IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 May 2024, accepted 27 May 2024, date of publication 3 June 2024, date of current version 12 June 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3409177*

RESEARCH ARTICLE

# Teacher-Free Knowledge Distillation Based on Non-Progressive Meta-Learned Multi Ranking Selection

**BING-RU JIANG, PIN-HSUAN HO, SYUAN-TING HE, AND ALBERT S. LIN, (Member, IEEE)**
Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu 300093, Taiwan

Corresponding author: Albert S. Lin (hdtd5746@gmail.com)

This work was supported in part by the National Science and Technology Council under Grant NSTC 112-2221-E-A49-173-.

**ABSTRACT** Knowledge Distillation (KD) is the procedure of extracting useful information from a previously trained model using an algorithm. The successful distillation pulls up the distilling model accuracy. In the context of model compression, the teacher model provides softened labels that facilitate distillation. While some effort has been devoted to progressive training, and a majority of the effort in literature has been in teacher-student distillation configuration or methodologies, less effort has been spent in non-progressive training, meta-learning controlled distillation, and progressive training with algorithm-controlled target distribution. In this paper, we proposed a framework of Teacher-Free Knowledge Distillation (TFKD) based on non-progressive meta-learned Reinforcement Learning (RL) method. The student model learns from free-form distribution, which will change during training. In this scenario, the target distribution is varied during the training epochs, and the variation is not necessarily continuously merging toward the true distribution. Due to the algorithm-controlled nature of the target distribution variation during KD, the meta-learning KD is established. We also designed a Multi-Ranking Selection (MRS) procedure to find a more potential model for continued training. We conducted our experiments using VGG-8 on CIFAR-100, CIFAR-10, and SVHN datasets. Our method has improved by 1.62% without MRS and 1.97% with MRS compared with the baseline model on CIFAR-100. Compared to State-Of-The-Art (SOTA) techniques, our approach achieves the highest accuracy of 72.41%.

**INDEX TERMS** Teacher-free knowledge distillation, knowledge distillation, reinforcement learning, meta-learning, model compression.

## I. INTRODUCTION

The development of deep learning has led to achievements in several fields of artificial intelligence, such as computer vision and natural language processing. Most State-Of-The-Art (SOTA) models become more significant for complicated calculations to reach a higher accuracy. However, deploying these high-performance models on edge devices with sparse resources, e.g., mobile phones and embedded devices, may be challenging due to the high complexity and the large model size. Many methods regarding model compression [1], [2] have been proposed to solve this problem, such as pruning [3], [4], quantization [5], Knowledge Distillation (KD) [6], [7], [8], [9], [10], [11], [12], [13], and low-rank matrix decomposition [14]. Network pruning can reduce the model size by deleting unnecessary weights or neurons. Parameter quantization is a method using fewer bits to represent the same values. On the other hand, inspired by how humans learn knowledge from the teacher, KD can transfer the knowledge from a vast teacher model to a small student model. There is also much research combining several model compression methods. For instance, Han et al. [15] use pruning, quantization, and Huffman coding to reduce the model size without accuracy reduction. Polino et al. [16] achieve model compression via distillation and quantization.

KD was first proposed by Buciluǎ et al. [17] and widely promoted by Hinton et al. [8]. The student model can learn the knowledge from the teacher model to improve their

The associate editor coordinating the review of this manuscript and approving it for publication was Wanqing Zhao.

performance. There are two common types of knowledge transfer. The first one only transfers the logits of the last layer [8], [18], [19]. The second method is feature-based knowledge transfers where not only the last layer but also the intermediate layers [7], [20], [21] are distilled so that the student can mimic the feature maps of the teacher. It is generally believed that the hints of intermediate layers are also important for student's learning. For example, Fitnets [7] is a famous model in this area, using the hidden layers of the teacher as hints to guide the student's training process.

The methods of KD above depend on a proper teacher model since, in traditional concepts, only the high-performance, properly-tuned teacher model can enhance the student model significantly. However, this will cause extra computation for training a cumbersome teacher model. If the model we want to train is big originally, it will lead to a problem for us to find another larger model for the training. Besides, the requirement of teacher models limits the flexibility of using label smoothing in KD. Recently, concepts such as Teacher-Free Knowledge Distillation (TFKD) [22] and Self-Knowledge Distillation (Self-KD) [23], [24], [25], [26], [27] were proposed. The central concept of TFKD and Self-KD is that the knowledge is not distilled from a teacher network. While a lot of teacher-based KD effort has been conducted in the literature, TFKD is worth more investigation.

In this paper, we proposed a TFKD method by designing a free-form distribution as the teacher, which will change during the training process [28], [29]. We use an optimization algorithm named the non-progressive meta-learned Reinforcement Learning (RL) method to optimize the distribution. Finally, we design a Multi-Ranking Selection (MRS) procedure to find a more potential model for training. The flowchart of the whole process is shown in Fig. 1. Unlike normal KD, we can enhance a model without training another large model. The free-form distribution can also provide more possibilities for training without the limitations of the teacher's prediction.

The remainder of this paper is organized as follows: Section II describes the related works. Section III describes the methodology and the training process we proposed. The experiment results and discussion are presented in section IV. In section V, the future work is described. In the end, the conclusion is given in section VI.

## II. RELATED WORKS

Li [30] proposed a category that the earliest instance of the teacher-free method can be defined as Dropout [31], DropBlock [32], and self-attention distillation [33]. These pioneering contributions established the basis for further research [22], [30], [34] in this field. Reference [30] purposed a teacher-free feature distillation framework by using intra-layer and inter-layer features for distillation. The intra-layer uses the features in the same layer as knowledge. The inter-layer uses the features of the deeper layer to guide the shallow layer. Some research discusses the relationship between Label Smoothing Regularization (LSR) and KD. Yuan et al. [22] have observations that the student model can enhance the teacher model and a poor teacher can also improve the student model. Thus, they concluded that the regularization of soft targets is also an important part of the success of KD. Based on this idea, they designed a distribution that is similar to LSR with a high probability of the correct class for the student to learn. Wang et al. [34] decomposed TFKD into output smoothing and teacher correction, demonstrating an intuitive approach to replace the traditional LSR method.

The results of this paper will be compared to the current works in the field of teacher-free method [30], [31], [32], [33], [35], [36] and Self-KD [23], [24], [25] in Table 5. Reference [23] presented a strategy for one-stage online distillation which can overcome the need of a strong teacher by training a single network while the teacher network is established simultaneously. Reference [24] purposed a framework of Self-KD by divided the network into several sections that the shallow sections are trained as the student model by distilling from the deeper section which is acted as the teacher model. Reference [25] purposed a regularization method in the dark knowledge of a single network which is named class-wise self-knowledge distillation.This work is included in our student thesis [37], and the related work by our group is [38].

## III. METHODOLOGY

### A. FREE FORM-BASED KNOWLEDGE DISTILLATION

The TFKD method proposed by Yuan et al. [22] considers KD as a type of LSR. The difference is that KD learns from the teacher's distribution, and the distribution in LSR is a pre-defined uniform distribution. Inspired by this, we let the model learn with the free-form distribution designed as follows:

$$P\_vector\ (k) = \begin{cases} z, & if\ k = c \\ random\left(\frac{100-z}{99}, \frac{100-z}{50}\right), & if\ k \neq c \end{cases} \tag{1}$$

where $z$ is a value randomly chosen from 90 to 99, $c$ is the correct label. In this formula, we initialize the unnormalized values rather than directly specify the normalized values in the vector, which are the probability distribution. The probability distribution for the student to learn in KD designed in Eq. (1) and (2) will change during training, which is named free-form distribution. Thus, we called our KD method a free-form-based KD.

