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

**Hayeon Lee**[*], **Sohyun An**[*], **Minseon Kim** **Sung Ju Hwang**
KAIST, South Korea
{hayeon926, sohyunan, minseonkim, sjhwang82}@kaist.ac.kr

## ABSTRACT

Distillation-aware Network Architecture Search (DaNAS) aims to search for an optimal student architecture that 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 performances 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. Code is available at https://github.com/CownowAn/DaSS.

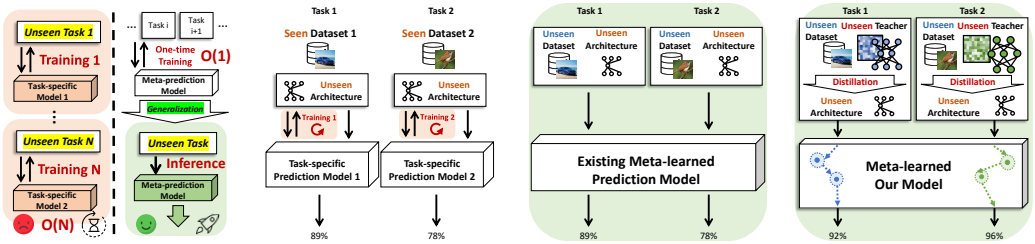
## 1 INTRODUCTION

Distillation-aware Neural Architecture Search (DaNAS) aims to search for an optimal student architecture that obtains the best performance and efficiency on a given dataset when distilling the knowledge from the given teacher to it (Liu et al., 2020; Gu & Tresp, 2020; Kim et al., 2022). For DaNAS task, we need to design a framework that considers the effect of Knowledge Distillation (KD), yet, conventional NAS frameworks may be sub-optimal as they do not consider KD components at all by searching for an architecture according to its evaluations trained from scratch. As explained in Liu et al. (2020), the sub-optimality of conventional NAS methods on DaNAS tasks results from: 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 only ground-truth labels may be different. 2) Even for the same dataset, the optimal student architecture may depend on the specific 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, 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 real-world scenarios since optimal student architectures may vary depending on the type of datasets, teacher, 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

[*]These authors contributed equally to this work.




(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.

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, allowing it to adapt to a novel unseen task by transferring the knowledge obtained over the meta-learning phase to the new task without training 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 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 fully training 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, not KD from a teacher, which may significantly impact the actual accuracy of the architecture 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 tasks (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.: 1) Distillation-aware design of the meta-prediction model and 2) a meta-learning scheme which utilizes already trained teacher, both of which are optimized for the DaNAS task. First, we propose a distillation-aware task encoding function that considers the output from the student whose parameters are remapped from the teacher to estimate the teacher's impact on the actual performance of the distilled student network. 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.

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 heterogeneous unseen datasets such as CUB, Stanford Cars, DTD, STL10, CropDisease, EuroSAT, ISIC, ChestX, and ImageNet-1K. The experimental results show that our meta-learned prediction model adapts to novel target tasks to estimate the actual performance of an architecture distilled by an unseen teacher within 3.45 (wall clock sec) on average without direct training on the target tasks. It 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 teachers, which can accurately predict the performance of an architecture when distilling the knowledge of the given teacher.
- 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 actual accuracies of the student architectures for a given teacher on multiple unseen datasets, significantly outperforming existing methods in the correlation between prediction and GT accuracies and in the end-to-end DaNAS performances.

## 2 Related Work

**Distillation-aware NAS** Knowledge Distillation (KD) is an efficient model compression technique under the resource budgets by transferring the knowledge from the large teacher to the smaller student network. Since distilling performance depends on architectures as verified in Liu et al. (2020), it is important to search for an optimal student architecture given the dataset and teacher. Recently, while Neural Architecture Search (NAS) suggests automating 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. Liu et al. (2020) verified the assumption with the RL agent, which KD-trains sampled architecture for a few epochs to obtain a KD-guided accuracy proxy. It takes 5 days with 200 TPUv2 on a single task, so it cannot be applied to real-world KD scenarios. Kim et al. (2022) suggested the trust region with a Bayesian optimization search algorithm and a new KD-guided score. However, it cannot be generalized to multiple tasks, which makes rapid search impossible. To obtain a pruned student architecture, (Gu & Tresp, 2020) specify scaling factors for all channels of a teacher and optimize that with KD-aware loss function applying a structured pruning method. As the optimization should be performed repeatedly for each task to get the optimal student architecture, the total time is still 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. 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 and distillation process, thereby an architecture obtained by them might be suboptimal when distilling knowledge from the teacher 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.

## 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 prediction 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

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** |

constant even if the number of tasks increases ($O(1)$). However, while ours is generalized for unseen teacher networks (Unseen 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 meta-prediction model on an unseen task in NAS (Fast Adapt.).

