


ORIGINAL PAPER

Cross-layer knowledge distillation with KL divergence and offline ensemble for compressing deep neural network

HSING-HUNG CHOU,¹ CHING-TE CHIU^{1,2} AND YI-PING LIAO² 

Deep neural networks (DNN) have solved many tasks, including image classification, object detection, and semantic segmentation. However, when there are huge parameters and high level of computation associated with a DNN model, it becomes difficult to deploy on mobile devices. To address this difficulty, we propose an efficient compression method that can be split into three parts. First, we propose a cross-layer matrix to extract more features from the teacher's model. Second, we adopt Kullback Leibler (KL) Divergence in an offline environment to make the student model find a wider robust minimum. Finally, we propose the offline ensemble pre-trained teachers to teach a student model. To address dimension mismatch between teacher and student models, we adopt a 1×1 convolution and two-stage knowledge distillation to release this constraint. We conducted experiments with VGG and ResNet models, using the CIFAR-100 dataset. With VGG-11 as the teacher's model and VGG-6 as the student's model, experimental results showed that the Top-1 accuracy increased by 3.57% with a $2.08 \times$ compression rate and $3.5 \times$ computation rate. With ResNet-32 as the teacher's model and ResNet-8 as the student's model, experimental results showed that Top-1 accuracy increased by 4.38% with a $6.11 \times$ compression rate and $5.27 \times$ computation rate. In addition, we conducted experiments using the ImageNet 64×64 dataset. With MobileNet-16 as the teacher's model and MobileNet-9 as the student's model, experimental results showed that the Top-1 accuracy increased by 3.98% with a $1.59 \times$ compression rate and $2.05 \times$ computation rate.

Keywords: Deep convolutional model compression, Knowledge distillation, Transfer learning

Received 1 June 2021; Revised 27 September 2021

1. INTRODUCTION

Recently, the area of deep learning is booming owing to the availability of high computation GPGPU and ability to process massive data. Many state-of-the-art performances have been achieved with deep learning for different tasks, including image classification [1], object detection [2], and semantic segmentation [3], and new tasks such as iterative reconstruction [4] and depth estimation [5]. However, when neural networks become deeper and wider, the complexity of deep neural network (DNN) models grows rapidly. Consequently, DNN models cannot practically work with a large number of parameters and high levels of computation, notably for Internet of Things (IoT) devices and self-driving car (Autonomous car).

There are five approaches to achieve a compact yet accurate model—frugal architecture, pruning, matrix

decomposition, quantization, and specialist knowledge distillation (KD). KD is researched to understand how to train a student deep neural network (S-DNN) by learning from teacher deep neural network (T-DNN). In general, S-DNN does not have the same layers as T-DNN; thus, it is difficult to train to find the optimization point. There are two branches of KD approaches. One is the conventional approach, referred to as offline methods [6–11], which train the T-DNN first, then use the S-DNN to mimic the pre-trained T-DNN. Although this two-phase approach incurs considerable computation time, we get a better performing S-DNN. Sometimes, this S-DNN is better performing than the pre-trained T-DNN, because the T-DNN is already pre-trained and has more layers than the S-DNN, while the S-DNN has a better initial weight. The other approach is referred to as online methods [12–14], which start both as scratch models that train together. This one-phase approach expects to train a better model than when only training the S-DNN model. Compared with the offline methods, these online methods do not need to train a T-DNN first, so there is less training time. However, in this research, we are focused on how to get a better performing S-DNN model with the same compression rate. Because most of

¹Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan

²Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

Corresponding author:
Hsing-Hung Chou
Email: paul8301526@gmail.com

the conventional methods showed better results in experiments, we decided to pick the first method in this paper.

After considering the training method, what is taught from the pre-trained model as knowledge to the S-DNN model is very sensitive to its performance. To address this, FSP [6] proposed to use the correlation between input and output feature maps of the layer module, in the form of a Gramian matrix.

In this work, our contributions are as follows:

- 1) We propose a cross-layer matrix to extract more knowledge and add Kullback Leibler (KL) Divergence and offline ensemble to improve image classification with the same compression.
- 2) We propose 1×1 convolutional layers to tune the channels of T-DNN to be identical to those of S-DNN to solve the constraints of our proposed method.
- 3) We propose using two-step KD to improve image classification when there is a huge difference in layers between T-DNN and S-DNN, to avoid the loss of KD.
- 4) Our method can be used not only for image classification tasks [15, 16] but also for other tasks, such as object detection [17, 18], semantic segmentation [19], and action estimation [20] in videos.