These numerical values are stored in a list, such as [0.1, 0.15, 0.12, …, 99]. Initially, these numbers are summed together (*P_vector_sum*). Subsequently, they are normalized to a range between 0 and 1 based on their proportional contributions.

$$P\ (k) = \frac{P\_vector\ (k)}{P\_vector\_sum} \tag{2}$$

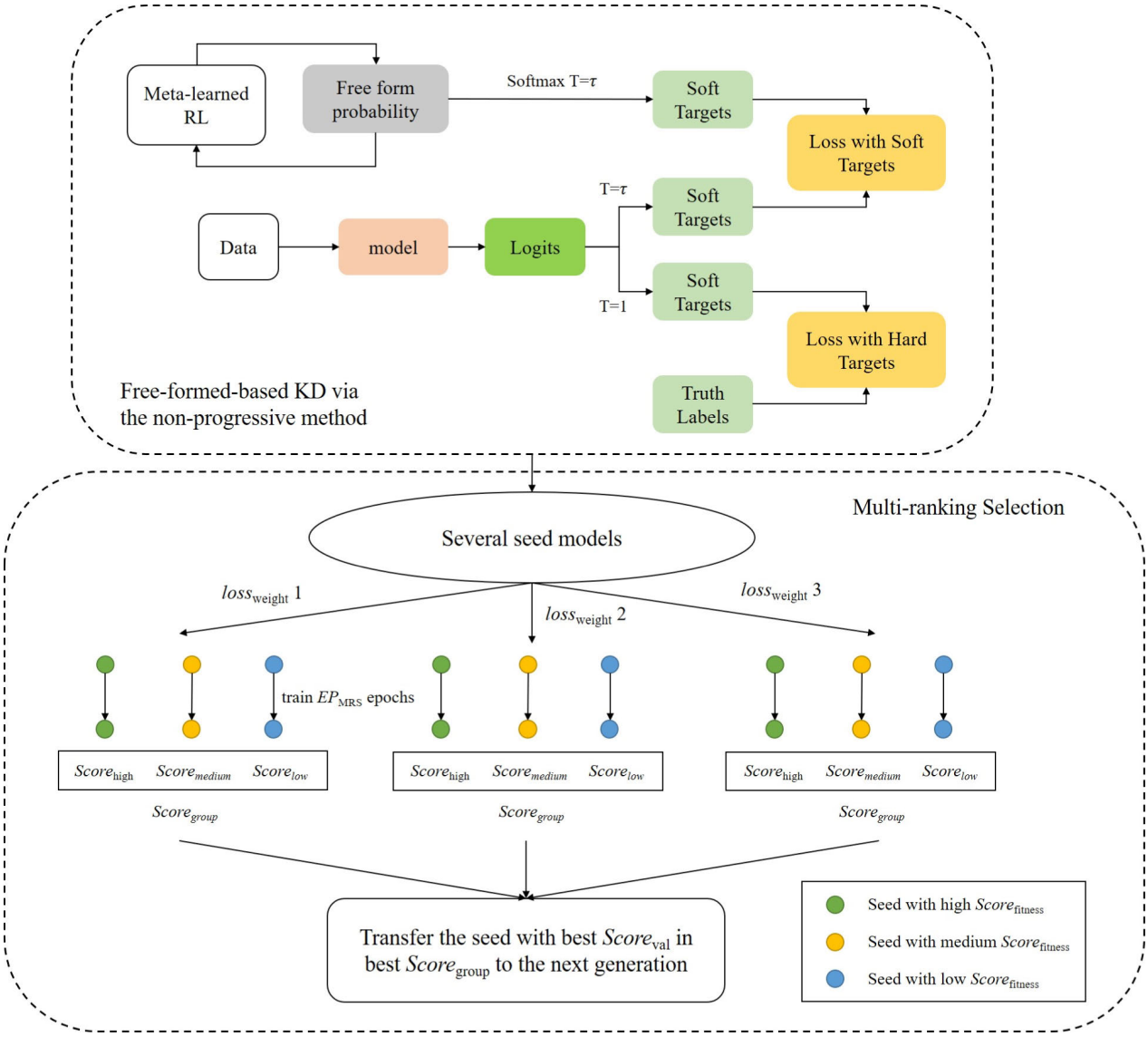



**FIGURE 1.** The illustration of algorithm architecture, including meta-learned free-form non-progressive KD and multi-ranking selection.

Following the initialization practice, the same practice is used when adjusting the target distribution during RL meta-learning non-progressive KD. Five classes surrounding the true class are adjusted while tuning the target distribution, and re-normalization according to (2) is carried out.

The concept of KD is to minimize the loss between soft targets and hard targets. Soft targets are the predictions of the teacher and student. The teacher here is our free-form distribution, and the hard targets correspond to the truth labels. The loss function of our TFKD method is

$$L = (1-\alpha)\,H(s,t) + \alpha D_{\mathrm{KL}}(s, P_\tau) \tag{3}$$

where $s$ is the output probability of the model, $t$ is the truth label, $H$ is the cross-entropy loss, $D_{\mathrm{KL}}$ is KL divergence, $\tau$ is the temperature to soften the free-form distribution, and $\alpha$ is a scaling factor.

### B. DATA PREPROCESS AND STRATIFICATION

In this paper, VGG-8 [39] is used on CIFAR-10 [40], CIFAR-100 [40], and SVHN [41] datasets. In the field of image classification, VGG is one of the most commonly used CNN models. To apply to the framework of KD, we select the smallest VGG8 model as our student model. It can also be compared to the results of current papers, which show that other researchers also use VGG8 for their KD experiments. Taking 5% of the training set into a validation set using a stratified train-validation method to prevent imbalanced training and validation sets, where the number of samples in certain categories is significantly greater than in others. Taking CIFAR-100, for example, this approach ensures that 25 images are chosen from each class.

After the data split procedure, a series of data augmentations are applied to the training set images, aiming to

**TABLE 1. The configuration of data augmentation.**

| Data Augmentation | Parameters |
|---|---|
| RandomCrop | size = 36, padding = 4 |
| CenterCrop | size = 32 |
| RandomHorizontalFlip | p = 0.5 (default) |
| RandomGrayscale | p = 0.1 (default) |
| RandomAutocontrast | p = 0.5 (default) |
| RandomRotation | degrees = 0.5, 5, 10 |
| ColorJitter | brightness=0, contrast=0, saturation=0 (default) |

enhance the diversity of the dataset. To obtain the best configuration of data augmentation, we conducted experiments with 32 combinations of 7 data augmentations, as displayed in Table. 1 by training the baseline model for 120 epochs on CIFAR-100. The configuration is selected with the best validation accuracy as our data augmentation, including random crop, center crop, random horizontal flipping, random grayscale, and random auto contrast. Data augmentation used in CIFAR-10 and SVHN are the same as in CIFAR-100. Eventually, this image is converted into a tensor, and pixel values are normalized using precomputed mean and standard deviation values. By doing the above process, a balanced and representative distribution of images from each class during the training process is maintained.

### C. NON-PROGRESSIVE METHOD

A non-progressive meta-learned RL algorithm is designed based on Fig. 2. The algorithm begins by initializing a model and establishing a free-form loss function tailored for meta-learning purposes. RL exploration factor [42] varies throughout the entire RL process, where the exploration is more likely to occur at the beginning of RL. During training, the agent selects actions based on Q-network. Our goal is to optimize the model's performance over time by adjusting its actions based on the estimated Q-values.

A strategy is applied, including 33 different actions, each corresponding to a specific adjustment in the five classes centering at the true class. The action is a 6-dimensional vector with the number from 0 to 63, and and only 0 to 32 is used in the action representation. The probability distribution will change when the action number is 0 to 31 and not when the action number is 32. Thus, there are a total of 33 possible actions. For instance: $[i, i, c, i, i]$, where $c$ represents the correct class and $i$ represents the incorrect classes adjacent to the correct class. Any adjustment will change all five values in this list where the change of $c$ in one RL step is true class tuning $T_{\text{tc}}$ whose values range from 0.1-2. Normalization at $[i, i, c, i, i]$ will be carried out after probability tuning by RL. The decision is whether each value should be increased or decreased. This leads to 32 different variations in probability distributions. Considering the original unchanged

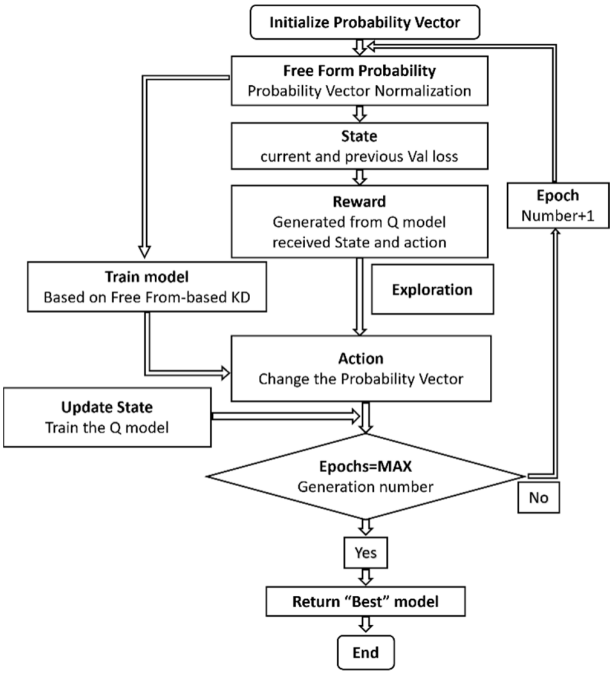



**FIGURE 2. The flowchart of the non-progressive meta-learning method via Reinforcement Learning (RL).**

option, there are 33 possible actions in total. The current state, these 33 actions, and the corresponding rewards are stored in a Q-table and combined with exploration to select the next action. Subsequently, the selected probability in these five classes is used in the continued training afterward. This method is intended to be applied to adjust the target distribution in situ with the goal of enhancing KD performance using a non-progressive strategy where the target distribution does not necessarily gradually converge to the true distribution.