**Task-specific Accudracy Prediction Model** During the 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 $f^\tau(s; \boldsymbol{\phi}) : \mathcal{S} \rightarrow \mathbb{R}$ parameterized by $\boldsymbol{\phi}$ is trained for each task $\tau$ 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 $f^\tau(\mathbf{s}; \boldsymbol{\phi})$ and actual measurements $y^\tau$ as follows:

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

However, the task-specific accuracy prediction model is highly limited in its practical application to a 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 highly time-consuming since we need to collect many architecture-accuracy pairs (at least thousands of 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: $f(s, \mathbf{D}^\tau; \boldsymbol{\phi}) : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{R}$. 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 $\tau$ per iteration via amortized meta-learning as follows:

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

After this one-time meta-learning, the set-conditioned accuracy prediction model $f(\cdot; \boldsymbol{\phi}^*)$ with meta-learned parameters $\boldsymbol{\phi}^*$ 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 an unseen dataset $\hat{\mathbf{D}}^{\hat{\tau}}$ which has 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 the following reasons:

1. **Distillation-independent accuracy prediction.** DaNAS is different from the existing NAS problems in that a task $\tau$ in the distillation-aware NAS domain includes a teacher $\mathbf{t}(\cdot; \tilde{\boldsymbol{\theta}})^\tau$ 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 teachers are different. However, the existing meta-prediction model does not encode the teacher and thus will search for the same student architecture for different teachers, 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 aim to learn a generalized accuracy prediction for multiple tasks with amortized meta-learning conditioned on the task embeddings, not leveraging the power of gradient-based adaptation steps (Finn

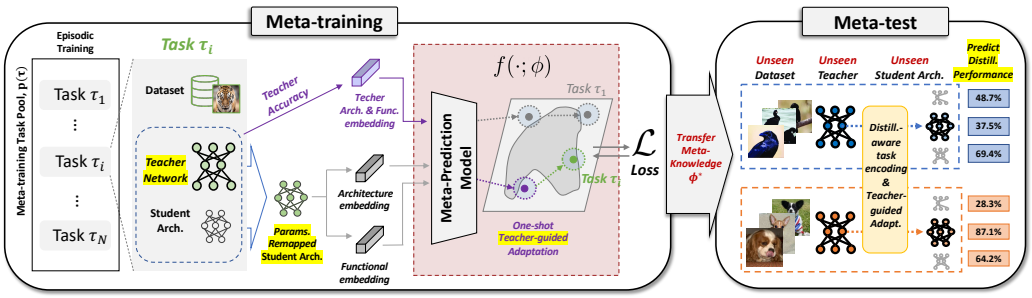



Figure 2: **Distillation-aware Meta-accuracy Prediction Model.** To search for an optimal student architecture that can distill the knowledge of given teacher $\mathbf{t}(\cdot;\tilde{\boldsymbol{\theta}})^{\tau}$ well, the sampled task should be encoded in a distillation-aware way. By taking a fixed input after remapping the trained parameters of the teacher to the randomly sampled student, we can encode the student functionally in a teacher-adaptive way. Along with the functional embedding, by considering the architecture configuration of the student, we encode sampled tasks in a way appropriate for the DaNAS. By transferring meta-knowledge $\phi^*$ learned from meta training phase, our meta-prediction model $f(\cdot;\phi^*)$ can adapt to an unseen task $\hat{\tau}$ quickly through the teacher-guided adaptation with almost no cost.

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 teacher-conditioned accuracy prediction is more accurate than set-conditioned accuracy prediction when estimating actual performances of student architectures in DaNAS scenarios. In addition, we show that it is possible to perform gradient-guided task adaptation using the accuracy of the teacher. Please refer to Figure 2, which illustrates our distillation-aware performance 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 meta-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 $\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau})$. We encode a task $\tau$ by utilizing the target dataset, teacher, and the student architecture in a way 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 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. More detailed descriptions about the parameter remapping function, $g$ is in Appendix Section B. Next, we embed each remapped student candidate $s(\cdot;\tilde{\boldsymbol{\theta}}_s^{\tau})$ 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 addition, as described in Appendix Section A, our ResNet search space is factorized hierarchical search space, and all architecture candidates in this search space are composed of 4 stages. Student architecture candidates in this search space can have various depths and channel widths at each stage. As suggested in Cai et al. (2020), the architecture configuration, such as depth and channel widths at each stage of $s$, can be represented as a one-hot encoding vector. Concatenating one-hot encoding 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 refer to the Appedix Section B.