II. RELATED WORK

The compression methods can be divided into four categories, knowledge distillation, pruning, low-rank decomposition, and quantization.

A) Knowledge distillation

Hinton *et al.* [7] distilled knowledge from a very large teacher model to promote a small student model by using a softened softmax of a teacher network. The rationale is to take advantage of extra supervision provided by the teacher model during the training of the student model, beyond a conventional supervised learning objective such as the cross-entropy loss subject to the training data labels. Romero *et al.* [8] proposed “Hint training” to train partial layers, then used the final output layers to train the student model to enhance the performance. Park *et al.* [11] proposed to transfer mutual relations of data examples. Yim *et al.* [6] transferred knowledge from the T-DNN to the S-DNN as output feature maps rather than as layer parameters. They become a certain layer group in the network and define the correlation between the input and output feature maps of the layer group as a Gram matrix so that the feature correlations of the S-DNN and T-DNN become similar. We expand the Gram matrix by adding more than cross-one layer. Furthermore, approach taken by Lee *et al.* [10] is based on the correlation between two feature maps as knowledge by using Singular Value Decomposition (SVD). However, this approach [10] will cost a significant computation time because feature maps that are needed to be decomposed by CPU. As a result, we take FSP [10] model as a baseline to enhance as our proposed methods.

Moreover, while earlier distillation methods often take an offline learning strategy that need two phases of the training procedure, the more recently proposed deep learning [12] method overcomes this limitation by conducting an online distillation in one-phase training between two peer student models. We will add KLDivergence in our proposed method, which makes it different than online methods. Anil *et al.* [13] extend [12] to decrease the training time of large-scale distributed neural networks. Lan *et al.* [14] present an On-the-fly Native Ensemble (ONE) learning strategy for one-stage online distillation. However, existing online methods lack a strong “teacher” model, which limits the efficacy of knowledge discovery.

In [21], Wang and Yoon provide a comprehensive survey on the recent progress of KD methods together with S-T frameworks typically used for vision tasks and systematically analyze the research status of KD in vision applications. KDGAN consisting of a classifier, a teacher, and a discriminator is proposed in [22]. The classifier and the teacher learn from each other via distillation losses and are adversarially trained against the discriminator via adversarial losses. From the concrete distribution, continuous samples are generated to obtain low-variance gradient updates, which speed up the training. To efficiently transmit extracted useful teacher information to the student DNN, Bae *et al.* propose to perform bottom-up step-by-step transfer of densely distilled knowledge [23].

B) Deep neural network compression and efficient processing

Scientists found that network pruning can be used not only to reduce network complexity but also to prevent overfitting. An old method [24] to pruning was the Biased Weight Decay. Han *et al.* [25] first proposed that it’s peaceful to remove neurons with zero input or output connections from the neural network. By using L1/L2 regularization, some of the weights converged to zeros after training. As a result, by using the combination of pruning, quantization, and Huffman coding [26], the compression of AlexNet can reach $35\times$. CLIP-Q [27] flexibly makes weight pruning choices that can adapt to compress the DNN during the training time. Apart from weight pruning, there is another pruning approach, named channel pruning, that assesses neuron importance. Li *et al.* [28] computed the importance of each filter by calculating its absolute weight sum. Furthermore, Hsiao *et al.* [29] measured the significance of each filter by calculating the largest singular value.

In [30], Deng *et al.* provide a comprehensive survey on reviewing the mainstream compression approaches such as compacted model, tensor decomposition, data quantization, and network sparsification to compress DNN without compromising accuracy. In [31], Sze *et al.* present a tutorial and survey on understanding the key design for DNN and evaluating different DNN hardware implementations with benchmarks in order to achieve processing efficiency.

C) Low-rank decomposition

Low-rank approximation [32–35] approaches have been widely studied. However, low-rank approximation is inconvenient because each decomposition of feature maps is computationally expensive. Moreover, the methods of low-rank approximation only consider a few layers; therefore, they cannot consider the compression of the whole network.