The training process unfolds iteratively, spanning a predetermined number of epochs. During each epoch, the model's parameters are optimized using a combination of training set predictions and a normalized probability vector. Simultaneously, validation loss is monitored to assess the model's performance. Throughout the training process, the algorithm precisely records a range of metrics, including the model's state, RL input, and rewards obtained from actions.

In a subsequent step, selected model is trained using the recorded inputs and rewards as the foundation. Using RL to implement meta-learning non-progressive KD is an innovation of our approach, as it enhances the model's performance by dynamically adapting the target distribution non-progressively. The procedure of our non-progressive meta-learned RL method is summarized in detail in Algorithm 1.

At the training stage, a mechanism is designed for saving the models as the seeds for the MRS method, as described in section II.D. The definition of the model fitness is

$$fitness = \begin{cases} best_{\text{acc}} = (1/val_{\text{acc}}), & if\ best_{\text{acc}} \geq (1/val_{\text{acc}}) \\ best_{\text{loss}} = val_{\text{loss}}, & if\ best_{\text{loss}} \geq val_{\text{loss}} \end{cases} \quad (4)$$

*Model saving*

$$= \begin{cases} True, & if\ (1/val_{\text{acc}}) - best_{\text{acc}} \leq \beta \\ & or\ val_{\text{loss}} - best_{\text{loss}} \leq \beta \\ False, & else \end{cases} \quad (5)$$

where $val_{\text{acc}}$ is validation accuracy, $val_{\text{loss}}$ is validation loss, $\beta$ is a range for model saving. If $(1/val_{\text{acc}})$ or $val_{\text{loss}}$ is a little higher than the fitness, the model will still be saved. $\beta$ will be increased if there is no model saving after 10 epochs. Otherwise, $\beta$ will be decreased.

**Algorithm 1** Non-Progressive Meta-Learned RL Method

**Input:** $model_q$, Probability vector $P_v$, Train Dataset $D_t$, Validation Dataset $D_v$;
**Output:** model;

```
Initialize model weight;
freeform-based loss loss_f;
Val loss loss_v;
P ← P_v/Sum(P_v)
epsilon ← 1
a ← 0.013
for i ← 0 to 240 epochs do
    reward ← model_q.predict(state, action)
    if exploration<epsilon then
        action ← random(0, 32)
    else if exploration<0.1 then
        action ← argmax(reward)
    else
        action ← argmin(reward)
    end if
    epsilon ← epsilon − a
    if epsilon<0.2 then
        epsilon ← 0.2
    end if
    preds ← model.predict(D_t)
    loss ← loss_f(D_t, preds, P)
    Update(model's weight, loss)
    val preds ← model.predict(D_v)
    val loss ← loss_v(D_v, val preds)
    state ← record(val loss)
    RLinput ← record(state)
    All reward ← record(reward)
    Train(model_q, RLinput, All reward)
end for
```

The exploration in RL is tuned by two parameters, i.e., epsilon and $a$. Epsilon, which is set to 1, is higher at first, so the random action is chosen with a higher probability for exploration. Epsilon will decrease by a each epoch, which is set to 0.013 to decrease the probability of exploration. The setting of epsilon and a is according to [42]. The model is trained for 240 epochs via the non-progressive method. First, a non-p

### D. MULTI-RANKING SELECTION METHOD

First, a non-progressive model is trained for $EP_{\text{MRS}} = 20$ as seed candidates. $N = 3$ $loss_{\text{weight}} = [a, b, c]$ is selected, where $a$, $b$, and $c$ are generated randomly. $N$ $loss_{\text{weight}}$ generates $N$ groups, and $Score_{\text{fitness}}$ is calculated with the corresponding $loss_{\text{weight}}$ for each group. In each group, the partition is carried out to form a high, medium, and low sub-group in terms of $Score_{\text{fitness}}$. Afterward, one model is selected from each sub-group. Each model is trained for the additional $EP_{\text{MRS}} = 20$ epochs with the previous RL non-progressive procedure, and $Score_{\text{val}}$ is calculated. Each group has a score $Score_{\text{group}}$ by summing $Score_{\text{val}}$ of the three models selected from each sub-group. The lowest $Score_{\text{group}}$ of $N$ groups is considered to be the most promising candidate, and the corresponding $loss_{\text{weight}}$ will be saved as best $loss_{\text{weight}}$. Then, the model and the corresponding target distribution are selected with the best $Score_{\text{val}}$ from this group as the seed. The seed and the best $loss_{\text{weight}}$ are transferred to the next round in MRS, and the above process is repeated until the end of the entire MRS procedure, which is shown in Fig. 3. Finally, the model is fine-tuned using cross-entropy loss and the best model weights is restored by assessing the validation metrics weighted by the best $loss_{\text{weight}}$. The scores mentioned above are defined as

$$\begin{aligned} Score_{\text{fitness}} = {} & loss_{\text{weight}}[0] \times train_{\text{loss}} \\ & + loss_{\text{weight}}[1] \times (1/train_{\text{acc1}}) \\ & + loss_{\text{weight}}[2] \times (1/train_{\text{acc5}}) \end{aligned} \quad (6)$$

$$Score_{\text{val}} = val_{\text{loss}} + (1/val_{\text{acc1}}) + (1/val_{\text{acc5}}) \quad (7)$$

$$Score_{\text{group}} = Score_{\text{high}} + Score_{\text{medium}} + Score_{\text{low}} \quad (8)$$

where $train_{\text{loss}}$ is training loss, $train_{\text{acc1}}$ and $train_{\text{acc5}}$ are top-1 and top-5 training accuracy, $val_{\text{acc1}}$ and $\text{val}_{\text{acc5}}$ are top-1 and top-5 validation accuracy, $Score_{\text{high}}$, $Score_{\text{medium}}$, and $Score_{\text{low}}$ are $Score_{\text{val}}$ of the three models in sub-group.

## IV. EXPERIMENT

The experiments are conducted on three datasets for image classification: CIFAR-100, CIFAR-10, and SVHN. VGG-8 with most settings similar to CRD [43], [44] is used as the baseline model for all experiments. In this study, python3.7.3, PyTorch 1.13.1, TorchVision 0.14.1, Numpy 1.21.5, SciPy 1.7.3, Scikit-Learn 1.0.2, pandas 1.3.5, and Matlab R2021a are used. The GPU Server we used is Intel Core i9-9900k / Nvidia RTX 2080ti.

### A. CIFAR-100

The baseline and non-progressive methods are conducted for 240 epochs. The baseline is trained with cross-entropy loss while the non-progrssive methods use the loss specified in Eq.(3). SGD is used as an optimizer with a momentum 0.9 and weight decay $5 \times 10^{-4}$. The batch size is set to 64. The initial learning rate is 0.05, multiplied by 0.1 at 150, 180, and 210 of 240 epochs [43], [44]. Afterward, the model

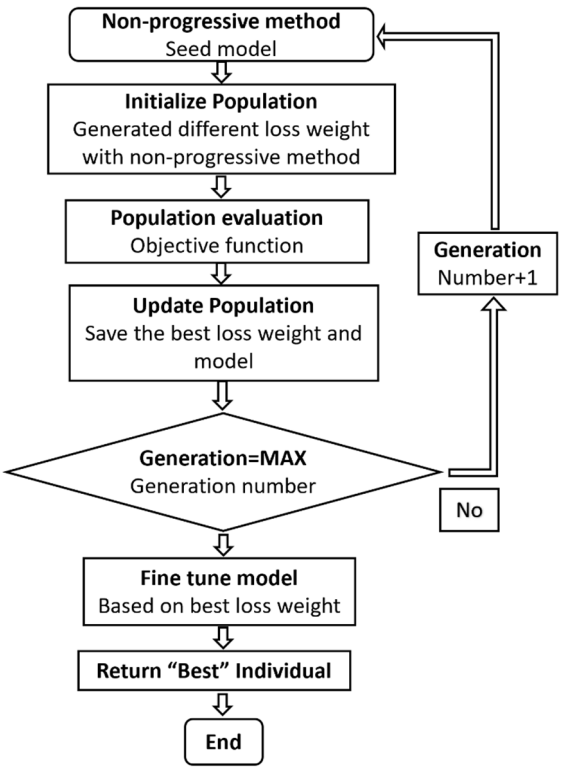



**FIGURE 3.** The flowchart of the Multi-Ranking Selection (MRS) method.

is trained with the MRS method by setting the $EP_{\text{MRS}}$ to 20. The process number is set to 3, and each process has three groups. The temperature $\tau$ for softening the free-form distribution is 20 [45]. We try to find the best $\alpha$ from 0.1 to 0.9 with temperature 20 and select $\alpha$ based on the validation accuracy,which is 0.6.