**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:

$$\min_{\boldsymbol{\phi}} \sum_{\tau\sim p(\tau)} \mathcal{L}\big(f\big(\mathbf{S},\mathbf{D}^\tau,\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}\big),\mathbf{Y}^\tau\big) \tag{5}$$

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 $\boldsymbol{t}^\tau$ and its accuracy $\tilde{y}^\tau$ evaluated on the target dataset, by conducting the inner gradient updates with them as follows:

$$\boldsymbol{\phi}^\tau_{(i+1)} = \boldsymbol{\phi}^\tau_{(i)} - \alpha\nabla_{\boldsymbol{\phi}_{(i)}}\mathcal{L}(f(\mathbf{t}^\tau,\mathbf{D}^\tau,\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}_{(i)}),\tilde{y}^\tau) \quad \text{for } i=1,\dots,I \tag{6}$$

where $i$ denotes the $i$-th inner gradient step, $I$ is the total number of inner gradient steps, and $\alpha$ 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 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 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 (5) and (6) is given as follows:

$$\min_{\boldsymbol{\phi}} \sum_{\tau\sim p(\tau)} \mathcal{L}\big(f\big(\mathbf{S},\mathbf{D}^\tau,\mathbf{t}^\tau(\cdot;\tilde{\boldsymbol{\theta}}^\tau);\boldsymbol{\phi}^\tau_{(I+1)}\big),\mathbf{Y}^\tau\big) \tag{7}$$

**Adapting the Meta-prediction Model on Unseen Tasks (Meta-test)** In Figure 2, 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 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 (6) 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 meta-prediction model in terms of 1) a novel distillation-aware task encoding and 2) a gradient-based one-shot adaptation strategy with the teacher by 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 our method compared with few-shot meta-prediction models in Section 5.2. Next, we demonstrate the performance of our model on end-to-end DaNAS tasks compared against existing rapid NAS methods in Section 5.3. In addition to this, we compared the performance of ours and the existing Meta-NAS methods through end-to-end DaNAS tasks on Cross-domain datasets in Section 5.4. Lastly, we validated our method on ImageNet-1K which is one of the representative large-scale datasets with 256 x 256 pixels in Section 5.5. A detailed description of our search space throughout the experiments is in Appendix Section A, and we use the largest architecture in our search space as teacher architecture for all experiments.

**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. Please refer to Appendix Section D for more detailed explanation.

**Generalization to Unseen Datasets** We transfer the meta-knowledge of the prediction model trained over the meta-training phase to meta-test tasks. Each meta-test task consists of an unseen dataset, an unseen teacher trained on the unseen dataset, and multiple unseen student architecture

Table 2: **Efficacy of the Proposed Meta-prediction Model for DaNAS.** We report Spearman's rank correlation coefficient between ranking by actual distillation performance and ranking by predicted distillation performance of 50 unseen student architectures on an unseen dataset, with an unseen teacher. The reported values are the average over 3 random runs. Our meta-prediction model with teacher-guided meta-learning strategy outperforms all rapid meta-prediction models.

| Method | Model Components | | | | Spearman's Rank Correlation Coefficient | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Arch. | Set Enc. | Distill-aware. | Guide | CUB | Cars | DTD | Draw | Mean |
| MAML (Finn et al., 2017) | ✓ | | | ✓ | 0.63 | 0.61 | 0.38 | 0.42 | 0.51 |
| Meta-SGD (Li et al., 2017) | ✓ | | | ✓ | 0.18 | 0.49 | 0.19 | 0.63 | 0.37 |
| MetaD2A (Lee et al., 2021a) | ✓ | ✓ | | | 0.63 | 0.51 | 0.40 | 0.26 | 0.45 |
| TANS (Jeong et al., 2021) | ✓ | ✓ | | | 0.43 | 0.61 | -0.15 | -0.17 | 0.18 |
| Ours (Non-Guide) | ✓ | | ✓ | | 0.69 | 0.77 | 0.50 | 0.62 | 0.64 |
| **Ours (Guide)** | ✓ | | ✓ | ✓ | **0.69** | **0.88** | **0.57** | **0.75** | **0.72** |

candidates. We covered 9 datasets, including fine-grained datasets (CUB (Wah et al., 2011), Stanford Cars (Krause et al., 2013) (Cars), CropDisease (Guo et al., 2020) - agriculture images), out-of-distribution datasets (DTD (Cimpoi et al., 2014) - texture images, Quick Draw (Jonas et al.) (Draw) - black-and-white drawing images, EuroSAT (Guo et al., 2020) - satellite images, ISIC (Guo et al., 2020) - medical images about skin lesions, ChestX (Guo et al., 2020) - X-ray images), and large-scale dataset - ImageNet-1K (Russakovsky et al., 2015).

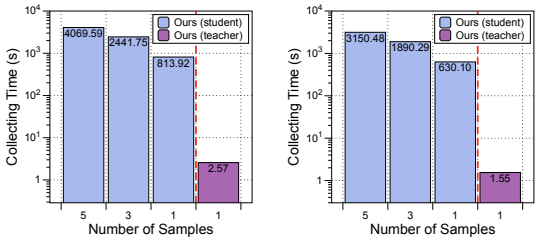
### 5.1 Efficacy of the Proposed Meta-prediction Model on Unseen Datasets

We validate whether our novel distillation-aware task encoding and meta-learning strategy, which is a gradient-based one-shot adaptation with the teacher, help improve the performance of the accuracy prediction model. We use fixed 50 student architectures and distill knowledge from the teacher on each unseen dataset to compute Spearman's rank correlation (the higher, the better) between the predicted and actual performances and report the efficacy of each method. In Table 2, **Arch.** means taking architecture configuration, **Set Enc.** means taking dataset embedding, **Distill-aware.** means taking the functional embeddings of student architecture candidate and teacher, and **Guide** means the adaptation strategy guided by teacher-accuracy pair in a meta-learning scheme.