D) Quantization

Quantization is a method for reducing the number of bits for weight and bias of each layer. We can divide methods either by using auxiliary data [36, 37], or not using auxiliary data [38–40]. Additionally, there are two research approaches in compressing on bit-level. One is to use fixed-point implementation, and the other is to use common quantization methods, for example, K-means and scalar quantization. For fixed-point implementation, Hwang and Sung [39] proposed a design with ternary weights, 3-bit signals, and an optimization process which was done by back-propagation-based re-training. For the other approach, Ji *et al.* [40] designed a supervised iteration quantization to reduce the bit resolution of the weights. They applied K-means-based adaptive quantization methods, such as vector quantization using K-means, product quantization, residual quantization, and discussed the efficacy of their design on compressing deep convolutional networks.

E) Deep learning

Special issues on deep learning framework architectures, hardware acceleration, DNN over the cloud, fog, edge, and end devices are elaborated [41]. In addition, methods and applications especially emphasizing on exploring recent advances in perceptual applications are addressed and discussed [41]. In [42], Wang *et al.* present to learn a proper prior from data for adversarial autoencoders. The notion of code generators is presented to transform manually selected simple priors into ones that can better characterize the data distribution.

III. PROPOSED ARCHITECTURE

The core idea of KD is how to define the vital information, then transfer the knowledge from the T-DNN to the S-DNN. As a result, we will divide our approach into four parts. Section A shows what the knowledge in T-DNN will transfer and its mathematical expression and the definition loss term L_{KD} . Moreover, we will add another loss function L_{KL} between the prediction of T-DNN and S-DNN in Section B. Furthermore, we will use offline ensemble pre-trained T-DNNs to teach one student in Section C. Subsequently, the overall loss function will be discussed in Section D. Finally, we will discuss the constraints of our proposed method and solutions. The overall architecture for our three proposed compression methods are shown in Fig. 1.

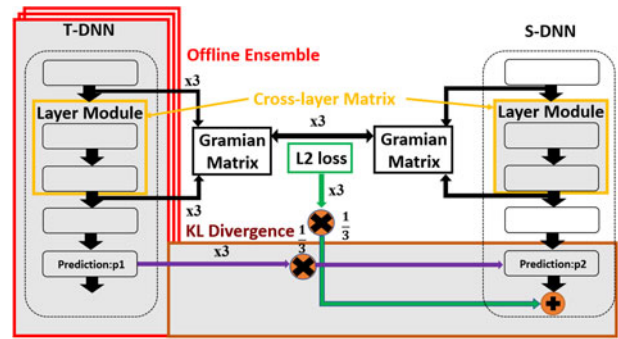


Fig. 1. Overall architecture of our proposed methods. There are three parts of our architecture. First, we propose cross-layer matrix to exact more features by FSP [6] adopting the proposed Gramian matrix in the orange part. Second, we adopt the KL Divergence in the offline environment to make S-DNN find a wider robust minimum in the brown part. Finally, we propose the use of offline ensemble pre-trained T-DNN to teach a S-DNN by using stochastic mean in the red part.

A) Cross-layer matrix

1) PROPOSED DISTILLED KNOWLEDGE

Yim *et al.* [6] proposed “FSP” by using Gramian matrix to mimic the generated features of the T-DNN, which can be a hard constraint for the S-DNN. Based on [6], we generate more Gramian matrices by crossing more than one layer. The numbers of cross matrices we add as loss function depends on how many layer modules are in DNN model. We believe that with more Gramian matrices in loss function, it would make the S-DNN get better performance.

The reason why we use the Gramian matrix created by feature maps is that we believe that instead of teaching the right answer to the student, it is better to teach the solution procedure to the student. Imagine there is a classroom, a teacher is teaching a student with a math question. It is better to teach how to use a formula first and the solution procedure than to provide the correct answer directly.