Table. 2 shows the improvement in top-1 accuracy by our two methods. For example, the non-progressive method improves the model accuracy by 1.62%. In addition, the method with MRS can achieve 1.97% enhancement. It can be seen that all of the joint non-progressive and MRS results are better than just using non-progressive RL meta-learning. For instance, the highest difference between the two methods can be 0.35% when $T_{\text{tc}}$ is set to 1. The magnitude of changing our free-form distribution also has a significant influence. It can be seen that the non-progressive with MRS method, there is a 0.64% difference when the true class tuning ($T_{\text{tc}}$) is set to 0.5 and 2. In Fig. 4, the training accuracy versus the training epoch is plotted, and it is observed that the training accuracy rises faster at the beginning and levels off until the learning rate changes. We can still see a promotion at epoch 150 and 180 when the learning rate becomes smaller.

### B. CIFAR-10 AND SVHN

The experimental settings on CIFAR10 and SVHN are the same as CIFAR-100, except that the training epoch of the baseline and non-progressive method is set to 100. The learning rate is multiplied by 0.1 at the $40^{\text{th}}$ and $80^{\text{th}}$ epoch during the 100-epoch training.

Table. 3 shows that the performance on CIFAR-10 can also be improved by 1.31% and 1.62% with the two methods. The highest accuracy of the two methods is 92.35% and 92.66%, respectively. It can be seen that the method with MRS is still better than only using a non-progressive method. Using a non-progressive method, the difference between different $T_{\text{tc}}$ from 0.1 to 2 can be up to 0.37%.Using joint non-progressive training and MRS, the difference between different $T_{\text{tc}}$ from 0.1 to 2 can be up to 0.46%. On SVHN in Table. 4, the non-progressive and non-progressive with MRS are better than the baseline, with an improvement of 0.87% and 0.69%, respectively. The difference between different $T_{\text{tc}}$ is 0.14% when the tuning is set from 0.1 to 2 by using a non-progressive with MRS method. Nevertheless, the MRS method in SVHN is not as useful as it is in CIFAR-100 and CIFAR-10.

From Fig. 5 and Fig. 6, we can see an obvious promotion when the learning rate gets lower for the first time at epoch 40. The training accuracy increases again when the learning rate decreases for the second time at epoch 80. Except for these two dramatic increases, train accuracy increases with a smooth slope.

### C. COMPARISON RESULTS

Although there is significant success in KD in improving the performance of a small model, there is a limitation in that the student can only mimic the target distribution or the feature maps of the teacher. Conceptually, TFKD methods have more flexibility since they do not require a student to match a teacher. Due to the freedom of learning, TFKD should have more chances to achieve superior results. However, recent research shows that the regular teacher-student KD is always better than TFKD. This result is not in line with our expectations, and a more complex and properly tailored algorithm is needed to improve the training procedure of TFKD. Thus, a non-progressive meta-learned method is investigated in this work to make TFKD comparable to or even exceed teacher-student KD.

At first, we use the Genetic Algorithm (GA) as an optimizer to control the target distribution. However, the result will not be satisfactory if the initial population is not large enough. The large population requires many recalculations of the individual target distribution accuracy after cross-over. This somehow reflects the inefficiency of GA in the current problem of meta-learning model compression. We also tried Simulated Annealing (SA). While the results and the runtime using SA are improved over GA, further improvement is expected when compared to SOTA. This is because SA is a simple optimization algorithm that is more related to random search. Therefore, SA does not allow mutations or significant changes from step to step, which is possible in GA or RL. As far as RL is concerned, it also constructs the objective function surface, and thus, the prediction of the unvisited target distribution becomes possible. Therefore, we change the algorithm to RL, which decides the change of the distribution

**TABLE 2.** Top-1 and top-5 test accuracy (%) on CIFAR-100.

| Method | True class tuning | Training Accuracy | Validation Accuracy | Top-1 Test Accuracy | Top-5 Test Accuracy |
|---|---|---|---|---|---|
| Baseline | X | 98.31 | 70.96 | 70.44 | 90.43 |
| Non-progressive | 0.1 | 94.22 | 71.44 | 71.59(+1.15) | 89.28 |
| | 0.5 | 94.22 | 72.12 | 71.58(+1.14) | 89.35 |
| | 1 | 94.09 | 71.68 | 71.62(+1.18) | 89.14 |
| | 2 | 94.07 | 71.60 | **72.06(+1.62)** | 89.26 |
| Non-progressive with MRS | 0.1 | 93.90 | | 71.79(+1.35) | 89.48 |
| | 0.5 | 93.96 | | 71.77(+1.33) | 89.42 |
| | 1 | 93.83 | | 71.98(+1.54) | 89.37 |
| | 2 | 93.93 | | **72.41(+1.97)** | 89.20 |

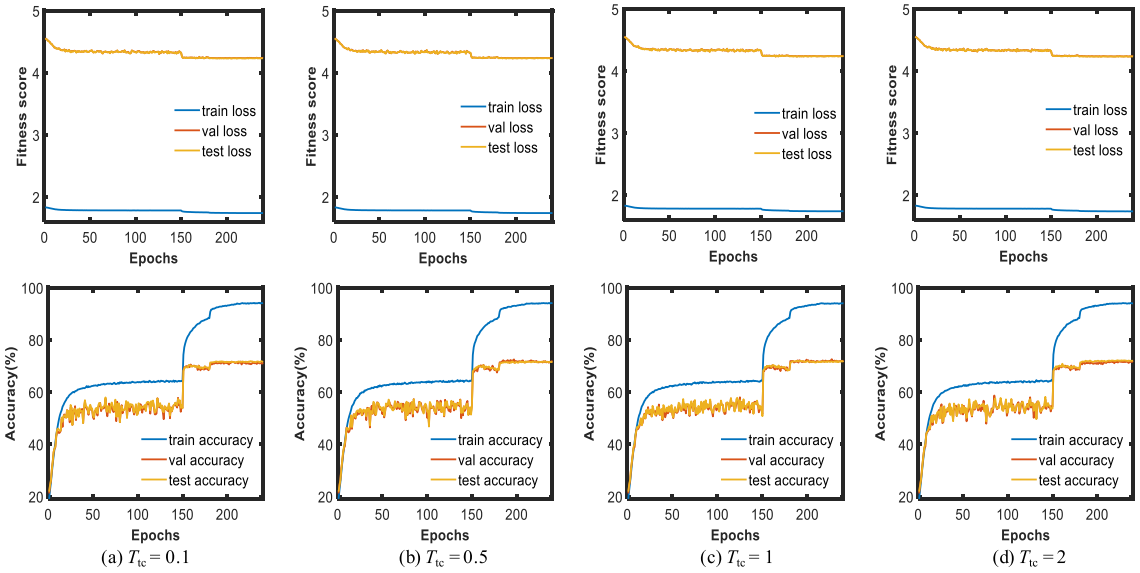



**FIGURE 4.** The training process on CIFAR-100 with different $T_{tc}$.

based on the current Q-table that is only partially trained. Initially, we also designed a method by selecting a proper model every 30 epochs, which can be named multi-restart. Besides, while the RL still selects the best target distribution using the reward mechanism, the selection at the end of 30 epochs will first choose the moderate target distribution. During the model training, the selection gradually shifted toward better individuals at the end of 30 epochs. However, the results with multi-restart are not comparable to those of SOTA. The reason is that the non-progressive RL already has an exploration, especially at the early stage. Difficult convergence can be observed if a non-progressive training method with multi-restart is employed. To solve this problem, we adjust our method by training the model with 240 epochs in the beginning, which accelerates the rate of convergence, and after that, the RL non-progressive training is continued by selecting a model in every 20 epochs. This method, i.e., MRS, is proposed in this paper and achieves higher accuracy. Finally, our TFKD framework is defined with three methods: the non-progressive method, RL, and MRS.

