Learning by Teaching, with Application to Neural Architecture Search

Parth Sheth PARTHFOUR@GMAIL.COM
Yueyu Jiang Y5JIANG@ENG.UCSD.EDU
Pengtao Xie* P1XIE@ENG.UCSD.EDU
University of California San Diego

Abstract

In human learning, an effective skill in improving learning outcomes is learning by teaching: a learner deepens his/her understanding of a topic by teaching this topic to others. In this paper, we aim to borrow this teaching-driven learning methodology from humans and leverage it to train more performant machine learning models, by proposing a novel ML framework referred to as learning by teaching (LBT). In the LBT framework, a teacher model improves itself by teaching a student model to learn well. Specifically, the teacher creates a pseudo-labeled dataset and uses it to train a student model. Based on how the student performs on a validation dataset, the teacher re-learns its model and re-teaches the student until the student achieves great validation performance. Our framework is based on three-level optimization which contains three stages: teacher learns; teacher teaches student; teacher re-learns based on how well the student performs. A simple but efficient algorithm is developed to solve the three-level optimization problem. We apply LBT to search neural architectures on CIFAR-10, CIFAR-100, and ImageNet. The efficacy of our method is demonstrated in various experiments.

1. Introduction

As the saying goes, a good learner is also a good teacher. In human learning, a commonly adopted strategy is to learn by teaching others. In the process of explaining a topic to others, the learner can further enhance his/her understanding of this topic. The efficacy of teaching in helping improve learning has been demonstrated in many studies. In (Fiorella and Mayer, 2013), the studies showed that students who teach what they have learned to their peers achieve better understanding and knowledge retention than students spending the same time re-studying. In (Koh et al., 2018), the study shows that teaching improves the teacher's learning because it encourages the teacher to retrieve what they have previously learned.

This teaching-driven learning methodology of humans motivates us to think about whether it can benefit machine learning as well. Toward this goal, we propose a novel ML framework called learning by teaching (LBT) (as illustrated in Figure 1), which improves the learning outcome of a model by encouraging this model to teach other models to perform well. In our framework, there is a teacher model and a student model, which

---

. *Corresponding author.

© Y. Jiang & P. Xie*.

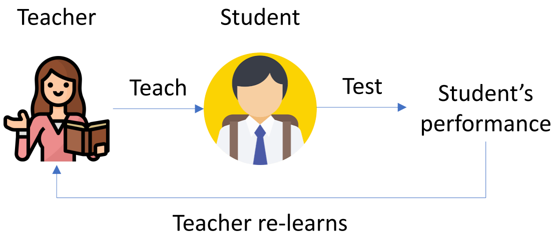

Figure 1: Illustration of learning by teaching. The teacher first learns a topic. Then the teacher teaches this topic to the student and the student learns this topic. The student performs a test to check how well he/she masters this topic. Based on the student's performance on the test, the teacher re-learns this topic to improve his/her understanding.

perform the same target task (e.g., text classification, time-series forecasting, etc.). The eventual goal is to make the teacher perform well on the target task. The way to achieve that is to let the teacher teach the student and use the student's performance as a feedback to guide the teacher to improve its learning capability and outcome. The teacher model consists of a learnable architecture and a set of learnable network weights. The student model consists of a predefined architecture (by human experts) and a set of learnable network weights. Similar to (Hinton et al., 2015b), teaching is conducted via pseudo-labeling: given an unlabeled dataset U, the teacher uses its intermediately trained model to make predictions on the input data examples in U; then the student model is trained on these pseudo-labeled data examples. Teacher-student learning based on pseudo-labeling has been studied in many previous works (Papernot et al., 2016; Tarvainen and Valpola, 2017; Xie et al., 2020). Our work differs from previous ones in that we aim to improve the learning ability of the teacher by letting it teach a student while previous works focus on improving the learning ability of a student model by letting it be taught by a fixed teacher model. In other words, our work focuses on learning the teacher while previous works focus on learning the student.

In our framework, the learning of the teacher and student are organized into three stages. In the first stage, the teacher learns its network weights on a training dataset while temporarily fixing its architecture. In the second stage, the teacher performs pseudo-labeling on an unlabeled dataset and uses the pseudo-labeled dataset to train the student model. Specifically, the teacher applies its model trained in the first stage to make predictions on unlabeled data examples and the student model is trained to fit these predictions. In the third stage, the student's model is evaluated on a validation set and the teacher adjusts its architecture based on the student's validation performance. The three stages are organized into a three-level optimization framework and are performed end-to-end in a unified manner, where earlier stages influence later stages and vice versa. We apply LBT for neural architecture search. Experiments on CIFAR-100, CIFAR-10, and ImageNet (Deng et al., 2009) demonstrate the effectiveness of our method.

The major contributions of this paper include:

- Motivated by the teaching-driven learning methodology of humans, we develop a novel machine learning framework called learning by teaching (LBT). In our approach, a teacher creates a pseudo-labeled dataset and uses it to train a student model. Based on how the student performs on the validation dataset, the teacher re-learns its model and re-teaches the student until the student achieves great validation performance.
- To formulate LBT, we develop a three-level optimization framework. This framework consists of three learning stages: 1) teacher performs learning; 2) teacher teaches what it has learned to a student; 3) teacher re-learns based on the performance of the student.
- An efficient algorithm is developed to solve the three-level optimization problem.
- We apply LBT for neural architecture search on CIFAR-100, CIFAR-10, and ImageNet, where the results show that our method is very effective in searching highly-performing neural architectures.

2. Related Works

2.1. Neural Architecture Search

Neural architecture search (NAS) is the task of developing algorithms to automatically find out architectures that can yield high ML-performance. Existing NAS methods can be categorized into three groups. Methods in the first group (Zoph and Le, 2017; Pham et al., 2018; Zoph et al., 2018) are based on reinforcement learning, where an architecture generation policy is learned by maximizing ML performance on validation data. Methods in the second group (Cai et al., 2019; Liu et al., 2019; Xie et al., 2019) are gradient-based and differentiable. These methods adopt a network pruning strategy where an overparameterized network with many building blocks is pruned into the final architecture and the optimal pruning is achieved by minimizing the validation loss. Methods in the third group (Liu et al., 2018b; Real et al., 2019) are based on evolutionary algorithms where architectures are represented as a population. Highly-performing architectures are allowed to generate offspring while poorly-performing architectures are eliminated.