2) MATHEMATICAL EXPRESSION OF THE KNOWLEDGE DISTILLATION

Based on FSP [6], the Gramian matrix can be defined by two output feature maps. We propose the Gramian matrix as the knowledge to transfer. The Gramian matrix $G \in \mathbb{R}^{m \times n}$ is generated by the features from two layers. One output feature map is defined as $F^1 \in \mathbb{R}^{h \times w \times m}$, where h , w , represent the height and width of output feature maps and m represents the number of output channels. The other output feature map is defined as $F^2 \in \mathbb{R}^{h \times w \times n}$. Then, the Gramian matrix $G \in \mathbb{R}^{m \times n}$ is calculated by (1)

$$G_{i,j}(x; W) = \sum_{s=1}^h \sum_{t=1}^w \frac{F_{s,t,i}^1(x; W) \times F_{s,t,j}^2(x; W)}{h \times w}, \quad (1)$$

where i, j represent the points of cross-one-layer results, x represents the input image and W are the weights of the network model. Unlike FSP [6], we select several points not only from cross-one module layer but also from cross-more-than-one module layer to generate more Gramian

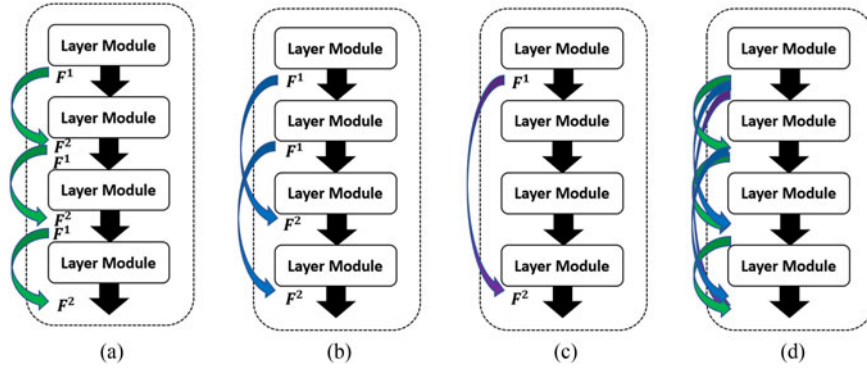


Fig. 2. (a) Cross one layer. (b) Cross two layers. (c) Cross three layers. (d) Our proposed.

matrices as shown in (2) and (3).

$$G_{i,q}(x; W) = \sum_{s=1}^h \sum_{t=1}^w \frac{F^1_{s,t,i}(x; W) \times F^2_{s,t,q}(x; W)}{h \times w}, \quad (2)$$

$$G_{i,r}(x; W) = \sum_{s=1}^h \sum_{t=1}^w \frac{F^1_{s,t,i}(x; W) \times F^2_{s,t,r}(x; W)}{h \times w}, \quad (3)$$

where i, q represent the points of cross-two-layer results as shown in Fig. 2(b) and i, r represent the points of cross-three-layer results as shown in Fig. 2(c). In Fig. 2, we see there are three different kinds of cross-layer matrix. Our proposed method is to combine all the Gramian matrix 2(d) as knowledge.

3) KD LOSS FOR THE GRAMIAN MATRIX

As discussed previously, the T-DNN will teach S-DNN the solution of question by using the Gramian matrix. We assume that there are B Gramian matrices $G_b^T, u = 1, \dots, B$, which are generated by the T-DNN, and B Gramian matrices $G_b^S, i = 1, \dots, B$, which are generated by the S-DNN. Next, each pair of Gramian matrices will be calculated as the cost function by using the squared L2 norm. The cost function of knowledge distillation $L_{KD}(W_t; W_s)$ is defined as (4):

$$L_{KD}(W_t; W_s) = \frac{1}{B} \sum_x \sum_{b=1}^B \lambda_i \times \|G_b^T(x; W_t) - G_b^S(x; W_s)\|_2^2, \quad (4)$$

where λ_i represents the weight for each KD loss and B represents the numbers of Gramian matrices. Because our proposed method adds more Gramian matrices by creating the cross matrices, we initially set all KD losses with the same weight. As a result, the values of λ_i are identical in our experiments.

B) KL Divergence

We propose using KL Divergence, which was used in DML [12], as our second-order loss function. In contrast to the online method [12] with two-direction learning, our offline method is only used in one direction from T-DNN to

S-DNN. Given D as the data examples $X = \{x_n\}_{n=1}^D$ from C classes, we represent the corresponding label set as $Y = \{y_i\}_{i=1}^C$ with $y_i \in \{1, 2, \dots, C\}$. The probability of class c for data example x_n is given by a neural network θ_1 and computed as

$$p_1^C(x_n) = \frac{\exp(z_1^C)}{\sum_{c=1}^C \exp(z_1^C)}, \quad (5)$$

where $p_1^C(x_n)$ represents the probability distribution of θ_1 and the logit z_1^C is the output of the “softmax” layer in θ_1 . As a result, the formulation of KL Divergence can be computed as

$$L_{KL}(p_T||p_S) = \sum_{d=1}^D \sum_{c=1}^C p_T^c(x_d) \log \frac{p_T^c(x_d)}{p_S^c(x_d)}, \quad (6)$$

where $L_{KL}(p_T||p_S)$ represent the probability distribution of teacher and student model. We believe that the student model can get full of knowledge from teacher model by having distribution similar to teacher’s distribution.