In the following section, we analyze our training process using Fig. 7, Fig. 8, and Fig. 9, and these figures show the action selected at each epoch during RL meta-learning. The results of the true class probability change during RL training in Fig. 10 correspond to Fig. 7, Fig. 8, and Fig. 9. It can be seen the action variation at the beginning is more significant than the variation in the later stage of training because exploration is decreased gradually during RL. The

**TABLE 3.** Top-1 and top-5 test accuracy (%) on CIFAR-10.

| Method | True class tuning | Training Accuracy | Validation Accuracy | Top-1 Test Accuracy | Top-5 Test Accuracy |
|---|---|---|---|---|---|
| Baseline | X | 99.49 | 91.64 | 91.04 | 99.63 |
| Non-progressive | 0.1 | 99.42 | 92.68 | **92.35 (+1.31)** | 99.43 |
| | 0.5 | 99.34 | 92.88 | 91.98(+0.94) | 99.49 |
| | 1 | 99.40 | 92.08 | 92.32(+1.28) | 99.48 |
| | 2 | 99.41 | 92.80 | 92.10(+1.06) | 99.49 |
| Non-progressive with MRS | 0.1 | 99.74 | | 92.48(+1.44) | 99.47 |
| | 0.5 | 99.74 | | 92.20(+1.16) | 99.52 |
| | 1 | 99.74 | | **92.66(+1.62)** | 99.51 |
| | 2 | 99.76 | | 92.25(+1.21) | 99.49 |

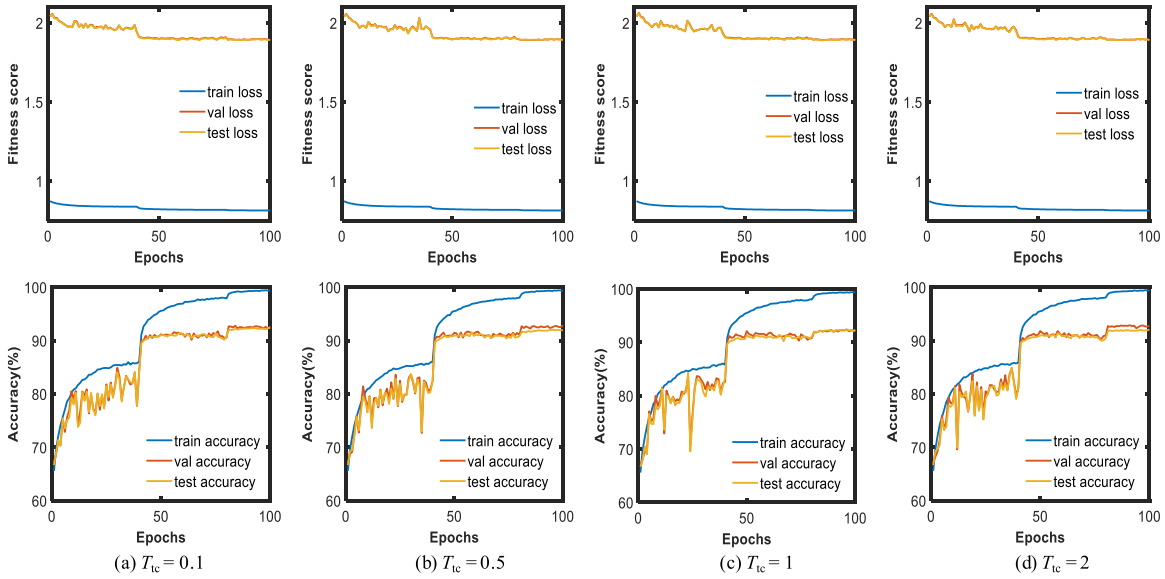
**TABLE 4.** Top-1 and top-5 test accuracy (%) on SVHN.

| Method | True class tuning | Training Accuracy | Validation Accuracy | Top-1 Test Accuracy | Top-5 Test Accuracy |
|---|---|---|---|---|---|
| Baseline | X | 99.22 | 95.11 | 95.44 | 99.52 |
| Non-progressive | 0.1 | 98.50 | 95.69 | 96.29(+0.85) | 99.53 |
| | 0.5 | 98.54 | 95.50 | 96.26(+0.82) | 99.53 |
| | 1 | 98.48 | 95.63 | 96.19(+0.75) | 99.48 |
| | 2 | 98.53 | 95.66 | **96.31(+0.87)** | 99.45 |
| Non-progressive with MRS | 0.1 | 99.39 | | 96.07(+0.63) | 99.46 |
| | 0.5 | 99.40 | | 96.03(+0.59) | 99.50 |
| | 1 | 99.42 | | **96.13(+0.69)** | 99.49 |
| | 2 | 99.42 | | 95.99(+0.55) | 99.50 |

RL selects the best action based on the best reward in the later stage of training. The reason for designing this mechanism is to reduce the rate of convergence and avoid the local optima.

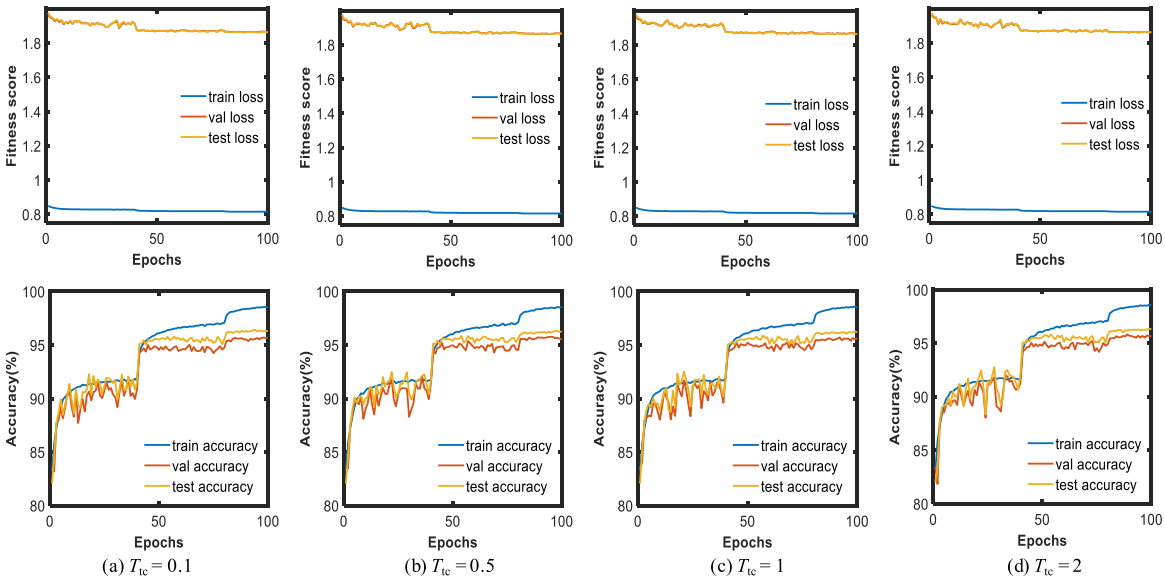
The probabilities of the five classes surrounding the true class are changed during RL. The magnitude of the change is from 0.1 to 2, as described in the Table. 2, Table. 3, and Table. 4. Fig. 10 shows that the probability changes randomly at the beginning and becomes stable at the end, which means the probability of the true class will continue to become high or lower after the training epochs are large enough when the RL Q-Table is adequately filled and when the model training is close to convergence. The converged true class probability values are very similar for the four $T_{tc}$ magnitudes in Fig. 10 using CIFAR-10. On the other hand, CIFAR-100 and SVHN datasets do not show similar true class converged probability values. In Fig. 10, it can be seen that the setting of $T_{tc}$ magnitude has a significant impact on the variation of the true class probability during training. When $T_{tc}$ is set to 2, the true class probability variation is the largest. The true class probability variation is very smooth when $T_{tc}$ is equal to 0.1 while the variation is increased when $T_{tc}$ is increased.

