# Learning by Teaching, with Application to Neural Architecture Search

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

## Abstract

Learning by teaching is a broadly used methodology in human learning and shows great effectiveness in improving learning outcome: a learner deepens his/her understanding of a topic by teaching this topic to others. We are interested in investigating whether this powerful learning technique can be borrowed from humans to improve the learning abilities of machines. We propose a novel machine learning approach called learning by teaching (LBT). In our approach, the 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. We propose a multi-level optimization framework to formulate LBT which involves three learning stages: teacher learns; teacher teaches student; teacher and student validate themselves. We develop an efficient algorithm to solve the LBT problem. We apply our approach to neural architecture search on CIFAR-100, CIFAR-10, and ImageNet, where the results demonstrate the effectiveness of our method.

## 1. Introduction

In human learning, an effective and widely used methodology for improving learning outcome is learning by teaching (LBT). 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.

Inspired by this teaching-driven learning technique of humans, we are interested in investigating whether this methodology is helpful for improving machine learning as well. We propose a novel learning framework called learning by teaching (LBT). In this framework, there is a teacher model and a student model. They perform the same learning task (e.g., image classification). 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 improve the teacher's learning capability and outcome during the teaching process. 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:

---

. *Corresponding author.

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, both the teacher and student perform learning. There are three learning stages. In the first stage, the teacher trains its network weights on its training dataset, with its architecture fixed. In the second stage, the teacher uses its optimally-trained model in the first stage to make predictions on an unlabeled dataset and generates a pseudo-labeled dataset; the student trains its network weights on the pseudo-labeled dataset and a human-labeled training set. In the third stage, the teacher updates its architecture by minimizing its validation loss and the student's validation loss. The three stages are performed jointly end-to-end in a multi-level optimization framework, where different stages influence each other. We apply our method for neural architecture search in image classification tasks. The results on CIFAR-100, CIFAR-10, and ImageNet (Deng et al., 2009) demonstrate the effectiveness of our method.

The major contributions of this paper is as follows:

- Inspired by the teaching-driven learning technique of humans, we propose a novel machine learning approach called learning by teaching (LBT). In our approach, the 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.
- We propose a multi-level optimization framework to formulate LBT which involves three stages: teacher learns; teacher teaches student; teacher and student validate themselves.
- We develop an efficient algorithm to solve the LBT problem.
- We apply our approach to neural architecture search. The results on CIFAR-100, CIFAR-10, and ImageNet demonstrate the effectiveness of our method.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 and 4 present the method and experiments respectively. Section 5 concludes the paper.

## 2. Methods

In this section, we propose a framework for learning by teaching (LBT) and develop an optimization for solving the LBT problem.

| 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 |

Table 1: Notations in Learning by Teaching

### 2.1. 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 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 trains its network weights on its training dataset, with the architecture fixed:

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

The architecture $A$ is used to define the training loss. But it is not updated at this stage. If $A$ is learned by minimizing this training loss, a trivial solution will be yielded where $A$ is very large and complex that it can perfectly overfit the training data but will generalize poorly on unseen data. Note that the optimally trained weights $T^*(A)$ is a function of $A$ since $T^*(A)$ is a function of $L(A, T, D_t^{(\mathrm{tr})})$ and $L(A, T, D_t^{(\mathrm{tr})})$ is a function of $A$. In the second stage, the teacher uses $T^*(A)$ to make predictions on $D_u = \{x_i\}_{i=1}^N$ and generates a pseudo-labeled dataset $D_{pl}(D_u, T^*(A)) = \{(x_i, T_A^*(x_i))\}_{i=1}^N$. $T_A^*(x_i)$ is a $K$-dimensional vector where $K$ is the number of classes and the $k$-th element of $T_A^*(x_i)$ is the predicted probability that $x_i$ belongs to the $k$-th class. The sum of all elements in $T_A^*(x_i)$ is one. Given this pseudo-labeled dataset, the student leverages $D_{pl}(D_u, T^*(A))$ together with $D_s^{(\mathrm{tr})}$ to train its network weights $S$:

$$S^*(T^*(A)) = \min_S \; L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A))), \tag{2}$$