C) Offline ensemble

The original method of FSP [6] is discussed with one T-DNN to transfer one S-DNN. Compared with FSP [6], we propose using offline ensemble pre-trained teachers to generate the stochastic mean and improve the image classification result. The cost functions of knowledge distillation and KL Divergence are defined as

$$L_{KD}^{Ensemble} = \frac{1}{K} \sum_{k=1}^K L_{KD,k}, \quad (7)$$

$$L_{KL}^{Ensemble} = \frac{1}{K} \sum_{k=1}^K L_{KL}(p_k||p_S), \quad (8)$$

where $L_{KD}^{Ensemble}$ represents the loss function of offline ensemble knowledge distillation, $L_{KL}^{Ensemble}$ represents the loss function of offline ensemble KL Divergence, K represents the numbers of pre-trained teacher models ($K=3$). We believe that the offline ensemble pre-trained teacher models with the same architecture, but the different weights will transfer knowledge to student model by using the stochastic mean.

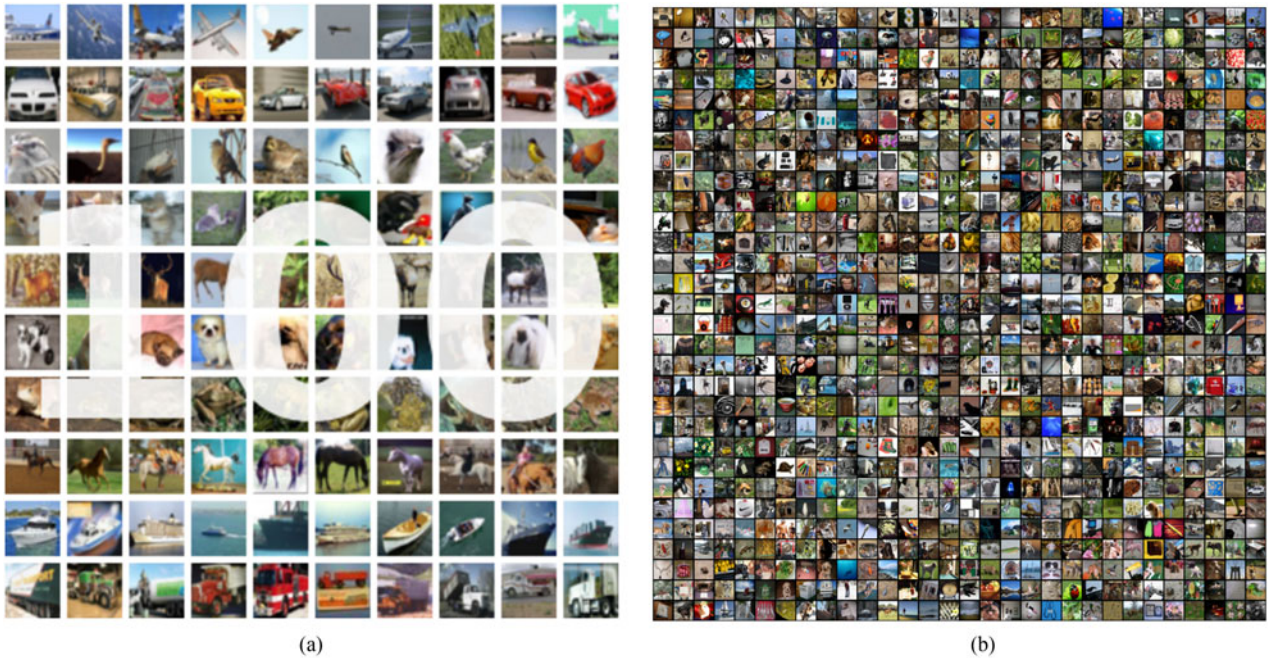


Fig. 3. (a) CIFAR-100. (b) ImageNet64*64.