2.2. Teacher-Student Learning

Teacher-student learning has been investigated in knowledge distillation (Hinton et al., 2015a), adversarial robustness (Carlini and Wagner, 2017), self-supervised learning (Xie et al., 2020), etc. Most of these methods are based on pseudo-labeling. In these existing methods, the focus is to learn a student model with the help of a trained and fixed teacher model. In these works, the teacher model is not updated. In contrast, our method focuses on learning a teacher model, by letting it teach a student model. The teacher model constantly updates itself based on the teaching outcome. Teacher-student learning has been investigated in several neural architecture search works as well (Li et al., 2020; Trofimov et al., 2020; Gu and Tresp, 2020). In these works, when searching the architecture of a student model, pseudo-labels generated by a trained teacher model whose architecture is fixed are leveraged. Our work differs from these works in that we focus on searching the

Table 1: Notations in Learning by Teaching

Notation	Meaning
A	Architecture of the teacher
T	Network weights of the teacher
S	Network weights of the student
$D_t^{(\mathrm{tr})}$	Training data of the teacher
$D_t^{(\mathrm{val})}$	Validation data of the teacher
$D_s^{(\mathrm{tr})}$	Training data of the student
$D_s^{(\mathrm{val})}$	Validation data of the student
D_u	Unlabeled dataset

architecture of a teacher model by letting it teach a student model where the student's architecture is fixed, whereas the existing works focus on searching the architecture of a student model by letting it be taught by a teacher where the teacher's architecture is fixed. In a recent work (Pham et al., 2020) which was conducted independently of and in parallel to our work, the teacher model is updated based on student's performance. Our work differs from this one in that our work is based on a three-level optimization framework which searches teacher's architecture by minimizing student's validation loss and trains teacher's network weights before using teacher to generate pseudo-labels, whereas in (Pham et al., 2020) the framework is based on two-level optimization which has no architecture search and does not train the teacher before using it to perform pseudo-labeling. In (Such et al., 2019), a meta-learning method is developed to learn a deep generative model which generates synthetic labeled-data. A student model leverages the synthesized data to search its architecture. Our work differs from this method in that we focus on searching the teacher's architecture via three-level optimization while (Such et al., 2019) focuses on searching the student's architecture via meta-learning.

3. Methods

In this section, we propose a three-level optimization framework to formalize learning-by-teaching (LBT) (as shown in Figure 2) and develop an efficient optimization algorithm for solving the LBT problem.

3.1. Learning by Teaching

In our framework, there is a teacher model and a student model, which both study how to perform the same target task. Without loss of generality, we assume the target task is classification. The eventual goal is to make the teacher achieve better learning outcomes. The way to achieve this goal to let the teacher teach the student to perform well on the target task. The intuition behind LBT is that a teacher needs to learn a topic very well in order to teach this topic to a student clearly. Teaching is performed based on pseudo-labeling (Hinton et al., 2015a): the teacher uses its model to generate a pseudo-labeled dataset; the student is trained on the pseudo-labeled dataset. The teacher has a learnable neural architecture A and a set of learnable network weights T. The student has a predefined architecture (by

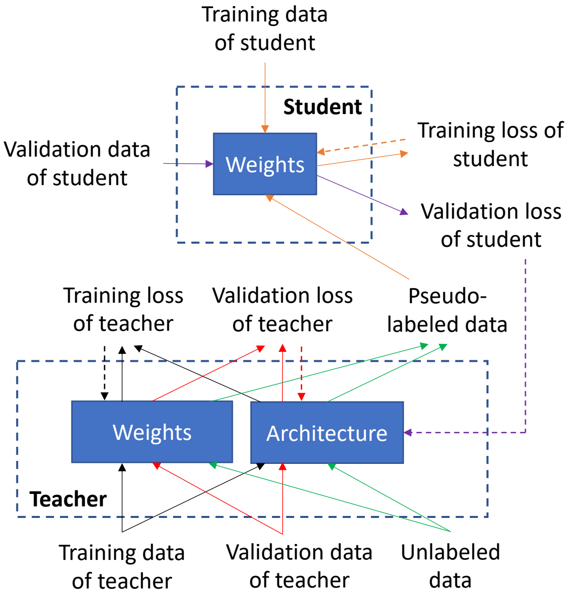

Figure 2: Learning by teaching. The solid arrows denote a forward pass where predictions are made and training/validation losses are defined. The dotted arrows denote a backward process where gradients of losses are calculated and parameters are updated.

humans) and a set of learnable network weights S. The teacher has a training dataset $D_t^{(\mathrm{tr})}$ and a validation dataset $D_t^{(\mathrm{val})}$. The student has a training dataset $D_s^{(\mathrm{tr})}$ and a validation dataset $D_s^{(\mathrm{val})}$. There is an unlabeled dataset D_u where pseudo labeling is performed. In our framework, both the teacher and student perform learning, which is organized into three stages. In the first stage, the teacher fixes its architecture and trains its network weights by minimizing the training loss defined on $D_t^{(\mathrm{tr})}$:

$$T^*(A) = \min_T L(T, A, D_t^{(\mathrm{tr})}). \tag{1}$$

The architecture A is needed to calculate the loss on training examples. However, it cannot be updated by minimizing the training loss. Otherwise, a degenerated solution will be produced where A has very large capacity to overfit the training examples but will yield poor prediction outcomes on unseen examples. $T^*(A)$ is a function of A: a different A will result in a different training loss $L(A, T, D_t^{(\mathrm{tr})})$; T trained by minimizing $L(A, T, D_t^{(\mathrm{tr})})$ will be different as well. In the second stage, the teacher teaches a student via pseudo-labeling. Given an unlabeled dataset $D_u = \{x_i\}_{i=1}^N$, the teacher uses its model $T^*(A)$ trained in the first stage to make predictions on D_u. Assuming the task is classification with K classes, the prediction $f(x_i; T^*(A))$ on x_i would be a K-dimensional vector, where the k-th element indicates the probability that x_i belongs to the k-th class and the sum of elements in $f(x_i; T^*(A))$ is one. Let $D_{pl}(D_u, T^*(A)) = \{(x_i, f(x_i; T^*(A)))\}_{i=1}^N$ denote the pseudo-labeled dataset. The network weights S of the student is trained on $D_{pl}(D_u, T^*(A))$