Ours (Guide) outperforms all rapid meta-prediction models (Finn et al., 2017; Li et al., 2017; Lee et al., 2021a; Jeong et al., 2021) on 4 unseen datasets. MAML and Meta-SGD, which learn accuracy prediction models by taking only the architecture configuration (Arch.) of a student candidate as an input, show poor performance, as they predict the distillation performance without consideration of a target dataset and teacher. Set-conditioned (Set Enc.) prediction models (MetaD2A and TANS) also achieve significantly lower performance than ours since they search architectures in a teacher-agnostic manner, which greatly impacts the performance of the student trained with distillation. On the other hand, using the proposed distillation-aware embeddings of student and teacher (Distill-aware.), Ours (Guide) significantly outperforms all baselines on all 4 unseen tasks. Furthermore, we observe that adaptation to the target task by guiding our meta-prediction model with the accuracy of the teacher on the target dataset largely improves the prediction performance over the non-guided search, Ours (Non-Guide). In sum, the results show the clear advantage of both the design of our distillation-aware task encoding method 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 the 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 a trained model, the **teacher**. There is no need to wait to get the accuracy-




(a) CUB (b) DTD

Figure 3: **Time and Sample Efficiency.**

architecture pair(s) for adaptation since we can get the accuracy of the teacher with a single forward pass. In Figure 3, we report the time efficiency of our meta-prediction model, compared against that of the baselines, when adapting to unseen tasks with student architecture-accuracy pair(s). The **Collecting time** refers to the time required to obtain architecture-performance pairs (**Number of samples**) in the guided meta-learning strategy. Our one-shot adaptation guided search with the teacher network which is already trained under the DaNAS task is much more efficient than utilizing a few-shot adaptation with a few student architecture-accuracy pair(s). Our model only needs $2.57$(s) and $1.55$(s) on **CUB** and **DTD**, respectively, which significantly reduces the adaptation time.

### 5.3 Comparison with Rapid NAS Methods

Table 3: The reported results are **SRC** between ranking by actual distillation performance and ranking by predicted distillation performance of unseen student architectures on an unseen dataset, with an unseen teacher. **Search Time** is the time for predicting the performance of each unseen student architecture by given method.

| | Method | Spearman's Rank Correlation Coefficient (**SRC**) | | | | | **Search Time (GPU sec)** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CUB | Cars | DTD | Draw | Mean | CUB | Cars | DTD | Draw | Mean |
| Zero-cost proxies | Grad Norm (Abdelfattah et al., 2021) | -0.07 | 0.09 | 0.19 | -0.60 | -0.09 | 0.98 | 1.00 | 1.11 | 2.11 | 1.30 |
| | Snip (Abdelfattah et al., 2021) | 0.11 | 0.27 | 0.24 | -0.81 | -0.04 | 1.05 | 1.03 | 0.97 | 2.14 | 1.30 |
| | Grasp (Abdelfattah et al., 2021) | -0.15 | -0.06 | -0.19 | 0.28 | -0.03 | 1.37 | 1.34 | 1.34 | 2.56 | 1.65 |
| | Fisher (Abdelfattah et al., 2021) | 0.03 | 0.08 | 0.20 | -0.79 | -0.12 | 0.97 | 1.04 | 1.05 | 2.10 | 1.29 |
| | Plain (Abdelfattah et al., 2021) | 0.12 | 0.22 | 0.26 | -0.39 | 0.05 | 0.90 | 1.01 | 1.03 | 2.10 | 1.26 |
| | Synflow (Abdelfattah et al., 2021) | -0.10 | 0.13 | 0.17 | -0.26 | -0.01 | 0.85 | 0.98 | 0.97 | 2.16 | 1.24 |
| | NASWOT (Mellor et al., 2021) | -0.45 | 0.15 | 0.32 | -0.42 | -0.10 | 0.92 | 1.00 | 1.02 | 2.03 | 1.24 |
| Meta-learning | MetaD2A (Lee et al., 2021a) | 0.63 | 0.51 | 0.40 | 0.26 | 0.45 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| | TANS (Jeong et al., 2021) | 0.43 | 0.61 | -0.15 | -0.17 | 0.18 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| | **Ours** | **0.69** | **0.88** | **0.57** | **0.75** | **0.72** | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |

Table 4: **End-to-end DaNAS on Unseen Datasets** The distillation performance for the student architecture searched by each method was evaluated on the target validation set. All MACs values shown in the Table are based on $64$ x $64$ pixels and the reported accuracies are the mean and standard deviations over 3 runs.