From the results on CIFAR-100 in Table. 2 and CIFAR-10 in Table. 3, the lowest accuracy occurs at $T_{tc} = 0.5$ for both the cases w/ or w/o MRS. Unlike CIFAR-100 and CIFAR-10, the lowest accuracy occurs at $T_{tc} = 2$ for the cases w/ MRS and at $T_{tc} = 1$ for the cases w/o MRS on SVHN. Regarding the best performance, $T_{tc} = 2$ leads to the highest accuracy among all cases on CIFAR-100 using MRS. For CIFAR-10 in Table. 3, the best result among all cases is generated at $T_{tc} = 1$ using MRS. For SVHN in Table. 4, the best result among all cases is generated at $T_{tc} = 2$ without using MRS. We can



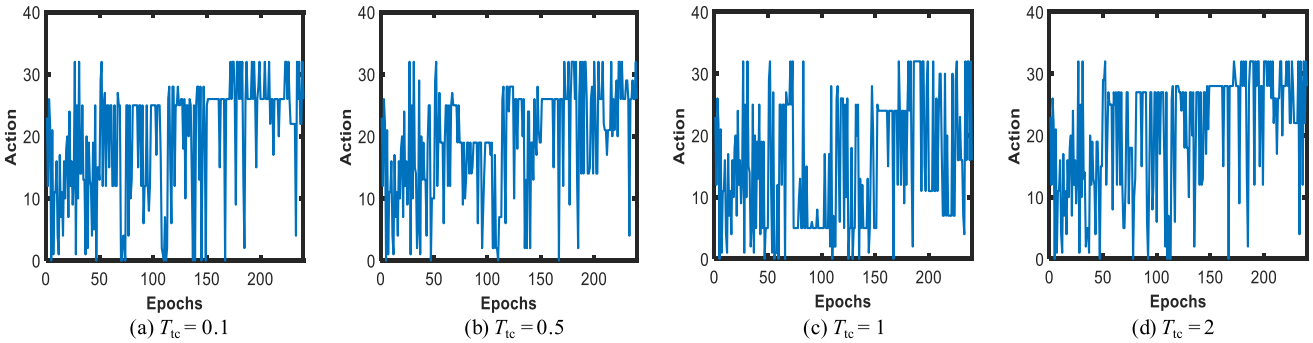

(a) $T_{tc} = 0.1$ (b) $T_{tc} = 0.5$ (c) $T_{tc} = 1$ (d) $T_{tc} = 2$

**FIGURE 5.** The training process on CIFAR-10 with different $T_{tc}$.




(a) $T_{tc} = 0.1$ (b) $T_{tc} = 0.5$ (c) $T_{tc} = 1$ (d) $T_{tc} = 2$

**FIGURE 6.** The training process on SVHN with different $T_{tc}$.




(a) $T_{tc} = 0.1$ (b) $T_{tc} = 0.5$ (c) $T_{tc} = 1$ (d) $T_{tc} = 2$

**FIGURE 7.** The selected actions of RL on CIFAR-100.

observe that different $T_{tc}$ have different influences on each dataset.

Our MRS method is visualized in the scatter plot in Fig. 11, Fig. 12, and Fig. 13. Each process possesses three groups,

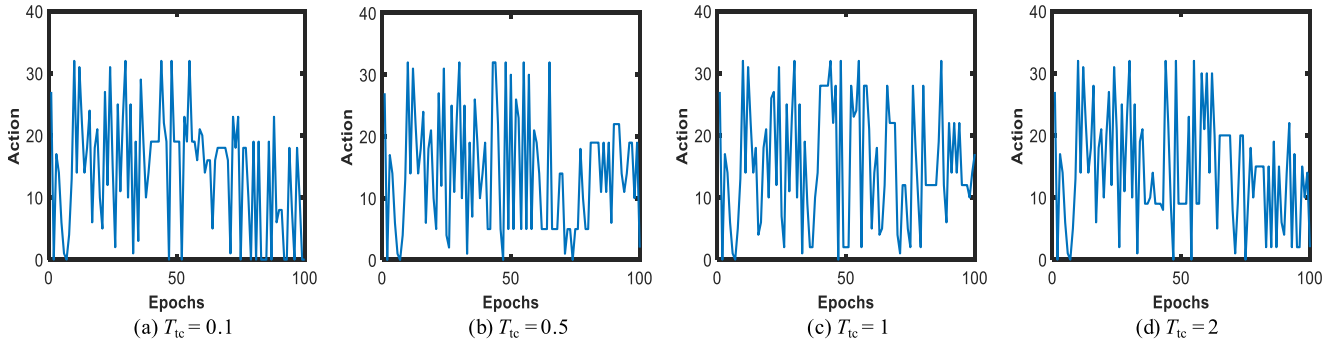



**FIGURE 8.** The selected actions of RL on CIFAR-10.

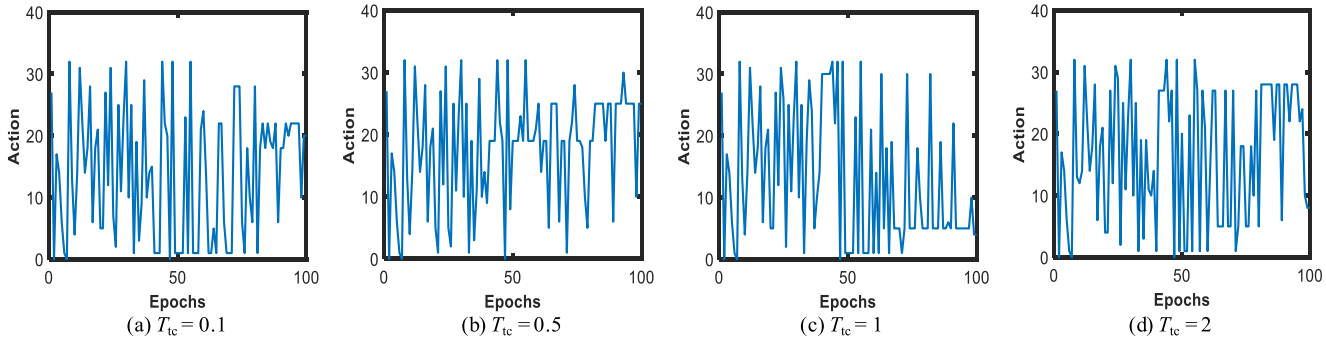



**FIGURE 9.** The selected actions of RL on SVHN.

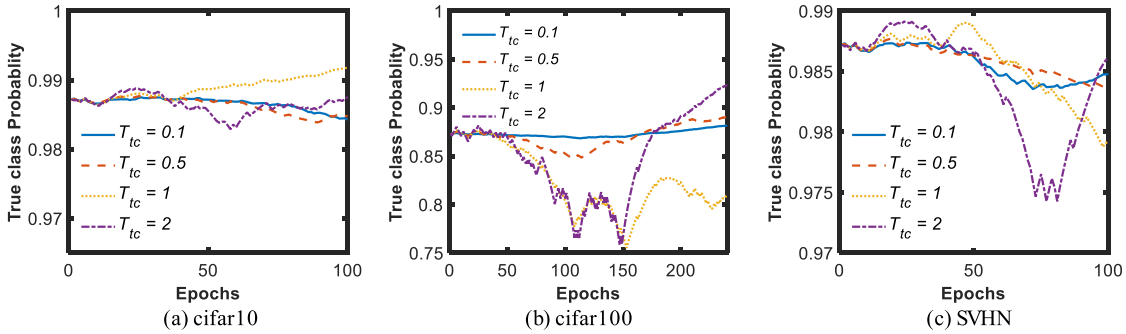



**FIGURE 10.** The change of the true class probability during RL meta-learning.

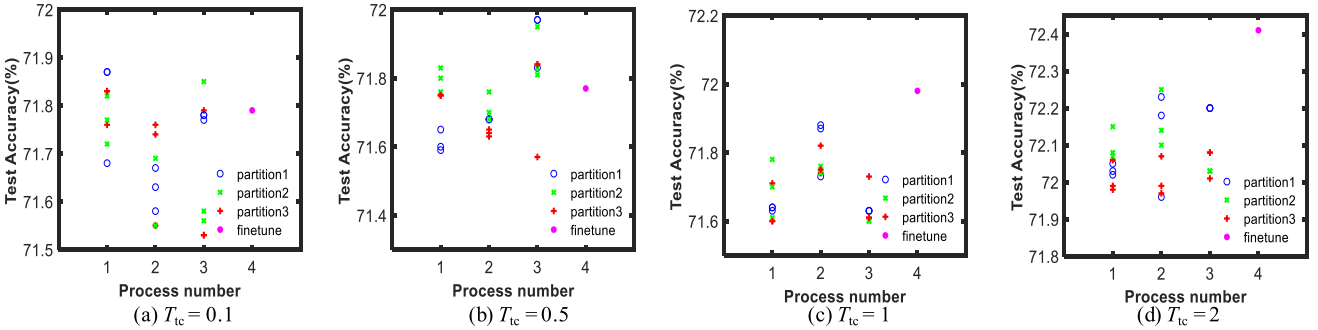



**FIGURE 11.** Results of MRS on CIFAR-100. Fine-tune of the model on the 4$^{th}$ process.

and each group possesses three models. Thus, there are nine dots in each process. After training for three processes, we fine-tuned the model. From Fig. 11, it can be seen that the performances of the models in each process are very different. Since the diversity of the models is ensured by MRS, it has more chances to find the most potential model for continued training. The final model is selected from all of the previously trained models at the intermediate steps.