and a human-labeled training set $D_s^{(\mathrm{tr})}$:

$$S^*(T^*(A)) = \min\nolimits_S \ L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A))).$$

where $L(\cdot)$ denotes a cross-entropy loss and λ is a tradeoff parameter. $S^*(T^*(A))$ is a function of $T^*(A)$: a different $T^*(A)$ will result in a different pseudo-labeled dataset $D_{pl}(D_u, T^*(A))$ which will render the training loss to be different; a different training loss will result in a different $S^*(T^*(A))$. In the third stage, the student's model $S^*(T^*(A))$ trained in the second stage is validated on $D_s^{(\mathrm{val})}$. Besides, we also validate the teacher's model $T^*(A)$ trained in the first stage on $D_t^{(\mathrm{val})}$. The validation performances provide feedback on how good the teacher's architecture A is. At this stage, A is optimized by minimizing the validation losses:

$$\min\nolimits_A \ L(T^*(A), A, D_t^{(\mathrm{val})}) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})}), \tag{2}$$

where γ is a tradeoff parameter.

Given the three learning stages, we propose a three-level optimization framework to stitch them together:

$$\begin{array}{ll} \min\limits_A & L(T^*(A), A, D_t^{(\mathrm{val})})) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})}) \\ s.t. & S^*(T^*(A)) = \min\limits_S \ L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A))) \\ & T^*(A) = \min\limits_T L(A, T, D_t^{(\mathrm{tr})}) \end{array} \tag{3}$$

From bottom to top, the three optimization problems correspond to the first, second, and third stage respectively. The first two optimization problems are on the constraints of the third optimization problem. The three stages are performed end-to-end in a joint manner where different stages mutually influence each other. $T^*(A)$ trained in the first stage is used to generate pseudo-labeled dataset in the second stage; $T^*(A)$ and $S^*(T^*(A))$ trained in the first two stages are validated in the third stage; after A is updated in the third stage, it will render the training loss in the first stage to be changed, which accordingly results in a new $T^*(A)$. For computational efficiency, we search A in a differentiable way (Liu et al., 2019): given an overparameterized network, a subnetwork is carved out as the final architecture. The overparameterized network contains a large number of basic building blocks such as convolution operations, pooling operations, etc. The output of each building block is multiplied with a scalar. The search algorithm optimizes these scalars by minimizing validation losses. In the end, building blocks with largest scalars form the final architecture.

3.2. Optimization Algorithm

In this section, we develop a gradient-based algorithm to solve the learning by teaching (LBT) problem. Drawing insights from (Liu et al., 2019), we calculate the gradient of $L(A, T, D_t^{(\mathrm{tr})})$ w.r.t T, update T by one-step gradient descent and get T', which is used as an approximation of $T^*(A)$. We substitute this approximation into $L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A)))$, yielding an approximated objective O_s. Similarly, we approximate $S^*(T^*(A))$ using one-step gradient descent update of S using the gradient of O_s.

Lastly, we substitute the approximations of $T^*(A)$ and $S^*(T^*(A))$ into $L(T^*(A), D_t^{(\mathrm{val})}) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})})$ and perform gradient-descent update of A by minimizing the approximated validation loss. Let $\nabla^2_{Y,X} f(X, Y)$ denote $\frac{\partial f(X,Y)}{\partial X \partial Y}$.

First, we approximate $T^*(A)$ using

$$T' = T - \xi_t \nabla_T L(T, A, D_t^{(\mathrm{tr})}) \tag{4}$$

where ξ_t is a learning rate. Substituting T' into $L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A)))$ results in an approximated objective $O_s = L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T'))$. Next, we approximate $S^*(T^*(A))$ using one-step gradient descent update of S w.r.t O_s:

$$S' = S - \xi_s \nabla_S (L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T'))). \tag{5}$$

Finally, we plug T' and S' into $L(T^*(A), D_t^{(\mathrm{val})}) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})})$ and get $O_v = L(T', D_t^{(\mathrm{val})}) + \gamma L(S', D_s^{(\mathrm{val})})$. We can update the teacher's architecture A by descending the gradient of O_v w.r.t A:

$$A \leftarrow A - \eta(\nabla_A L(T', A, D_t^{(\mathrm{val})}) + \gamma \nabla_A L(S', D_s^{(\mathrm{val})})) \tag{6}$$

where

$$\begin{aligned}
&\nabla_A L(T', A, D_t^{(\mathrm{val})}) = \\
&\nabla_A L(T - \xi_t \nabla_T L(T, A, D_t^{(\mathrm{tr})}), A, D_t^{(\mathrm{val})}) = \\
&-\xi_t \nabla^2_{A,T} L(T, A, D_t^{(\mathrm{tr})}) \nabla_{T'} L(T', A, D_t^{(\mathrm{val})}) + \nabla_A L(T', A, D_t^{(\mathrm{val})})
\end{aligned} \tag{7}$$

The matrix-vector multiplication in the first term on the third line is computationally expensive. To reduce computational cost, following (Liu et al., 2019), we approximate the multiplication using a finite difference:

$$\nabla^2_{A,T} L(T, A, D_t^{(\mathrm{tr})}) \nabla_{T'} L(T', A, D_t^{(\mathrm{val})}) \approx \tfrac{1}{2\alpha}(\nabla_A L(T^+, A, D_t^{(\mathrm{tr})}) - \nabla_A L(T^-, A, D_t^{(\mathrm{tr})})), \tag{8}$$

where $T^{\pm} = T \pm \alpha \nabla_{T'} L(T', A, D_t^{(\mathrm{val})})$ and α is $0.01/\|\nabla_{T'} L(T', A, D_t^{(\mathrm{val})})\|_2$.