| **Unseen Dataset** | | **CUB** | | **Stanford Cars** | | **DTD** | | **Quick Draw** | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) |
| | Teacher | 37.19 | 1459.51 | 43.25 | 1459.50 | 41.49 | 1456.47 | 44.14 | 1459.50 |
| Zero-cost proxies | Grad Norm | 42.23 $\pm$ 0.25 | 1029.07 | 48.07 $\pm$ 0.08 | 1010.42 | 42.80 $\pm$ 0.22 | 1235.98 | 45.30 $\pm$ 0.10 | 981.49 |
| | Snip | 39.55 $\pm$ 0.05 | 1175.61 | 47.83 $\pm$ 0.21 | 1150.48 | 43.19 $\pm$ 0.49 | 1235.98 | 45.24 $\pm$ 0.06 | 981.49 |
| | Grasp | 39.41 $\pm$ 0.11 | 711.06 | 48.22 $\pm$ 0.12 | 719.71 | 44.92 $\pm$ 0.68 | 711.06 | 45.31 $\pm$ 0.04 | 745.29 |
| | Fisher | 39.55 $\pm$ 0.05 | 1175.61 | 48.07 $\pm$ 0.08 | 1010.42 | 43.75 $\pm$ 0.34 | 1076.26 | 45.28 $\pm$ 0.11 | 981.49 |
| | Plain | 40.85 $\pm$ 0.45 | 702.63 | 51.75 $\pm$ 0.30 | 903.76 | 42.98 $\pm$ 0.68 | 1137.60 | 45.24 $\pm$ 0.10 | 981.49 |
| | Synflow | 39.97 $\pm$ 0.28 | 1251.26 | 46.75 $\pm$ 0.25 | 1251.26 | 43.57 $\pm$ 0.22 | 1251.26 | 45.71 $\pm$ 0.06 | 1251.26 |
| | NASWOT | 40.94 $\pm$ 0.07 | 1030.65 | 52.03 $\pm$ 0.20 | 746.44 | 45.47 $\pm$ 0.51 | 874.63 | 45.48 $\pm$ 0.09 | 702.72 |
| Meta-learning | MetaD2A | 41.64 $\pm$ 0.04 | 708.75 | 56.74 $\pm$ 0.36 | 734.97 | 45.13 $\pm$ 0.26 | 912.10 | 46.80 $\pm$ 0.09 | 761.73 |
| | TANS | 41.74 $\pm$ 0.34 | 743.99 | 56.00 $\pm$ 0.12 | 995.44 | 45.26 $\pm$ 0.78 | 836.21 | 45.61 $\pm$ 0.06 | 702.60 |
| | **Ours** | **43.48** $\pm$ **0.11** | 792.97 | **59.09** $\pm$ **0.06** | 789.13 | **47.86** $\pm$ **0.40** | 789.13 | **47.27** $\pm$ **0.10** | 714.13 |

In Table 3, we validate our meta-prediction model with other rapid performance prediction methods. Zero-cost proxies, which are representative rapid performance prediction methods, calculate the scores of architectures by feeding a few batches of the target dataset into randomly initialized architectures. Therefore, they are efficient but cannot consistently predict the distillation performance across all unseen datasets. As shown in Table 3, ours outperforms all rapid NAS baseline models in terms of SRC, including these $6$ zero-cost proxies (Abdelfattah et al., 2021; Mellor et al., 2021) and $2$ meta-accuracy prediction models (Lee et al., 2021a; Jeong et al., 2021) on $4$ unseen datasets. Additionally, the average search time of ours is only $0.05$ GPU seconds, which is competitive with other rapid NAS methods. We further validate the proposed meta-prediction model by conducting end-to-end DaNAS tasks on $4$ unseen datasets. For a fair comparison, we randomly sample $3,000$ student architectures and select the Top-1 student architecture whose estimated performance by each method is the highest, then distill the knowledge from the same teacher trained on the target dataset to the selected student architecture with the same number of epochs. The results in Table 4 show that the student architecture found by ours distills the knowledge from the teacher network better than the student architecture found by other baselines on $4$ unseen datasets.

### 5.4 Comparison with Meta-accuracy Prediction Models on Unseen Datasets across Domains

We further conducted an evaluation of our method on Broader Study of Cross-Domain Few-Shot Learning (BSCD-FSL) benchmark datasets (Guo et al., 2020). The benchmark datasets are from different domains compared with the datasets that we have used to learn our meta-prediction model

Table 5: **End-to-end DaNAS on Unseen Datasets across Domains** We compared ours to the meta-accuracy prediction model baselines on BSCD-FSL benchmark containing datasets across domains. All MACs values are based on 64 x 64 pixels and the reported accuracies are the mean and standard deviations over 3 runs.