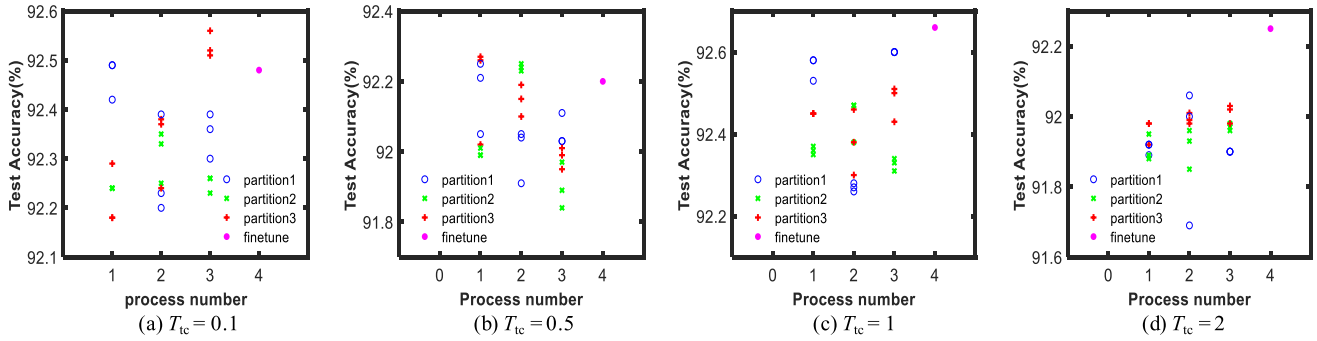



**FIGURE 12.** Results of MRS on CIFAR-10. Fine-tune of the model on the 4th process.

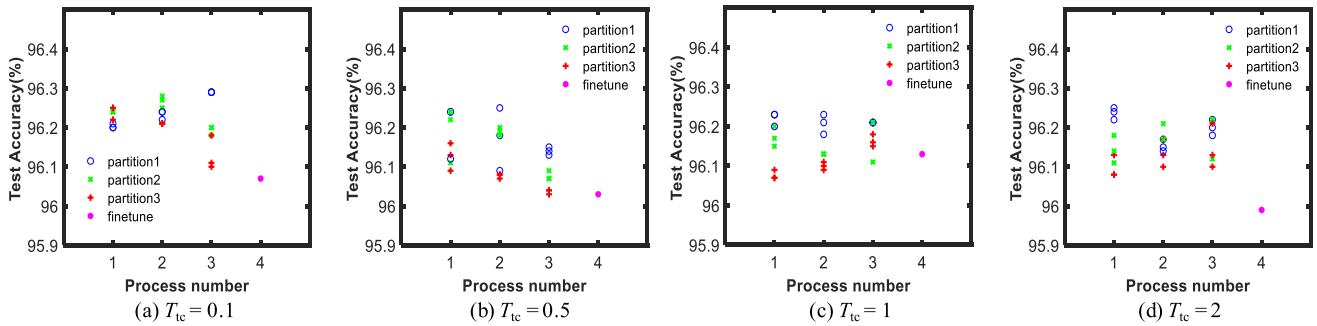



**FIGURE 13.** Results of MRS on SVHN. Fine-tune of the model on the 4th process.

In CIFAR-100 and CIFAR-10, the models after MRS and fine-tuning are always the selected final models. On the other hand, MRS and fine-tuning do not show much strength in SVHN, and the final selected models are normally from the first 100 epochs.

In Table. 5, we show various teacher-free or self-KD methods. Compared to these SOTA methods currently, our method with both non-progressive and MRS can achieve the highest accuracy by 72.41% on CIFAR-100, which exceeds other methods significantly.

### D. DISCUSSION

It is worth to mention the definition of the target distribution. The encoding of the phenol is always a concern in optimization problems. In the literature, most of the current efforts use parameters instead of probability distribution to tune the target distribution in the KD problems. This work utilizes a larger searching space, and a freeform target distribution is made possible with each class probability can be tuned independently. An overly large optimization space, including an extensive range and many optimization variables, always leads to a less converged optimization where the results tend to be settled in the local extremum. To circumvent this problem, a reduced class number can be used in the optimization by selecting fewer key classes to tune their probability in target softened KD. Here, we choose to vary the probabilities of the five classes surrounding the true class in each epoch during KD, while other class probabilities are affected by normalization. In addition, the accuracy-1 and accuracy-5 are used during the selection procedure in the seed model saving and MRS to amplify the significance of the true class or the classes surrounding the true one.

**TABLE 5.** Comparison of the performance using the VGG-8 model on the CIFAR-100 dataset against several State-Of-The-Art (SOTA) methods.

| Method | | Acc1 (%) |
|---|---|---|
| Teacher free | Dropout[31] | 70.52 |
| | DropBlock[32] | 70.76 |
| | SAD[33] | 70.72 |
| | LS[35] | 70.87 |
| | Tf-KD[36] | 71.05 |
| | Tf-FD[30] | 71.62 |
| | **Our method** | **72.41** |
| Self-KD | CS-KD[25] | 71.26 |
| | BYOT[24] | 70.88 |
| | ONE[23] | 72.01 |

Nowadays, a majority of the effort is in the KD based on teacher-student configuration. The efforts mainly focus on looking for a better teacher in terms of the final trained student accuracy or a better teacher-student distillation scheme. The literature has shown that multiple teachers [46], multiple students [47], and adding teaching assistants [48] that somehow bring up the idea of progressive KD [49] and target softened KD, and degraded teachers all contribute to a better trained student model. On the other hand, the training procedure in KD is also investigated in the literature, but the extent seems to be less compared to the effort of forming a proper teacher-student pair. TFKD or Self-KD is an effort of this kind. In our perspective, the training procedure should be further expanded in order to fully utilize the KD strength. Even with the same teacher, different training paths can lead to different results. The non-progressive KD seems to make less sense, but it provides the chance to get around the accuracy bottleneck inherited in progressive KD. This is due to the fact that the complex model and dataset can lead to nonlinear convergence and searching space. Thus, progressive KD does not always give the best results. An algorithm-controlled scheme is needed to tackle the problem of low convergence in non-progressive KD effectively. While many different optimization algorithms can be used, it is found that Reinforcement Learning (RL) can be effective, and Genetic Algorithm (GA) is less effective based on our study. On the other hand, Simulated Annealing (SA) is moderate. The difference between RL meta-learning and meta-learning using conventional optimization algorithms such as GA and SA is that the conventional algorithms do not require training or fitting the model. A Q-table is always required in RL, and the trained network approximating the Q-table is updated at each step in RL. Essentially, the scheme leads to lower optimization efficiency at the early stage since the Q-Table entries are not accurate. Restarting from the RL starting point in each episode helps overcome the problem. Nevertheless, we do not employ this scheme in this work. Instead, the improvement is observed using a large initial stage RL exploration and MRS along the training path. The advantage of model-based RL-based meta-learning is that it builds a prediction model, i.e., intermediate accuracy surface in KD, and thus, even for unvisited points in the searching space of the target distribution, some prediction can be made based on the previous experience. Comparing GA to SA, the computation loading is much greater in GA since it requires a tenth of individuals in each generation. In contrast, SA only steps on one point in the annealing process, which is more similar to the direct or pattern search problem. In our numerical test, SA performs better than GA does. The reason can be that the overly large non-progressive searching space requires an even larger population size to see the effect or that the GA crossover scheme dilutes the valuable information of the desired intermediate target distribution. A different encoding scheme in the phenol has to be used to remedy this obstacle in GA non-progressive KD. Since RL-based meta-learning shows a superior result, we do not explore this aspect in GA further.