For $\nabla_A L(S', D_s^{(\mathrm{val})})$ in Eq.(6), it can be calculated as $\frac{\partial S'}{\partial A} \nabla_{S'} L(S', D_s^{(\mathrm{val})})$ according to the chain rule, where

$$\begin{aligned}
\frac{\partial S'}{\partial A} &= \frac{\partial(S - \xi_s \nabla_S (L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T'))))}{\partial A} && (9) \\
&= \frac{\partial(-\xi_s \lambda \nabla_S L(S, D_{pl}(D_u, T')))}{\partial A} && (10) \\
&= -\xi_s \lambda \frac{\partial T'}{\partial A} \nabla^2_{T',S} L(S, D_{pl}(D_u, T')) && (11)
\end{aligned}$$

For $\frac{\partial T'}{\partial A}$, it can be calculated as:

$$\frac{\partial T'}{\partial A} = \frac{\partial(T - \xi_t \nabla_T L(T, A, D_t^{(\mathrm{tr})}))}{\partial A} = -\xi_t \nabla^2_{A,T} L(T, A, D_t^{(\mathrm{tr})}) \tag{12}$$

The algorithm for solving LBT is presented in Algorithm 1.

Algorithm 1 Optimization algorithm for learning by teaching

```
while not converged do
    1. Update the teacher's network weights $T$ using Eq.(4)
    2. Update the student's network weights $S$ using Eq.(5)
    3. Update the teacher's architecture $A$ using Eq.(6)
end
```

4. Experiments

In this section, we apply learning-by-teaching (LBT) to search neural architectures in image classification tasks. We follow the experimental protocol in (Liu et al., 2019), consisting of two phrases: one for architecture search and the other for architecture evaluation. In the search phrase, an optimal cell is searched by minimizing the validation loss. In the evaluation phrase, the searched cell is replicated and composed into a large network, which is trained from scratch on training and validation sets. Its performance is reported on the test set. More hyperparameter settings, additional results, and significance tests of results are deferred to the supplements.

4.1. Datasets

The experiments were conducted on three image classification datasets: CIFAR-10, CIFAR-100, and ImageNet (Deng et al., 2009), with 10, 100, and 1000 classes respectively. For CIFAR-10 and CIFAR-100, each of them is split into a 25K training set, a 25K validation set, and a 5K test set. The training set is used as $D_t^{(\mathrm{tr})}$ of the teacher and $D_s^{(\mathrm{tr})}$ of the student. The validation set is used as $D_t^{(\mathrm{val})}$ of the teacher and $D_s^{(\mathrm{val})}$ of the student. For experiments on CIFAR-10, input images in CIFAR-100 (removing labels) are used as the unlabeled dataset D_u. For experiments on CIFAR-100, input images in CIFAR-10 (removing labels) are used as the unlabeled dataset D_u. For ImageNet, it is split into a training set of 1.2M images and a test set of 50K images.

4.2. Experimental Settings

In LBT, for the search space of A, we experimented the spaces in DARTS (Liu et al., 2019), P-DARTS (Chen et al., 2019), and PC-DARTS (Xu et al., 2020). These search spaces are similar, with the following candidate operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. For the student's architecture, we experimented with ResNet-18 and ResNet-50 (He et al., 2016b). λ and γ are both set to 1.

During architecture search, for CIFAR-10 and CIFAR-100, the teacher's architecture is a stack of 8 cells, each consisting of 7 nodes. The initial channel number was set to 16. The rest hyperparameters for the teacher's architecture and network weights follow those in DARTS, P-DARTS, and PC-DARTS. The search algorithm ran for 50 epochs, with a batch size of 64. Network weights are optimized using SGD, with an initial learning rate of 0.025 (adjusted using a cosine decay scheduler), a momentum of 0.9, and a weight decay of 3e-4. For architecture search on ImageNet, following (Xu et al., 2020), we randomly sample 10%

of the 1.2M ImageNet images as $D_t^{(\mathrm{tr})}$ and $D_s^{(\mathrm{tr})}$ in LBT, randomly sample 2.5% of the 1.2M images as $D_t^{(\mathrm{val})}$ and $D_s^{(\mathrm{val})}$, and randomly sample another 10% of the 1.2M images as D_u.

During architecture evaluation, for CIFAR-10 and CIFAR-100, 20 copies of the optimal cell searched in the search phrase are stacked into a large network, which is trained using the combined training and validation datasets. The initial channel number was set to 36. The network was trained for 600 epochs, with mini-batch size set to 96. These experiments were conducted on a Tesla v100 GPU. For ImageNet, we evaluate the architectures searched by PC-DARTS on the subset of ImageNet and architectures searched by DARTS-2nd and P-DARTS on CIFAR-10 and CIFAR-100, by stacking 14 searched cells into a large network and training it on the 1.2M training images and reporting its performance on the 50K test images. The initial channel number was set to 48. The network was trained for 250 epochs with a batch size of 1024 on 8 Tesla v100s. Each LBT experiment was repeated for ten times with different random seeds. Mean and standard deviation of the 10 runs are reported.