| **Unseen Dataset** | **CropDisease** | | **EuroSAT** | | **ISIC** | | **ChestX** | |
|---|---|---|---|---|---|---|---|---|
| | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) | Acc. (%) | MACs (M) |
| Teacher | 99.66 | 1464.66 | 97.41 | 1464.65 | 79.27 | 1464.65 | 40.32 | 1464.65 |
| MetaD2A | 99.68 $\pm$ 0.07 | 374.10 | 97.21 $\pm$ 0.29 | 398.06 | 79.27 $\pm$ 1.56 | 374.10 | 39.95 $\pm$ 1.38 | 393.50 |
| TANS | **99.72** $\pm$ **0.02** | 398.90 | 97.33 $\pm$ 0.48 | 384.57 | 79.79 $\pm$ 0.89 | 387.32 | 39.47 $\pm$ 1.40 | 385.99 |
| **Ours** | 99.71 $\pm$ 0.01 | 397.84 | **97.95** $\pm$ 0.06 | 378.99 | **81.52** $\pm$ **1.08** | 374.82 | **42.25** $\pm$ **0.94** | 371.47 |

during meta-training phase. As shown in Table 5, for the EuroSAT, ISIC, and ChestX of the BSCD-FSL benchmark except the most easy dataset, CropDisease, the student architectures obtained by ours are better at KD than other meta-prediction model baselines. The results show that the meta-knowledge of our prediction model with distillation-aware task encoding trained over the meta-training tasks with our adaptation strategy with teacher is effective in cross-domain datasets as well as in the same-domain datasets.

### 5.5 Validation on a Large-scale dataset with Different Pixels and Models

Table 6: MACs values in this Table is based on 256 x 256 pixels and the architecture configuration is based on Cai et al. (2020).

| | **ImageNet-1K (256 $\times$ 256)** | | | |
|---|---|---|---|---|
| | Acc. (%) | MACs (M) | Depth Config. | Width Config. |
| Teacher | 76.73 | 9843.17 | [4, 4, 6, 4] | [256, 512, 1024, 2048] |
| TANS | 78.27 | 4211.43 | [2, 3, 4, 2] | [208, 512, 1024, 2048] |
| MetaD2A | 78.43 | 5411.69 | [4, 3, 5, 4] | [208, 512, 816, 2048] |
| **Ours** | **79.00** | 4029.64 | [2, 3, 5, 2] | [256, 512, 664, 2048] |

We validate our method with the larger pixels (256 x 256) on ImageNet-1K, a representative large-scale real-world dataset, by conducting end-to-end DaNAS with the pre-trained teacher, which is the largest architecture in a ResNet search space. The search space in this Section is defined by Cai et al. (2020), and it is based on more depth and channel widths compared to previous experiments. Using our method (Ours) and other meta-accuracy prediction model baselines (TANS and MetaD2A), we select an architecture with a Top-1 score in each method. Then we distill knowledge from the same pre-trained teacher to the obtained student architecture in the same training setting and report the best performance evaluated on the validation set. As shown in Table 6, ours successfully searched for more accurate and efficient student architecture, which shows 79.00% accuracy, outperforming the student architectures retrieved by the baseline methods. In sum, the results show that our method is not limited to specific models and works well with larger pixels, demonstrating its practicality. More results of experiments with larger pixels are in Appendix Section F, and more description about the experiment setting is in Appendix Section E.

## 6 Conclusion

We proposed a novel meta-prediction model that generalizes across datasets, architectures, and teacher networks on DaNAS tasks, which can accurately predict the performance of an architecture when distilling the knowledge of the given 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 sub-optimal in that they do not consider the KD aspects. In order to devise a meta-prediction model for solving DaNAS tasks, we introduce a novel distillation-aware task encoding based on the embedding of the teacher and a parameter remapping scheme. We also leverage a rapid gradient-based one-shot adaptation of the meta-prediction model on a target task by guiding it with a teacher-accuracy pair, unlike existing meta-accuracy prediction methods, which do not take inner-gradient steps for task adaptation. 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 on the heterogeneous unseen DaNAS tasks.

**Acknowledgements** This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, No.2022-0-00713). This material is based upon work supported by Google Research Grant and the Google Cloud Research Credits program with the award (PMF4-4RK0-M9N1-YB37).

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

**Reproducibility Statement** First, we attached the code https://github.com/CownowAn/DaSS of the proposed method to help with reproducibility. 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 provide the details of the *search space* that we used in the all experiments in the main document.
- **Section B** - 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 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.
- **Section F, and Section G** - We provide additional *experimental results*.

## 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.

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.

Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. ECCV, 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.

Jongejan Jonas, Rowley Henry, Kawashima Takashi, Kim Jongmin, and Fox-Gieg Nick. The quick, draw! – a.i. experiment. URL https://quickdraw.withgoogle.com/.