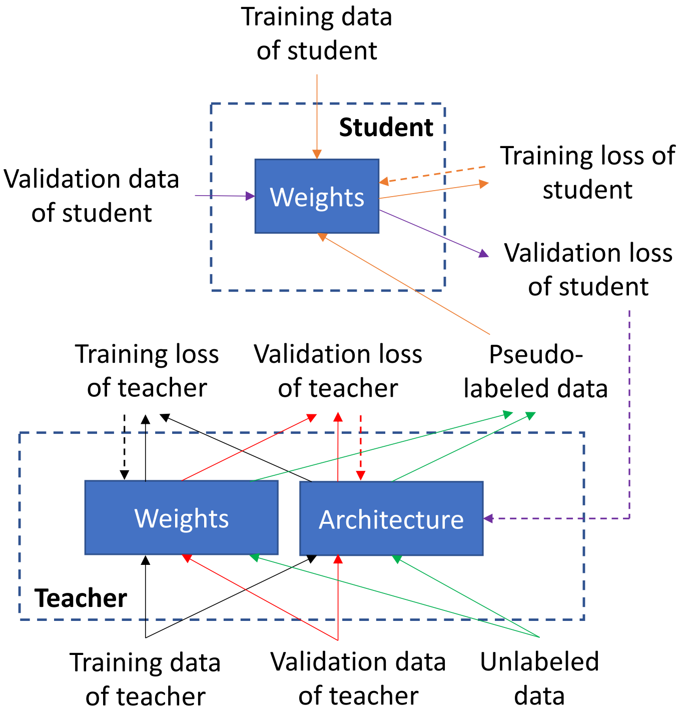

Figure 1: Illustration of learning by teaching. The solid arrows denote the process of making predictions and calculating losses. The dotted arrows denote the process of updating learnable parameters by minimizing corresponding losses.

where the first loss is defined on the human-labeled training dataset and the second loss is defined on the teacher-labeled dataset. Both losses are cross-entropy losses. $\lambda$ is a tradeoff parameter. Note that $S^*(T^*(A))$ is a function of $T^*(A)$ since $S^*(T^*(A))$ is a function of $L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A)))$ which is a function of $T^*(A)$. In the third stage, the teacher validates its network weights $T^*(A)$ on its validation set $D_t^{(\mathrm{val})}$ and the student validates its network weights $S^*(T^*(A))$ on its validation set $D_s^{(\mathrm{val})}$. The teacher optimizes its architecture by minimizing its validation loss and the student's validation loss:

$$\min_A \; L(T^*(A), A, D_t^{(\mathrm{val})}) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})}), \tag{3}$$

where $\gamma$ is a tradeoff parameter.

The three stages are mutually dependent: $T^*(A)$ learned in the first stage is used to define the objective function in the second stage; $T^*(A)$ and $S^*(T^*(A))$ learned in the first two stages are used to define the objective in the third stage; the updated $A$ in the third stage in turn changes the objective function in the first stage, which subsequently renders $T^*(A)$ to be changed.

Putting these pieces together, we have the following LBT framework, which is a three-level optimization problem:

$$\begin{aligned} \min_{A} \;\; & L(T^*(A), A, D_t^{(\mathrm{val})})) + \gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})}) \\ s.t. \;\; & S^*(T^*(A)) = \min_{S} \;\; L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A))) \\ & T^*(A) = \min_{T} L(A, T, D_t^{(\mathrm{tr})}) \end{aligned} \tag{4}$$

This formulation nests three optimization problems. On the constraints of the outer optimization problem are two inner optimization problems corresponding to the first and second learning stage respectively. The objective function of the outer optimization problem corresponds to the third learning stage.

Similar to (Liu et al., 2019), we represent the architecture $A$ of the teacher in a differentiable way. The search space of $A$ is composed of a large number of building blocks. The output of each block is associated with a variable $a$ indicating how important this block is. After learning, blocks whose $a$ is among the largest are retained to form the final architecture. In this end, architecture search amounts to optimizing the set of architecture variables $A = \{a\}$.

### 2.2. Optimization Algorithm

In this section, we derive an optimization algorithm to solve the LBT problem defined in Eq.(4). Inspired by (Liu et al., 2019), we approximate $T^*(A)$ using one-step gradient descent update of $T$ with respect to $L(A, T, D_t^{(\mathrm{tr})})$. We plug the approximation $T'$ of $T^*(A)$ into $L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A)))$ and obtain an approximated objective $O_s$. Then we approximate $S^*(T^*(A))$ using one-step gradient descent update of $S$ with respect to $O_s$. Finally, we plug $T'$ and the approximation $S'$ of $S^*(T^*(A))$ into $L(T^*(A), D_t^{(\mathrm{val})}))+\gamma L(S^*(T^*(A)), D_s^{(\mathrm{val})})$ and perform gradient-descent update of $A$ with respect to this approximated objective. In the sequel, we use $\nabla^2_{Y,X} f(X, Y)$ to denote $\frac{\partial f(X,Y)}{\partial X \partial Y}$.