4.3. Results

In Table 2, we compare different NAS methods in terms of classification error on the test set of CIFAR-100, number of network weights, and search cost measured using GPU days. From these results, we observe the following. **First**, with the help of our proposed learning-by-teaching (LBT) framework, the architectures searched by various methods including DARTS, P-DARTS, and PC-DARTS can be greatly improved. For example, applying LBT to DARTS-2nd, the error is reduced from 20.58% to 17.06%. With the assistance of LBT, the error of P-DARTS decreases from 17.49% to 16.29%. Without using LBT, the error of PC-DARTS is 17.96%; adding LBT reduces this error to 16.88%. These results strongly demonstrate that LBT is an effective learning framework that helps to improve a wide variety of NAS methods. In our method, the teacher model improves its learning ability by teaching a student model to perform well on the classification task. The student is trained on the pseudo-labeled dataset created by the teacher. If the student does not perform well on the validation set, that means the pseudo labels are not correct, which indicates the teacher's model is not accurate. To avoid such an outcome, the teacher enforces itself to learn better to generate correct pseudo labels. **Second**, a stronger student helps the teacher to learn better. Here we consider a student model is stronger if its architecture (manually designed) is more powerful and expressive. For example, ResNet with 50 layers (RN50) is generally considered to have stronger representation learning power than ResNet with 18 layers (RN18). In LBT applied to DARTS, P-DARTS, and PC-DARTS, we experimented with two student models with RN50 and RN18 as their architectures respectively. As can be seen, LBT with RN50 as the student achieves better performance than LBT with RN18 as the student. For example, on DARTS-2nd, LBT with RN50 achieves an error of 17.06% while LBT with RN18 achieves an error of 17.93%. As another example, on P-DARTS, the error achieved by LBT(RN50) is 16.29%, which is lower than the 16.53% error achieved by LBT(RN18). The reason is that to teach a stronger student to learn better, the teacher has to be even stronger. For example, given a relatively weak student such as a CNN with only three layers, since there is a large room for the student to improve, an ordinary teacher such as a CNN with ten layers would be sufficient to teach the student to perform better. However, if the student (e.g., ResNet with 50 layers) is very strong whose performance is

Table 2: Results on CIFAR-100, including classification error (%) on the test set, number of model weights (millions), and search cost (GPU days). LBT(RN18,DARTS-1st) represents that in LBT the search space is the same as that of DARTS-1st and the architecture of the student is ResNet-18. Similar meanings hold for other notations in such a format. RN50 denotes ResNet-50. DARTS-1st and DARTS-2nd indicates that first-order and second-order approximation is used in DARTS' optimization algorithm. * denotes that the results are taken from DARTS$^-$ (Chu et al., 2020a). † denotes that we re-ran this method for 10 times. The search cost is measured by GPU days on a Tesla v100.

Method	Error(%)	Param(M)	Cost
*ResNet (He et al., 2016a)	22.10	1.7	-
*DenseNet (Huang et al., 2017)	17.18	25.6	-
*PNAS (Liu et al., 2018a)	19.53	3.2	150
*ENAS (Pham et al., 2018)	19.43	4.6	0.5
*AmoebaNet (Real et al., 2019)	18.93	3.1	3150
*GDAS (Dong and Yang, 2019)	18.38	3.4	0.2
*R-DARTS (Zela et al., 2020)	18.01±0.26	-	1.6
*DARTS$^-$ (Chu et al., 2020a)	17.51±0.25	3.3	0.4
*DropNAS (Hong et al., 2020)	16.39	4.4	0.7
†DARTS-1st (Liu et al., 2019)	20.52±0.31	1.8	0.4
LBT(RN18,DARTS-1st) (ours)	19.28±0.13	1.9	0.5
LBT(RN50,DARTS-1st) (ours)	**18.74**±0.09	1.9	0.6
*DARTS-2nd (Liu et al., 2019)	20.58±0.44	1.8	1.5
LBT(RN18,DARTS-2nd) (ours)	17.93±0.18	2.0	1.9
LBT(RN50,DARTS-2nd) (ours)	**17.06**±0.22	2.1	2.2
*P-DARTS (Chen et al., 2019)	17.49	3.6	0.3
LBT(RN18,P-DARTS) (ours)	16.53±0.16	3.6	0.5
LBT(RN50,P-DARTS) (ours)	**16.29**±0.07	3.7	0.6
†PC-DARTS (Xu et al., 2020)	17.96±0.15	3.9	0.1
LBT(RN18,PC-DARTS) (ours)	17.21±0.13	3.8	0.1
LBT(RN50,PC-DARTS) (ours)	**16.88**±0.09	3.8	0.2

already high, it is very challenging to teach the student to improve unless the teacher is even stronger. To better train a strong student, the pseudo-labels generated by the teacher are required to be highly accurate. To achieve this goal, the teacher is forced to escalate the performance of its model to an excellent level. **Third**, our method LBT(RN50,P-DARTS) achieves the lowest error among all methods in this table. This shows that our method is very competitive in bringing the NAS performance to a state-of-the-art level. **Fourth**, while our LBT framework can greatly help to improve the quality of searched architectures, it does not substantially increase the number of model parameters or search cost. As shown

Table 3: Results on CIFAR-10, including classification error (%) on the test set, number of model weights (millions), and search cost (GPU days). * denotes that the results are taken from DARTS^{-} (Chu et al., 2020a), NoisyDARTS (Chu et al., 2020b), and DrNAS (Chen et al., 2020). The rest notations are the same as those in Table 2.

Method	Error(%)	Param(M)	Cost
*DenseNet (Huang et al., 2017)	3.46	25.6	-
*HierEvol (Liu et al., 2018b)	3.75±0.12	15.7	300
*NAONet-WS (Luo et al., 2018)	3.53	3.1	0.4
*PNAS (Liu et al., 2018a)	3.41±0.09	3.2	225
*ENAS (Pham et al., 2018)	2.89	4.6	0.5
*NASNet-A (Zoph et al., 2018)	2.65	3.3	1800
*AmoebaNet-B (Real et al., 2019)	2.55±0.05	2.8	3150
*R-DARTS (Zela et al., 2020)	2.95±0.21	-	1.6
*GDAS (Dong and Yang, 2019)	2.93	3.4	0.2
*SNAS (Xie et al., 2019)	2.85	2.8	1.5
*BayesNAS (Zhou et al., 2019)	2.81±0.04	3.4	0.2
*MergeNAS (Wang et al., 2020)	2.73±0.02	2.9	0.2
*NoisyDARTS (Chu et al., 2020b)	2.70±0.23	3.3	0.4
*ASAP (Noy et al., 2020)	2.68±0.11	2.5	0.2
*SDARTS (Chen and Hsieh, 2020)	2.61±0.02	3.3	1.3
*DropNAS (Hong et al., 2020)	2.58±0.14	4.1	0.6
*PC-DARTS (Xu et al., 2020)	2.57±0.07	3.6	0.1
*FairDARTS (Chu et al., 2019)	2.54	3.3	0.4
*DrNAS (Chen et al., 2020)	2.54±0.03	4.0	0.4
*GTN (Such et al., 2019)	2.92±0.06	8.2	0.67
*GTN(F=128) (Such et al., 2019)	2.42±0.03	97.9	0.67
*DARTS-1st (Liu et al., 2019)	3.00±0.14	3.3	0.4
LBT(RN18,DARTS-1st) (ours)	2.87±0.05	3.2	0.6
LBT(RN50,DARTS-1st) (ours)	**2.79**±0.07	3.3	0.7
*DARTS-2nd (Liu et al., 2019)	2.76±0.09	3.3	1.5
LBT(RN18,DARTS-2nd) (ours)	2.65±0.03	3.4	1.9
LBT(RN50,DARTS-2nd) (ours)	**2.61**±0.05	3.4	2.1
*P-DARTS (Chen et al., 2019)	2.50	3.4	0.3
LBT(RN18,P-DARTS) (ours)	2.64±0.11	3.4	0.4
LBT(RN50,P-DARTS) (ours)	2.57±0.15	3.4	0.5
*PC-DARTS (Xu et al., 2020)	2.57±0.07	3.6	0.1
LBT(RN18,PC-DARTS) (ours)	2.59±0.03	3.7	0.1
LBT(RN50,PC-DARTS) (ours)	2.56±0.04	3.7	0.2

