}_m \circ \cdots \circ \tilde{\mathcal{B}}_1(x)) \tag{17}$$

For every knowledge distillation process, we distill the knowledge of teacher network to the student by leveraging the KD-aware loss, $\mathcal{L}_{total}$, for 30 epochs with learning rate of 1e-2 and fine-tune the student network for 60 epochs with learning rate of 5e-2.

E EXPERIMENTAL SETUP

E.1 TIME MEASUREMENT FOR ADAPTATION AND NEURAL ARCHITECTURE SEARCH

In Section 5.2, we measure the time for collecting the few-shot student architecture-accuracy pair(s) and the time for conducting the single forward pass to get a teacher network-accuracy pair. In all cases, the time taken was measured on a single NVIDIA's RTX 2080 Ti GPU. As shown in Table 4, we also measure the time required to find an optimal student architecture in DaNAS settings. We randomly sample 10,000 student architectures and select the top-1 student architecture found to be most optimal by each method. For every method, after obtaining the total time for validation for 10,000 samples first, the average search time per network was calculated. As in the previous case, all the time required for searching was measured on a single NVIDIA's RTX 2080 Ti GPU.

E.2 CONDITIONS FOR STUDENT ARCHITECTURES IN THE SEARCH SPACE

In the Experiment 5.3, to provide an environment closer to real-world KD scenarios, when random sampling of student architectures in the search space, architectures exceeding 80% of FLOPs of given teacher network were filtered out. Additionally, in Experiment 5.4, since we leverage the proposed teacher-parameter-remapping-based embedding, we do search among the student architecture candidates which are capable of parameter remapping from a smaller-sized teacher network.