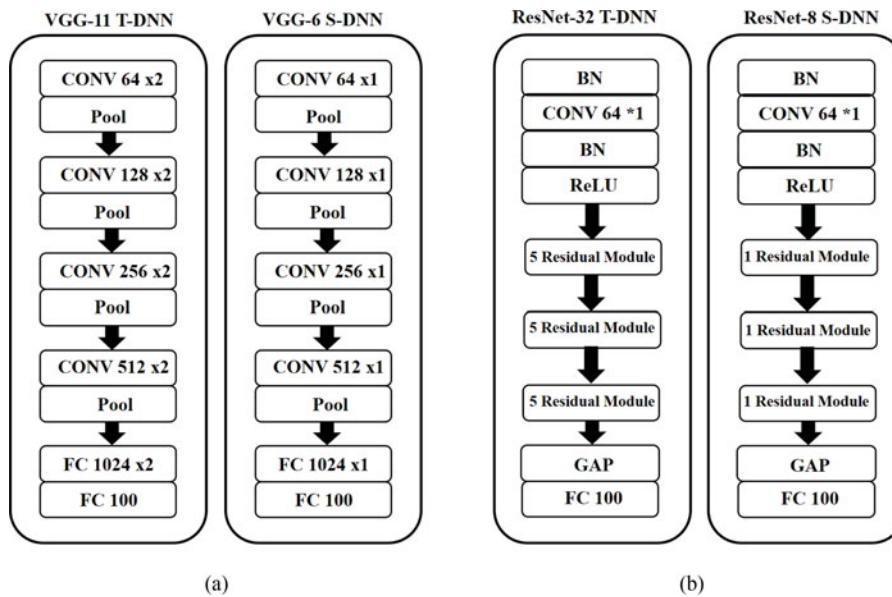


Fig. 4. T-DNN and S-DNN of the VGG and ResNet models. T-DNN: VGG-11 and ResNet-32. S-DNN: VGG-6 and ResNet-8.

D) Overall loss function

We had already proposed L_{KD} , L_{KL} , and stochastic mean for our method. Hence, the overall loss function $L_{total}(\theta_1)$ for training S-DNN is shown as (9)

$$L_{total}(\theta_1) = L_{CE}(\theta_1) + \frac{1}{K} \sum_{k=1}^K L_{KL}(p_k || p_s) + \frac{1}{K} \sum_{k=1}^K L_{KD,k} \tag{9}$$

with the objective function of multi-class image classification $L_{CE}(\theta_1)$ to train the network θ_1 is defined as the cross entropy error between the predicted values and the

correct labels:

$$L_{CE}(\theta_1) = - \sum_{d=1}^D \sum_{c=1}^C I(y_d, c) \log(p_1^c(x_d)), \tag{10}$$

with an indicator function I defined as

$$I(y_i, m) = \begin{cases} 1, & y_i = c \\ 0, & y_i \neq c \end{cases} \tag{11}$$

To prevent $L_{KD,k}$ larger than $L_{CE}(\theta_1)$ from inducing gradient explosion, we will adopt gradient clipping [43] to limit the

gradient of knowledge distillation $\nabla(\theta_1)_{KD}^{clipped}$ during training procedure as shown in Equation (12):

$$\nabla(\theta_1)_{KD}^{clipped} = \begin{cases} \beta \times \nabla(\theta_1)_{KD}, & \nabla(\theta_1)_{KD} < \nabla(\theta_1)_{CE} \\ \nabla(\theta_1)_{KD}, & \text{otherwise,} \end{cases} \quad (12)$$

$$\beta = \frac{1}{1 + \exp(-\tau + p)}, \quad (13)$$

$$\tau = \frac{\|\nabla(\theta_1)_{CE}\|_2}{\|\nabla(\theta_1)_{KD}\|_2}, \quad (14)$$

where β is a sigmoid function. In Equation (13), p means the current epoch of training. Furthermore, the L_2 -norm ratios are the L_{CE} and $L_{KD,k}$ in Equation (14). Hence, the rich knowledge distilled from T-DNN can be transferred knowledge S-DNN without worrying about gradient explosion.