in the third column and fourth column, the model size and search cost of our methods are similar to those of baselines.

In Table 3, we show the results on CIFAR-10, including the classification error on the test set, number of model parameters, and search cost measured by GPU days. From

Table 4: Results on ImageNet, including top-1 and top-5 classification errors on the test set, number of model weights (millions), and search cost (GPU days). * denotes that the results are taken from DARTS$^-$ (Chu et al., 2020a) and DrNAS (Chen et al., 2020). The rest notations are the same as those in Table 2. From top to bottom, on the first, second, and third panel are manually-designed networks, non-differentiable search methods, and differentiable search methods. LBT(RN50,DARTS-2nd,CIFAR10) means the architecture is searched using LBT on CIFAR-10 with ResNet-50 as the student, where the search space is the same as that in DARTS-2nd. Similar meanings hold for other notations like this.

Method	Top-1 Error (%)	Top-5 Error (%)	Param (M)	Cost (GPU days)
*Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	-
*MobileNet (Howard et al., 2017)	29.4	10.5	4.2	-
*ShuffleNet 2× (v2) (Ma et al., 2018)	25.1	7.6	7.4	-
*NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	1800
*PNAS (Liu et al., 2018a)	25.8	8.1	5.1	225
*MnasNet-92 (Tan et al., 2019)	25.2	8.0	4.4	1667
*AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	3150
*SNAS-CIFAR10 (Xie et al., 2019)	27.3	9.2	4.3	1.5
*BayesNAS-CIFAR10 (Zhou et al., 2019)	26.5	8.9	3.9	0.2
*PARSEC-CIFAR10 (Casale et al., 2019)	26.0	8.4	5.6	1.0
*GDAS-CIFAR10 (Dong and Yang, 2019)	26.0	8.5	5.3	0.2
*DSNAS-ImageNet (Hu et al., 2020)	25.7	8.1	-	-
*SDARTS-ADV-CIFAR10 (Chen and Hsieh, 2020)	25.2	7.8	5.4	1.3
*PC-DARTS-CIFAR10 (Xu et al., 2020)	25.1	7.8	5.3	0.1
*ProxylessNAS-ImageNet (Cai et al., 2019)	24.9	7.5	7.1	8.3
*FairDARTS-CIFAR10 (Chu et al., 2019)	24.9	7.5	4.8	0.4
*FairDARTS-ImageNet (Chu et al., 2019)	24.4	7.4	4.3	3.0
*DrNAS-ImageNet (Chen et al., 2020)	24.2	7.3	5.2	3.9
*DARTS$^-$-ImageNet (Chu et al., 2020a)	23.8	7.0	4.9	4.5
*DARTS$^+$-CIFAR100 (Liang et al., 2019)	23.7	7.2	5.1	0.2
*DARTS-2nd(CIFAR10) (Liu et al., 2019)	26.7	8.7	4.7	1.5
LBT(RN50,DARTS-2nd,CIFAR10) (ours)	**25.5**	**7.9**	4.8	2.1
*P-DARTS(CIFAR10) (Chen et al., 2019)	24.4	7.4	4.9	0.3
LBT(RN50,P-DARTS,CIFAR10) (ours)	**24.1**	**7.2**	4.9	0.5
*P-DARTS(CIFAR100) (Chen et al., 2019)	24.7	7.5	5.1	0.3
LBT(RN50,P-DARTS,CIFAR100) (ours)	**24.2**	**7.1**	5.3	0.6
*PC-DARTS(ImageNet) (Xu et al., 2020)	24.2	7.3	5.3	3.8
LBT(RN50,PC-DARTS,ImageNet) (ours)	**23.5**	**6.8**	5.4	4.1

this table, we make similar observations as those in Table 2. **First**, our proposed LBT framework is widely effective in helping different NAS methods to improve the quality of searched architectures. For example, applying LBT to DARTS-2nd reduces the error from 2.76% to 2.61%. This further demonstrates that by teaching a student model to learn well, the teacher can improve itself greatly. In the GTN (Such et al., 2019) baseline approach where the architecture of a student is searched by leveraging the synthetic data generated by a trainable generative model, the classification error is 2.92%, which is much worse than those achieved by our methods, while the number of parameters in GTN is twice more than ours. Setting the network width to 128, GTN achieves an error of 2.42%; however, the resulting number of parameters in GTN is about 30 times more than ours. GTN focuses on searching the student's architecture while our method focuses on searching the teacher's architecture. These results demonstrate that searching teacher's architecture is more advantageous than searching student's architecture. **Second**, using ResNet with 50 layers (RN50) as the student results in better performance than using ResNet-18 (RN18), which further demonstrates that teaching a stronger student can drive the teacher to learn better. **Third**, while achieving better classification accuracy than baselines, our method does not substantially increase the model size or search cost compared with baselines.