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

$$T' = T - \xi_t \nabla_T L(T, A, D_t^{(\mathrm{tr})}) \tag{5}$$

where $\xi_t$ is a learning rate. Plugging $T'$ into $L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T^*(A)))$, we obtain an approximated objective $O_s = L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T'))$. Then we approximate $S^*(T^*(A))$ using one-step gradient descent update of $S$ with respect to $O_s$:

$$S' = S - \xi_s \nabla_S (L(S, D_s^{(\mathrm{tr})}) + \lambda L(S, D_{pl}(D_u, T'))). \tag{6}$$

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 $A$:

$$A \leftarrow A - \eta(\nabla_A L(T', A, D_t^{(\mathrm{val})}) + \gamma \nabla_A L(S', D_s^{(\mathrm{val})})) \tag{7}$$

where

$$\begin{aligned}&\nabla_A L(T', A, D_t^{(\text{val})}) = \\ &\nabla_A L(T - \xi_t \nabla_T L(T, A, D_t^{(\text{tr})}), A, D_t^{(\text{val})}) = \\ &-\xi_t \nabla^2_{A,T} L(T, A, D_t^{(\text{tr})}) \nabla_{T'} L(T', A, D_t^{(\text{val})}) + \nabla_A L(T', A, D_t^{(\text{val})})\end{aligned} \tag{8}$$

The first term in the third line involves expensive matrix-vector product, whose computational complexity can be reduced by a finite difference approximation:

$$\begin{aligned}&\nabla^2_{A,T} L(T, A, D_t^{(\text{tr})}) \nabla_{T'} L(T', A, D_t^{(\text{val})}) \approx \\ &\tfrac{1}{2\alpha}(\nabla_A L(T^+, A, D_t^{(\text{tr})}) - \nabla_A L(T^-, A, D_t^{(\text{tr})})),\end{aligned} \tag{9}$$

where $T^{\pm} = T \pm \alpha \nabla_{T'} L(T', A, D_t^{(\text{val})})$ and $\alpha$ is a small scalar that equals $0.01/\|\nabla_{T'} L(T', A, D_t^{(\text{val})})\|_2$.

For $\nabla_A L(S', D_s^{(\text{val})})$ in Eq.(7), it can be calculated as $\frac{\partial S'}{\partial A} \nabla_{S'} L(S', D_s^{(\text{val})})$ according to the chain rule, where

$$\frac{\partial S'}{\partial A} = \frac{\partial (S - \xi_s \nabla_S (L(S, D_s^{(\text{tr})}) + \lambda L(S, D_{pl}(D_u, T'))))}{\partial A} \tag{10}$$

$$= \frac{\partial (-\xi_s \lambda \nabla_S L(S, D_{pl}(D_u, T')))}{\partial A} \tag{11}$$

$$= -\xi_s \lambda \frac{\partial T'}{\partial A} \nabla^2_{T',S} L(S, D_{pl}(D_u, T')) \tag{12}$$

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

$$\frac{\partial T'}{\partial A} = \frac{\partial (T - \xi_t \nabla_T L(T, A, D_t^{(\text{tr})}))}{\partial A} = -\xi_t \nabla^2_{A,T} L(T, A, D_t^{(\text{tr})}) \tag{13}$$