IV. EXPERIMENTAL RESULTS

In this section, we will evaluate our proposed compression method with two datasets and three different models. The two datasets are the familiar CIFAR-100 [44] and the rich collection of images, ImageNet64*64 [45], as shown in Fig. 3. Additionally, there are two models, VGG and ResNet, training and testing on CIFAR-100 and one model named MobileNet, training and testing on ImageNet64*64.

A) Environment and datasets

Our proposed method is implemented in TensorFlow [46] with Python 3.5 interference on the computers (CPU: Intel[®] Core[™] i7-7800X @ 3.5 GHZ, main memory: 32 GB DRAM, GPU: NVIDIA GEFORCE[®] GTX 1080).

The CIFAR-100 dataset consists of 60000 images with a size of 32×32 , divided as 50000 training data and 10000 test data, and 100 classes. We used random shift, random rotation and horizontal flip as data augmentations. Our proposed method was tested under the same conditions as FSP [6], and for increasing the dependability of the testing results, we ran the experiments three times and took the average as the final experimental results. We take VGG and ResNet as the DNN to prove that our proposed method works. The T-DNN and S-DNN models are shown in Fig. 4. We picked VGG as our first model because its architecture is very simple and can be implemented quickly. As in SSKD_SVD [10], we defined VGG-11 as T-DNN and VGG-6 as S-DNN. As in FSP [6], we defined ResNet-32 as T-DNN and partially reduced the residual modules to create ResNet-8 as S-DNN.

The ImageNet64*64 dataset consists of about 1.2 million images with a size of 64×64 , divided with about 1.2 million training data and 50 000 test data, and 1000 classes. We used the same data augmentations as same with CIFAR-100 and the experiments were run three times and took the average as the final result. On ImageNet64*64, we defined MobileNet-16 as T-DNN and MobileNet-9 as S-DNN as shown in Fig. 5.

On CIFAR-100, the training procedure for networks was considered by FSP [6] and SSKD_SVD [10]. We set the batch

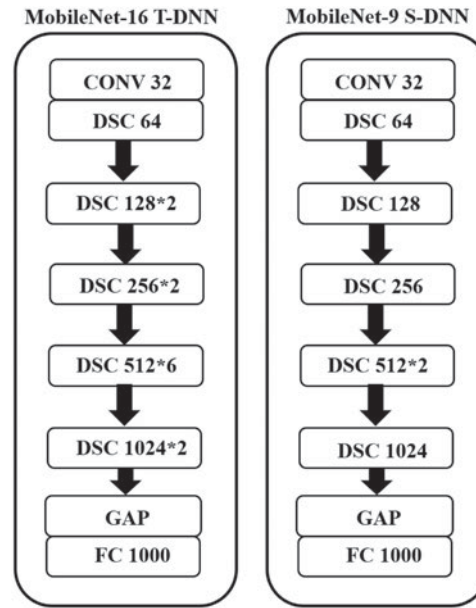


Fig. 5. T-DNN and S-DNN of the MobileNet models. T-DNN: MobileNet-16. S-DNN: MobileNet-9.

size to 128 and the training epochs to 200 during training, optimized the procedure by stochastic gradient descent [47], and adopted Nesterov accelerated gradient [48]. The initial learning rate was set to 10^{-2} and the momentum was set to 0.9. The decay parameter was set to 10^{-4} . The learning rate was reduced to 0.1 per 50 epochs. Additionally, we set the batch size to 64 during training, training epochs to 40, and the learning rate was reduced to 0.1 per 10 epochs for ImageNet64*64.

B) Results

In this section, we show the final results of our proposed method with the computation rate, computation, Top-1 accuracy, and inference time. With VGG-11 as the teacher's model and VGG-6 as the student's model, experimental results show that the student's model increases 0.57% Top-1 accuracy while decreasing 53.9% of parameters and 72.8% computation compared to T-DNN and reducing inference time from 61.6 to 49.8 ms. Furthermore, with ResNet-32 as the teacher's model and ResNet-8 as the student's model, experimental results indicate that the student's model decreases 0.55% Top-1 accuracy by 0.55% while decreasing 83.65% of parameters and 82.6% computation compared to T-DNN and reducing inference time from 115.2 to 51.6 ms. The experimental results of VGG and ResNet are shown in Table 1 and 2, respectively. In most of the offline methods, the training procedure will lose some of their accuracies. However, it is surprising that our proposed method the S-DNN can train even better than T-DNN, as shown in Table 1.