## V. FUTURE WORK

A future effort can be made in the meta-learn controlled approximate state in RL and a meta-learn MRS. The key usefulness and strength of meta-learning is that controlling the procedure by humans can be less effective, time-consuming, and require repeated effort in a new problem. Therefore, the meat-learning has emerged as a promising new direction in nearly all machine-learning fields. Specifically, the approximate state of RL is a key to its success. Essentially, the need for approximate states is also due to the fact that overly large search space is highly undesired. In RL model compression, the RL states representing a model can contain a large number of variables, including all of the weights and biases in the model. A reduced set of state variables is desired, but the control should be made adaptively by the algorithms in KD. Secondly, the MRS should also be considered by an algorithm. It can be difficult for humans to know which partially trained, intermediate student models have more potential. In this scheme, the meta-learn can be implemented either in situ or by a further partitioned train set.

## VI. CONCLUSION

In conclusion, a novel, non-progressive, meta-learned, MRS KD is utilized here to boost the accuracy of the student model. With proper setup in the RL, the trained accuracy of the student model is 72.41% in CIFAR-100, VGG-8. To compare with the SOTA results in the literature, we have to point out that the highest accuracy achieved to date in TFKD or Self-KD is 72.01% in CIFAR-100, VGG-8. While most of the KD efforts in Internet Of Things (IOT) and edge computing model compression are teacher-student configuration methodologies, in recent years, TFKD has emerged as a promising alternative by providing more flexibility since no teacher needs to be formed. The training procedure becomes the key in TFKD or Self-KD, and a very flexible training scheme is used where the non-progressiveness of the training procedure, in terms of the target distribution, is formed. In fact, the non-progressiveness can be established not only in the target distribution but also in other aspects of KD, such as hyperparameters. The method's effectiveness here lies in the non-progressive KD exploring a large search space with a proper algorithm control to balance the accompanied hazard of less convergent training. The entire target softening procedure can more effectively pull up the student model accuracy by adjusting the intermediate model training with an algorithm-selected intermediate target distribution. In addition, the saved models using Multi-Ranking Selection (MRS) can overcome the problem of early convergence or being trapped in the undesired location in the search space during KD. The MRS scheme is essential since non-progressiveness in KD can lead to an elevated risk of converging into undesired points even if it also provides more chance of

locating better extrema. We believe the current work contributes to the new knowledge and methodology of KD in edge computing and model compression.

## REFERENCES


[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[2] K. Nan, S. Liu, J. Du, and H. Liu, "Deep model compression for mobile platforms: A survey," *Tsinghua Sci. Technol.*, vol. 24, no. 6, pp. 677–693, Dec. 2019.

[3] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," presented at the Int. Conf. Learn. Represent., Toulon, France, Apr. 24, 2017.

[4] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5058–5066.

[5] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[6] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," 2016, *arXiv:1606.07947*.

[7] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," presented at the Int. Conf. Learn. Represent., May 9, 2015.

[8] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Stat*, vol. 1050, p. 9, Jan. 2015.

[9] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4793–4801.

[10] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3962–3971.

[11] B. Peng, X. Jin, D. Li, S. Zhou, Y. Wu, J. Liu, Z. Zhang, and Y. Liu, "Correlation congruence for knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5006–5015.

[12] C. Yang, L. Xie, S. Qiao, and A. L. Yuille, "Training deep neural networks in generations: A more tolerant teacher educates better students," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5628–5635.

[13] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[14] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction method," *Pacific J. Optim.*, vol. 9, pp. 167–180, 2013.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," presented at the Int. Conf. Learn. Represent. San Juan, Puerto Rico: Caribe Hilton, May 4, 2016.

[16] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. Int. Conf. Learn. Represent.*, 2018, p. 5668.

[17] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 535–541.

[18] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2654–2662.

[19] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5191–5198.

[20] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," 2017, *arXiv:1707.01219*.

[21] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9155–9163.

[22] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisit knowledge distillation: A teacher-free framework," 2019, *arXiv:1909.11723*.

[23] X. Zhu and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 7517–7527.

[24] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3712–3721.

[25] S. Yun, J. Park, K. Lee, and J. Shin, "Regularizing class-wise predictions via self-knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13873–13882.

[26] M. Ji, S. Shin, S. Hwang, G. Park, and I.-C. Moon, "Refine myself by teaching myself: Feature refinement via self-knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10659–10668.

[27] K. Kim, B. Ji, D. Yoon, and S. Hwang, "Self-knowledge distillation with progressive refinement of targets," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6547–6556.

[28] X. Hu, S. Wen, and H. K. Lam, "Dynamic random distribution learning rate for neural networks training," *Appl. Soft Comput.*, vol. 124, Jul. 2022, Art. no. 109058.

[29] J. Park, S. Kim, D. W. Nam, H. Chung, C. Y. Park, and M. S. Jang, "Free-form optimization of nanophotonic devices: From classical methods to deep learning," *Nanophotonics*, vol. 11, no. 9, pp. 1809–1845, May 2022.

[30] L. Li, "Self-regulated feature learning via teacher-free feature distillation," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 347–363.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[32] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 10727–10737.

[33] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection CNNs by self attention distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1013–1021.

[34] J. Wang, P. Zhang, Q. He, Y. Li, and Y. Hu, "Revisiting label smoothing regularization with knowledge distillation," *Appl. Sci.*, vol. 11, no. 10, p. 4699, May 2021.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[36] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3902–3910.

[37] B.-R. Jiang, "Construction of compact device model and model compression based on machine learning," M.S. thesis, Inst. Electron. Eng., Nat. Yang Ming Chiao Tung Univ., Hsinchu, Taiwan, 2023.

[38] P. H. Ho, B. R. Jiang, and A. S. Lin, "Teacher-free knowledge distillation based on non-progressive meta-learned simulated annealing," in *Proc. Int. Conf. Mach. Intell.*, 2024.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, vol. 2011, no. 5, p. 7.

[42] A. Lin, Y.-W. Feng, H.-T. Hsu, H.-C. Tung, C.-H. Huang, H.-L. Liu, and P. Yu, "A meta-learning reinforcement training method for machine learning image-to-image optical proximity correction," *engrXiv*, 2023, doi: 10.31224/3197.

[43] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in *Proc. Int. Conf. Learn. Represent.*, 2019, Art. no. 10699.

[44] Y. Tian, D. Krishnan, and P. Isola. (2019). *Contrastive Representation Distillation*. [Online]. Available: https://github.com/HobbitLong/RepDistiller

[45] L. Yuan, F. E. Tay, T. W. G. Li, and J. Feng. (2019). *Revisit Knowledge Distillation: A Teacher-Free Framework*. [Online]. Available: https://github.com/yuanli2333/Teacher-free-Knowledge-Distillation

[46] H. Zhang, D. Chen, and C. Wang, "Confidence-aware multi-teacher knowledge distillation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 4498–4502.

[47] D. Y. Park, "Learning student-friendly teacher networks for knowledge distillation," in *Proc. NeurIPS*, vol. 34, 2021, pp. 13292–13303.

[48] W. Son, J. Na, J. Choi, and W. Hwang, "Densely guided knowledge distillation using multiple teacher assistants," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9375–9384.

[49] W. Shi, Y. Song, H. Zhou, B. Li, and L. Li, "Follow your path: A progressive method for knowledge distillation," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2021, pp. 596–611.

**BING-RU JIANG** received the B.S. degree from the Department of Photonics, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2021. He is currently pursuing the master's degree with the Institute of Electronics Engineering, National Yang Ming Chiao Tung University. The interest of his master's studies includes compact device models, machine learning, and the application of machine learning in semiconductor manufacturing.

**PIN-HSUAN HO** received the B.S. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2020. Currently, she is pursuing the M.S. degree with the Institute of Electronics, National Yang Ming Chiao Tung University, Taiwan. Her current research interests include machine learning and model compression.

**SYUAN-TING HE** received the B.S. degree in biomedical engineering and environment sciences from National Tsing-Hua University, Hsinchu, Taiwan, in 2021. He is currently pursuing the M.S. degree in artificial intelligence with the National Yang Ming Chiao Tung University, Taiwan. His research interest includes the application of machine learning in model compression.

**ALBERT S. LIN** (Member, IEEE) received the B.S. degree in electronics engineering from National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 2005, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, Michigan, in 2007 and 2010, respectively. He was an Assistant Professor with National Yang Ming Chiao Tung University, in 2011, where he is currently a Professor. His current research interests include machine learning, semiconductor process, and intelligent manufacturing. He receives the Rackham Fellowship from the University of Michigan, in 2005.