The algorithm for solving LBT is summarized 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.(5)
    2. Update the student's network weights $S$ using Eq.(6)
    3. Update the teacher's architecture $A$ using Eq.(7)
end
```

## 3. Experiments

We apply LBT for neural architecture search in image classification tasks. Following (Liu et al., 2019), we first perform architecture search which finds out an optimal cell, then perform architecture evaluation which composes multiple copies of the searched cell into a large network, trains it from scratch, and evaluates the trained model on the test set.

### 3.1. Datasets

We used three datasets in the experiments: CIFAR-10, CIFAR-100, and ImageNet (Deng et al., 2009). The CIFAR-10 dataset contains 50K training images and 10K testing images, from 10 classes (the number of images in each class is equal). Following (Liu et al., 2019), we split the original 50K training set into a new 25K training set and a 25K validation set. In the sequel, when we mention "training set", it always refers to the new 25K training set. During architecture search, the training set is used as $D_t^{(\mathrm{tr})}$ and $D_s^{(\mathrm{tr})}$. The validation set is used as $D_t^{(\mathrm{val})}$ and $D_s^{(\mathrm{val})}$. During architecture evaluation, the combination of the training data and validation data is used to train the large network stacking multiple copies of the searched cell. The CIFAR-100 dataset contains 50K training images and 10K testing images, from 100 classes (the number of images in each class is equal). Similar to CIFAR-100, the 50K training images are split into a 25K training set and 25K validation set. The usage of the new training set and validation set is the same as that for CIFAR-10. When searching architectures on CIFAR-10 in LBT, CIFAR-100 (removing class labels) is used as the unlabeled dataset. When searching architectures on CIFAR-100 in LBT, CIFAR-10 (removing class labels) is used as the unlabeled dataset. The ImageNet dataset contains a training set of 1.2M images and a validation set of 50K images, from 1000 object classes. The validation set is used as a test set for architecture evaluation. Following (Liu et al., 2019), we evaluate the architectures searched using CIFAR-10 and CIFAR-100 on ImageNet: given a cell searched using CIFAR-10 and CIFAR-100, multiple copies of it compose a large network, which is then trained on the 1.2M training data of ImageNet and evaluated on the 50K test data.

### 3.2. Experimental Settings

In our framework, the search space of the teacher's architecture is the same as that in DARTS (Liu et al., 2019). The candidate operations include: $3 \times 3$ and $5 \times 5$ separable convolutions, $3 \times 3$ and $5 \times 5$ dilated separable convolutions, $3 \times 3$ max pooling, $3 \times 3$ average pooling, identity, and zero. In LBT, the network of the teacher is a stack of multiple cells, each consisting of 7 nodes. For the architecture of the student, we used ResNet-18 (He et al., 2016b). $\lambda$ and $\gamma$ are both set to 1.

For CIFAR-10 and CIFAR-100, during architecture search, the teacher's network is a stack of 8 cells, with the initial channel number set to 16. The search is performed for 50 epochs, with a batch size of 64. The hyperparameters for the teacher's architecture and weights are set in the same way as DARTS and PC-DARTS. The network weights of the student are optimized using SGD with a momentum of 0.9 and a weight decay of 3e-4. The initial learning rate is set to 0.025 with a cosine decay scheduler. During architecture evaluation, 20 copies of the searched cell are stacked to form the teacher's network, with the initial channel number set to 36. The network is trained for 600 epochs with a batch size of 96 (for both CIFAR-10 and CIFAR-100). The experiments are performed on a single Tesla v100. For ImageNet, following (Liu et al., 2019), we take the architecture searched on CIFAR-10 and evaluate it on ImageNet. We stack 14 cells (searched on CIFAR-10) to form a large network and set the initial channel number as 48. The network is trained for 250 epochs with a batch size of 1024 on 8 Tesla v100s. Each experiment on LBT is repeated

| 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 |
| †DARTS-1st (Liu et al., 2019) | 20.52±0.31 | 1.8 | 0.4 |
| *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 |
| †DARTS$^-$ (Chu et al., 2020a) | 18.97±0.16 | 3.1 | 0.4 |
| *P-DARTS (Chen et al., 2019) | 17.49 | 3.6 | 0.3 |
| $^\triangle$DARTS$^+$ (Liang et al., 2019) | 17.11±0.43 | 3.8 | 0.2 |
| †PC-DARTS (Xu et al., 2020) | 17.01±0.06 | 4.0 | 0.1 |
| *DropNAS (Hong et al., 2020) | 16.39 | 4.4 | 0.7 |
| *DARTS-2nd (Liu et al., 2019) | 20.58±0.44 | 1.8 | 1.5 |
| LBT-DARTS-2nd (ours) | 17.90 | 2.5 | 2.0 |

Table 2: Results on CIFAR-100, including classification error (%) on the test set, number of parameters (millions) in the searched architecture, and search cost (GPU days). LBT-DARTS-1st denotes that our method LBT is applied to the search space of DARTS. Similar meanings hold for other notations in such a format. DARTS-1st and DARTS-2nd denotes that first order and second order approximation is used in DARTS. * means the results are taken from DARTS$^-$ (Chu et al., 2020a). † means we re-ran this method for 10 times. The search cost is measured by GPU days on a Tesla v100.

for ten times with the random seed to be from 1 to 10. We report the mean and standard deviation of results obtained from the 10 runs.