The results on ImageNet are shown in Table 4, including top-1 and top-5 classification errors on the test set, number of weight parameters (millions), and search costs (GPU days). LBT(RN50,DARTS-2nd,CIFAR10), which is the architecture searched by applying LBT to DARTS-2nd on CIFAR10 with ResNet-50 as the student, achieves lower error than DARTS-2nd(CIFAR10) which does not use LBT. Similarly, LBT(RN50,P-DARTS,CIFAR10) outperforms P-DARTS(CIFAR10), LBT(RN50,P-DARTS,CIFAR100) outperforms P-DARTS(CIFAR100), and LBT(RN50,PC-DARTS,ImageNet) outperforms PC-DARTS(ImageNet). These results again demonstrate the effectiveness of our method in improving a model by letting it teach another model to learn well.

4.4. Ablation Studies

In this section, we perform several ablation studies to better understand the individual learning stages in LBT. For each ablation setting, we compare it with the full LBT framework.

- **Ablation setting 1**. In this setting, the teacher updates its architecture by minimizing the validation loss of the student only, without considering the validation loss of itself. The corresponding formulation is:

$$\begin{aligned} &\min_{A} \;\; L(S^*(T^*(A)), D_s^{(\mathrm{val})}) \\ &s.t. \;\; S^*(T^*(A)) = \min_{S} \;\; L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A))) \\ &\qquad T^*(A) = \min_{T} L(A, T, D_t^{(\mathrm{tr})}) \end{aligned}$$

  In this study, λ is set to 1. The student's architecture is ResNet-18. LBT is applied to DARTS-2nd.

- **Ablation setting 2**. In this setting, in the second stage of LBT, only the pseudo-labeled dataset is used to train the student; $D_s^{(\mathrm{tr})}$ labeled by humans is not used. The corresponding formulation is:

$$\begin{aligned} &\min_{A} \;\; L(T^*(A), A, D_t^{(\mathrm{val})})) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})}) \\ &s.t. \;\; S^*(T^*(A)) = \min_{S} \;\; L(S, D_{pl}(D_u, T^*(A))) \\ &\qquad T^*(A) = \min_{T} L(A, T, D_t^{(\mathrm{tr})}) \end{aligned}$$

Table 5: Classification errors in ablation setting 1. "Student only" means that only the student's validation loss is used to update the teacher's architecture. "Student + teacher" means that both the student's validation loss and teacher's validation loss are minimized to update the teacher's architecture.

Method	Error (%)
Student only (CIFAR-100)	20.27±0.24
Student + Teacher (CIFAR-100)	**17.93**±0.18
Student only (CIFAR-10)	3.01±0.08
Student + teacher (CIFAR-10)	**2.61**±0.05

In this study, γ is set to 1. The student's architecture is ResNet-18. LBT is applied to DARTS-2nd.

- Ablation study on λ. We investigate how the teacher's test error changes with the tradeoff parameter λ in Eq.(3). In this study, the other tradeoff parameter γ in Eq.(3) is set to 1. For either CIFAR-100 or CIFAR-10, from their training set and validation set, 5K examples are uniformly sampled to form a new test set. Architecture search is performed on the remaining training and validation sets. Architecture evaluation result is reported on the 5K new test set. Student's architecture is ResNet-18. LBT is applied to DARTS-2nd.
- Ablation study on γ. We investigate how the teacher's test error changes with the tradeoff parameter γ in Eq.(3). In this study, the other tradeoff parameter λ is set to 1. Similar to the ablation study on λ, the test error is reported on the 5K dataset. Student's architecture is ResNet-18. LBT is applied to DARTS-2nd.

In Table 5, we show the classification errors on the test set of CIFAR-10 and CIFAR-100 in ablation setting 1. As can be seen, on both datasets, minimizing both student's validation loss and teacher's validation loss results in better architectures for the teacher than minimizing student's validation only. The reason is that student's validation loss indirectly measures the quality of the teacher's architecture. How well the student performs depends on not only how well the teacher teaches the student but also how strong the student itself is. If the student is a very strong learner, its validation loss may be largely determined by the student itself and less influenced by the teacher. In this case, student's validation would be a relatively weak signal for guiding the learning of the teacher. In contrast, the validation loss of the teacher directly depends on its architecture and can serve as a direct (hence strong) signal to guide the teacher to learn. In the end, combining the direct signal (teacher's validation loss) and indirect signal (student's validation loss) together is more beneficial than using the indirect signal only.

In Table 6, we show the classification errors on the test sets of CIFAR-10 and CIFAR-100 in ablation setting 2. We can see that using both the pseudo-labeled dataset and human-labeled dataset to train the student yields better performance than using the pseudo-labeled dataset only. The reason is that since the pseudo-labels are automatically generated by a model, they are not entirely reliable. Trained on less reliable labels, the student's model

Table 6: Classification errors in ablation setting 2. "Pseudo labels only" means in the second learning stage, only the pseudo-labeled dataset is used to train the student. "Pseudo labels + human labels" means both the pseudo-labeled dataset and a human-labeled dataset $D_s^{(\mathrm{tr})}$ are used to train the student.

Method	Error (%)
Pseudo labels only (CIFAR-100)	19.82±0.31
Pseudo labels + human labels (CIFAR-100)	**17.93**±0.18
Pseudo labels only (CIFAR-10)	2.93±0.07
Pseudo labels + human labels (CIFAR-10)	**2.61**±0.05

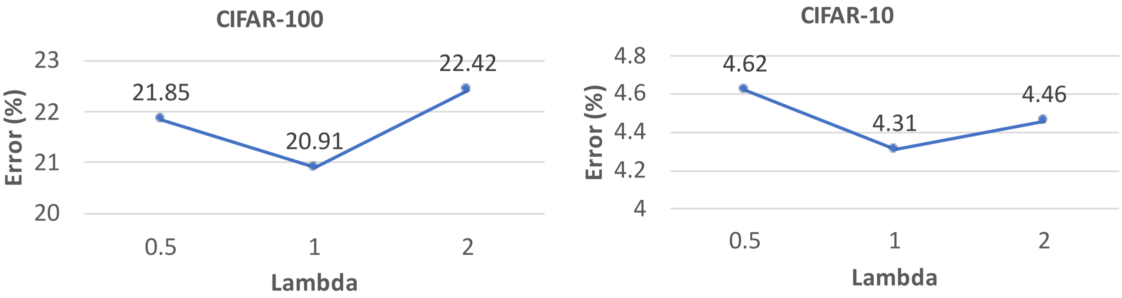

Figure 3: How errors change as λ increases.

may have low quality and a poorly-performing student cannot drive the teacher to learn better. This risk can be reduced by incorporating human-provided labels which are more reliable. As a result, using human labels and pseudo-labels jointly yields better performance than solely using pseudo-labels.