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.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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 SEARCH SPACE

Our ResNet (He et al., 2016) search space is factorized hierarchical search space; thus, architecture candidates in this search space can have variable depth and channel widths in each stage, which means a sub-module that divides the entire network into smaller units. More specifically, our ResNet search space in Section 5 consists of 4 stages. 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 number of channel widths for each layer. To further explain the architecture configuration notation in Section 5, **Detph Config.** means how many blocks exist in each stage respectively and **Width Config.** means the number of channels in the ResNet block convolutional layers at each stage that are obtained by multiplying a channel width shrink ratio by base channel widths predefined for each stage. In Section 5 except for Section 5.5, Depth Config. can be chosen from $\{1, 2, 3, 4\}$ for each stage, the base channel width is $[32, 64, 128, 256]$, and the channel shrink ratio for each layer can be chosen from $\{0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1.0\}$. In the case of Section 5.5, Depth Config. can be chosen from $\{0, 1, 2\}$ and it indicates how much more depth is added to the existing base depth at each stage. Additionally, the base channel width is $[256, 512, 1024, 2048]$, and the channel shrink ratio for each layer can be chosen from $\{0.65, 0.8, 1.0\}$. For all architectures 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.

# B IMPLMENTATION DETAILS

## B.1 PARAMETER REMAPPING

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 in our search space as:

$$\mathbf{t}^{\tau}(\cdot;\tilde{\boldsymbol{\theta}}^{\tau}) = \tilde{\mathcal{B}}_M \circ \cdots \circ \tilde{\mathcal{B}}_1(\cdot;\tilde{\boldsymbol{\theta}}_1^{\tau}), \quad \mathbf{s}^{\tau}(\cdot;\theta^{\tau}) = \mathcal{B}_M \circ \cdots \circ \mathcal{B}_1(\cdot;\theta_1^{\tau}) \tag{8}$$

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. When performing distillation-aware task encoding, we encode each sampled student architecture in a teacher-adaptive manner with a 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, $\tilde{\boldsymbol{\theta}}_s^{\tau} = g(\theta^{\tau};\tilde{\boldsymbol{\theta}}^{\tau})$, is as follows:

$$\tilde{\boldsymbol{\theta}}_{s,m,o}^{\tau} = \tilde{\boldsymbol{\theta}}_{m,o}^{\tau} \quad \text{for } \ o = 1, \ldots, d. \tag{9}$$

$$\tilde{\boldsymbol{\theta}}_{s,m,o,k}^{\tau} = \tilde{\boldsymbol{\theta}}_{m,o,k}^{\tau} \quad \text{for } \ k = 1, \ldots, k_o. \tag{10}$$

## B.2 FUNCTIONAL EMBEDDING

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 $p(\tau)$, as decribed in Appendix Section A. 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.

### B.3 Architecture Embedding

Our prediction model takes an architecture configuration of sampled student architecture which contains information such as depth and channel widths. As our 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 $\{0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1.0\}$ except for Section 5.5. 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, $p(\boldsymbol{a})$. For the case of Section 5.5, we obtain the architecture embedding in a similar way.

### B.4 Distillation-aware Task Encoding

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, $\sigma$, which outputs distillation-aware network embedding with 64 dimensions.

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

### D.1 Meta-training Database

To learn the proposed meta-prediction model, we collect multiple DaNAS tasks. Each task consists of a dataset and a teacher trained on the dataset, a set of student architectures, and actual accuracies of

student architectures after distilling knowledge from the teacher network. As described in Section 5, we use TinyImageNet (Le & Yang, 2015) as the source dataset for meta-training and 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. We train a teacher on each sub-dataset for 180 epochs. Then 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 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).

### D.2 Meta-training the proposed prediction model

We learn our meta-prediction model over task distribution via episodic training (Vinyals et al., 2016). For each episode, we sample a meta-batch containing multiple tasks for each iteration. More specifically, the number of meta-batch is 8; in other words, we randomly sample 8 tasks from the meta-training database for each iteration. Each task consists of a specific teacher network trained on a specific dataset, the accuracy of the teacher network measured on the dataset, 50 student architecture candidates, and their actual accuracy obtained by distilling the knowledge from the teacher network to each student architecture.

For each given task, the meta-prediction model 1) Takes a student architecture and teacher network, 2) Encodes the task in a proposed distillation-aware way; we leverage the embedding of the teacher, architecture embedding of a student, and functional embedding of the student whose parameters are remapped from the teacher, and 3) Predict the distillation performance of the given student architecture. By minimizing the predicted and actual accuracy of the student, we learn the proposed meta-prediction model.

The total number of episodes is 500, and we used Adam optimizer with the learning rate 1e-4. The time to meta-train our meta-prediction model, which takes only once, is 2 hours with a single NVIDIA's RTX 2080 Ti GPU. As mentioned in Equation (6), 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 steps is set to 1. The initial value of each element in the multi-dimensional inner learning rate vector for our meta-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 adopt a model with the lowest meta-validation loss on the meta-validation dataset as our meta-prediction model. After fixing our meta-prediction model, a meta-test phase is conducted. As in the meta-learning phase, the one-shot adaptation strategy with the teacher network-accuracy pair was used in the meta-test phase.