### 3.3. Results

Table 2 shows the classification error (%), number of weight parameters (millions), and search cost (GPU days) of different NAS methods on CIFAR-100. From this table, we can see that when our method LBT is applied to DARTS-2nd (second order approximation), the test error is greatly reduced from 20.58% to 17.90%. This demonstrates the effectiveness of our method in searching for a better architecture. 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 in the pseudo-labeled dataset are not correct, which indicates that the teacher's model is not accurate. To avoid such an outcome, the teacher enforces itself to learn better to generate correct pseudo labels.

| 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 |
| *DARTS-1st (Liu et al., 2019) | 3.00±0.14 | 3.3 | 0.4 |
| *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 |
| *P-DARTS (Chen et al., 2019) | 2.50 | 3.4 | 0.3 |
| *DARTS-2nd (Liu et al., 2019) | 2.76±0.09 | 3.3 | 1.5 |
| LBT-DARTS-2nd (ours) | 2.67 | 3.4 | 1.8 |

Table 3: Results on CIFAR-10. * means 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.

Table 3 shows the classification error (%), number of weight parameters (millions), and search cost (GPU days) of different NAS methods on CIFAR-10. As can be seen, applying our proposed LBT to DARTS-2nd reduces the error from 2.76% to 2.67%. This further demonstrates the efficacy of our method in searching better-performing architectures, by teaching a student model to perform well on the classification task.

Table 4 shows the results on ImageNet, including top-1 and top-5 classification errors on the test set, number of weight parameters (millions), and search costs (GPU days). Following (Liu et al., 2019), we take the architecture searched by LBT-DARTS-2nd on CIFAR-10 and evaluate it on ImageNet. As can be seen, applying our LBT method to DARTS-2nd, the top-1 error reduces from 26.7% to 26.1% . This further demonstrates the effectiveness of our method.

| 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× (v1) (Zhang et al., 2018) | 26.4 | 10.2 | 5.4 | - |
| *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 |
| *P-DARTS (CIFAR100) (Chen et al., 2019) | 24.7 | 7.5 | 5.1 | 0.3 |
| *P-DARTS (CIFAR10) (Chen et al., 2019) | 24.4 | 7.4 | 4.9 | 0.3 |
| *DrNAS-ImageNet (Chen et al., 2020) | 24.2 | 7.3 | 5.2 | 3.9 |
| *PC-DARTS-ImageNet (Xu et al., 2020) | 24.2 | 7.3 | 5.3 | 3.8 |
| *DARTS$^{+}$-ImageNet (Liang et al., 2019) | 23.9 | 7.4 | 5.1 | 6.8 |
| *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 | 4.0 |
| LBT-DARTS-2nd-CIFAR10 (ours) | 26.1 | 8.2 | 4.7 | 4.0 |

Table 4: Results on ImageNet, including top-1 and top-5 classification errors on the test set, number of weight parameters (millions), and search cost (GPU days). * means 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. The first row block shows networks designed by humans manually. The second row block shows non-gradient based search methods. The third block shows gradient-based methods.

## 4. Related Works

Neural architecture search (NAS) has achieved remarkable progress recently, which aims at searching for the optimal architecture of neural networks to achieve the best predictive performance. In general, there are three paradigms of methods in NAS: reinforcement learning (RL) approaches (Zoph and Le, 2017; Pham et al., 2018; Zoph et al., 2018), evolutionary learning approaches (Liu et al., 2018b; Real et al., 2019), and differentiable approaches (Cai et al., 2019; Liu et al., 2019; Xie et al., 2019). In RL-based approaches, a policy is learned to iteratively generate new architectures by maximizing a reward which is the accuracy on the validation set. Evolutionary learning approaches represent the architectures as individuals in a population. Individuals with high fitness scores (validation accuracy) have the privilege to generate offspring, which replaces individuals with low fitness scores. Differentiable approaches adopt a network pruning strategy. On top of an over-parameterized network, the weights of connections between nodes are learned using gradient descent. Then weights close to zero are later pruned. There have been many efforts devoted to improving differentiable NAS methods. In P-DARTS (Chen et al., 2019), the depth of searched architectures is allowed to grow progressively during the training process. Search space approximation and regularization approaches are developed to reduce computational overheads and improve search stability. PC-DARTS (Xu et al., 2020) reduces the redundancy in exploring the search space by sampling a small portion of a super network. Operation search is performed in a subset of channels with the held out part bypassed in a shortcut. Our proposed LBT framework can be applied to any differentiable NAS methods.

## 5. Conclusions

In this paper, we propose a new machine learning approach – learning by teaching (LBT), inspired by the teaching-driven learning technique of humans. In LBT, a teacher model improves its learning outcome on a target task by teaching a student model to perform well on this task, based on the intuition that to be able to teach others to well accomplish a task, the teacher itself needs to thoroughly understand this task and knows to how to perform it excellently. We propose a multi-level optimization framework to formalize LBT which involves 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.

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.

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.

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.

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.

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.

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.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

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.