In Figure 3, we show how the classification errors of LBT on CIFAR-10 and CIFAR-100 vary as λ increases. From the plot on CIFAR-100, we observe the following. First, when λ increases from 0.5 to 1, the error decreases. This is because a larger λ incurs a stronger effect of teaching, where the training of the student relies more on the pseudo-labeled dataset created by the teacher. When the teaching effect is strong, the teacher can gain more feedback from the student's performance, which helps the teacher to learn better. Second, if we continue to increase λ, the performance becomes worse. The reason is that if λ is too large, the teaching effect would be excessively strong. Under such circumstances, the student is mainly trained on the pseudo labels which are less reliable than human-provided labels and consequently its model may be of low quality. A mediocre student will not be very helpful in driving the teacher to improve. Similar phenomenon are observed from the plot on CIFAR-10.

In Figure 4, we show how the classification errors of LBT vary as we increase γ. As can be seen, on CIFAR-100, when we increase γ from 0.1 to 1, the error decreases. This is because a larger γ encourages the teacher to pay more attention to the feedback obtained from the student. This feedback is valuable because the validation performance of the student reflects the correctness of the pseudo-labels generated by the teacher and the quality of pseudo-

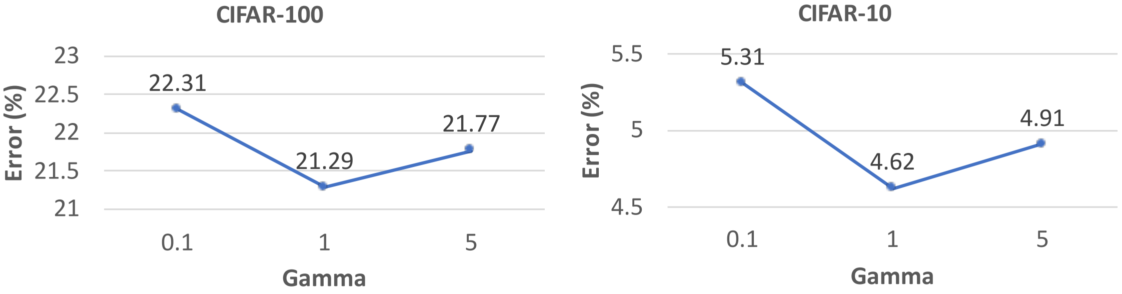

Figure 4: How errors change as γ increases.

labels reflects the quality of the teacher's architecture. Paying more attention to such feedback enables the teacher to identify its weakness and strive for improvement. However, if γ is too large, the learning of the teacher's architecture would be guided excessively by student's validation loss which is an indirect (hence weaker signal) but inadequately influenced the validation loss of the teacher itself which is a direct (hence stronger signal). Similar observations can be made from the plot on CIFAR-10 as well.

5. Conclusions

Motivated by the teaching-driven learning methodology of humans, we propose a novel machine learning framework called learning by teaching (LBT). LBT improves the training of a teacher model by encouraging the teacher to teach what it has learned to a student model. Based on how the student performs on a validation set, the teacher refines its model. The intuition behind LBT is that to be able to teach others to well accomplish a task, the teacher itself needs to thoroughly understand this task and know how to perform it excellently. We develop a multi-level optimization framework to formulate LBT and the framework consists of three learning stages: the teacher learns; the teacher teaches the student by pseudo-labeling; the teacher improves its architecture based on the validation results of itself and of the student. Our framework is applied for neural architecture search on CIFAR-100, CIFAR-10, and ImageNet. The results demonstrate the effectiveness of our method.

References

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

Francesco Paolo Casale, Jonathan Gordon, and Nicoló Fusi. Probabilistic neural architecture search. *CoRR*, abs/1902.05116, 2019.

Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. *CoRR*, abs/2002.05283, 2020.

Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *CoRR*, abs/2006.10355, 2020.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. *CoRR*, abs/1911.12126, 2019.

Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: robustly stepping out of performance collapse without indicators. *CoRR*, abs/2009.01027, 2020a.

Xiangxiang Chu, Bo Zhang, and Xudong Li. Noisy differentiable architecture search. *CoRR*, abs/2005.03566, 2020b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *CVPR*, 2019.

Logan Fiorella and Richard E Mayer. The relative benefits of learning by teaching and teaching expectancy. *Contemporary Educational Psychology*, 38(4):281–288, 2013.

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 *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015a.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015b.

Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. Dropnas: Grouped operation dropout for differentiable architecture search. In *IJCAI*, 2020.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. DSNAS: direct neural architecture search without parameter retraining. In *CVPR*, 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Aloysius Wei Lun Koh, Sze Chi Lee, and Stephen Wee Hun Lim. The learning benefits of teaching: A retrieval practice hypothesis. *Applied Cognitive Psychology*, 32(3):401–410, 2018.

Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. DARTS+: improved differentiable architecture search with early stopping. *CoRR*, abs/1909.06035, 2019.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018a.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018b.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, 2018.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. ASAP: architecture search, anneal and prune. In *AISTATS*, 2020.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

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*, volume 33, pages 4780–4789, 2019.

Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. *CoRR*, abs/1912.07768, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. Multi-fidelity neural architecture search with knowledge distillation. *CoRR*, abs/2006.08341, 2020.

Xiaoxing Wang, Chao Xue, Junchi Yan, Xiaokang Yang, Yonggang Hu, and Kewei Sun. Mergenas: Merge operations into one for differentiable architecture search. In *IJCAI*, 2020.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.