### D.3 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:

$$\mathcal{L}_{total}(\mathbf{s}^{\tau}, \mathbf{t}^{\tau}; \mathbf{D}^{\tau}, y) = \beta \cdot \mathcal{L}_{KD,\alpha}(\mathbf{s}^{\tau}, \mathbf{t}^{\tau}; \mathbf{D}^{\tau}, y) + (1-\beta) \cdot \mathcal{L}_{IR}(\mathbf{s}^{\tau}, \mathbf{t}^{\tau}; \mathbf{D}^{\tau}) \tag{11}$$

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:

$$p^T_{t^\tau,i} = \frac{\exp(z_{t^\tau,i}/T)}{\sum_j \exp(z_{t^\tau,j}/T)}, \quad p^T_{s^\tau,i} = \frac{\exp(z_{s^\tau,i}/T)}{\sum_j \exp(z_{s^\tau,j}/T)} \tag{12}$$

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:

$$\mathcal{L}_{KD,\alpha}(p^T_{s^\tau,i}, p^T_{t^\tau,i}; \mathbf{D}^\tau, y) = \alpha \cdot H(y, p^T_{s^\tau,i}) + (1-\alpha) \cdot H(p^T_{s^\tau,i}, p^T_{t^\tau,i}) \tag{13}$$

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:

$$\mathcal{L}_{IR}(x;\, \mathbf{s}^\tau, \mathbf{t}^\tau, \mathbf{D}^\tau) = L_2(\mathcal{B}_1(x),\, \tilde{\mathcal{B}}_1(x)) + \sum_{m=2}^{M} L_2(\mathcal{B}_m(\tilde{\mathcal{B}}_{m-1} \circ \cdots \circ \tilde{\mathcal{B}}_1(x)),\, \tilde{\mathcal{B}}_m \circ \cdots \circ \tilde{\mathcal{B}}_1(x)) \tag{14}$$

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 Experimental Setup in Section 5.3

KD is one of representative model compression techniques to obtain small network to be used under resource limited environment or real-time applications. Considering the real-world KD scenarios, it is natural to assume that there is a resource budget in a given environment, and the optimal student architecture must be found within it. In Section 5.3, the teacher network is a network with 7.46M parameters and MACs constraint on the DaNAS task is as follows: MACs < 1300M, when the MACs of the teacher network is about 1450M. All the experimental values shown in this Section are based on $64 \times 64$ pixels.

## E.2 Experimental Setup in Section 5.5

We validate our method with the larger pixels ($256 \times 256$) on ImageNet-1K, a representative large-scale real-world dataset, by conducting end-to-end DaNAS with the largest ResNet defined by Cai et al. (2020) as the teacher in Section 5.5. We first randomly sample $1,000$ architecture candidates; the size of them in terms of MACs is $0.9\times$ smaller than that of the teacher, from the ResNet search space defined by Cai et al. (2020). We predict the performance of the $1,000$ architectures and select the architecture with Top-1 predicted performances computed by each method. Then we distill knowledge from the same pre-trained ResNet50 teacher to the obtained student architecture with the same 50 epochs and $256 \times 256$ pixels and report the best performance evaluated on the validation set.

## E.3 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 3, we also measure the time required to find an optimal student architecture in DaNAS settings. We randomly sample $3,000$ 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 scoring for $3,000$ samples first, the average search time per architecture was calculated. As in the previous case, all the time required for searching was measured on a single NVIDIA's RTX 2080 Ti GPU.

# F Validation on Large-Size Pixels

To demonstrate the practical value of the proposed approach, we conducted an experiment on a large-scale real-world dataset, which is ImageNet-1K, with large-size pixels of $256 \times 256$ in Senction 5.5. Furthermore, we conducted an experiment similarly on ChestX which is the hardest cross-domain dataset with large-size pixels of $224 \times 224$ in Table 7. As shown in Table 6 and 7, our meta-prediction model is robust to various image sizes, increasing practicality. We observed that the proposed method successfully searched for more accurate and efficient student architecture, which shows 79.00% and 44.34% distillation performances on ImageNet-1K and ChestX, respectively, outperforming other meta-accuracy prediction baseline methods.

Table 7: **Validation on large-size pixels** MACs values in this Table is based on 224 x 224 pixels and the reported accracies are the mean and standard deviations over 3 runs.

| | **ChestX (224 x 224)** | | | |
|---|---|---|---|---|
| | Acc. (%) | MACs (M) | Depth Config. | Width Config. |
| Teacher | 42.00 | 17878.31 | [5, 5, 5, 5] | [32, 64, 128, 256] |
| MetaD2A | $41.13 \pm 1.05$ | 4795.47 | [2, 4, 2, 2] | [20, 32, 104, 128] |
| TANS | $41.79 \pm 0.68$ | 4707.01 | [1, 2, 5, 1] | [32, 44, 72, 256] |
| **Ours** | $\mathbf{44.86 \pm 0.58}$ | 4532.10 | [1, 1, 2, 5] | [32, 64, 96, 128] |

## G 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 8, 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 8: **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 |