With MobileNet-16 as the teacher's model and MobileNet-9 as the student's model, experimental results

Table 1. Classification results after knowledge distillation (VGG-11->6) on CIFAR-100 dataset.

VGG	Baseline(T-DNN)	Our proposed
Compression	100%	47.9%
Computation	100%	28.2%
Top-1 accuracy	66.10%	66.67%
Inference time[ms]	61.6	49.8

Table 2. Classification results after knowledge distillation (ResNet-32->8) on CIFAR-100 dataset.

ResNet	Baseline(T-DNN)	Our proposed
Compression	100%	16.35%
Computation	100%	17.4%
Top-1 accuracy	69.00%	68.45%
Inference time[ms]	115.2	51.6

Table 3. Classification results after knowledge distillation (MobileNet-16->9) on ImageNet64*64.

MobileNet	Baseline(T-DNN)	Our proposed
Compression	100%	62.6%
Computation	100%	48.7%
Top-1 accuracy	53.72%	49.80%
Inference time[ms]	98.7	52.9

show that the student's model decreases 3.92% Top-1 accuracy while decreasing 37.4% of parameters and 51.3% computation compare to T-DNN and reducing inference time from 98.7 ms to 52.9 ms. The results of MobileNet are shown in Table 3.

C) Ablation

1) CROSS-LAYER MATRIX

The different options of the cross matrix are shown in Fig. 6. Figure 6 (a) represents the "original" method FSP [6]. Figures 6(b) and 6(c) are our proposed methods in VGG and ResNet models. The combination of cross-one layer and cross-two layers is named as "P₁(Cross two layers)". Furthermore, the combination of cross-one layer, cross-two layers, and cross-three layers is named as "P₁(Cross three layers)".

The simulation results of VGG models are shown in Table 4. The result of S-DNN is set as the baseline. Compared with the baseline, the method of "P₁(Cross two layers)" and "P₁ (Cross three layers)" achieves a low performance in the testing result. Additionally, "P₁ (Cross three layers)" have an increase in performance of 0.4% compared with FSP [6]. Subsequently, let us see the deeper architecture ResNet as shown in Table 5. Moreover, the methods of "P₁(Cross two layers)" and "P₁ (Cross three layers)" achieve a low performance in the testing results. The "P₁ (Cross three layers)" have an increase in performance of 0.13% compared with FSP [6].

Additionally, the different choices of the cross-layer matrix are shown in Fig. 7. Figure 7 (a) represents the

Table 4. Different proposed method of cross matrix (VGG-11->6) with CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	66.01%	65.75%	66.54%	66.10%	3.00%
S-DNN{Baseline}	63.06%	62.79%	63.45%	63.10%	0.00%
FSP [6]	64.51%	64.18%	64.53%	64.40%	1.30%
P ₁ (Cross 2 layers)	64.71%	64.79%	64.76%	64.75%	1.65%
P ₁ (Cross 3 layers)	65.25%	64.87%	64.54%	64.80%	1.70%

Table 5. Different proposed method of cross matrix (ResNet-32->8) with CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	68.80%	69.43%	68.79%	69.00%	4.93%
S-DNN{Baseline}	64.34%	63.68%	64.20%	64.07%	0.00%
FSP [6]	65.30%	65.48%	65.65%	65.47%	1.40%
P ₁ (Cross 2 layers)	65.30%	65.49%	65.66%	65.48%	1.41%
P ₁ (Cross 3 layers)	66.00%	65.47%	65.53%	65.60%	1.53%

Table 6. Different proposed method of cross matrix (MobileNet-16->9) with ImageNet64*64. T-DNN: MobileNet-16, S-DNN: MobileNet-9.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	53.48%	53.33%	53.72%	53.51%	7.69%
S-DNN{Baseline}	45.73%	45.86%	45.89%	45.82%	0.00%
FSP [6]	46.38%	46.15%	46.65%	46.39%	0.57%
P ₁ (Cross 2 layers)	49.26%	49.24%	49.16%	49.22%	3.40%
P ₁ (Cross 3 layers)	49.29%	49.29%	49.04%	49.20%	3.38%
P ₁ (Cross 4 layers)	49.35%	48.98%	49.08%	49.12%	3.30%

'original' method FSP [6]. Figures 7(b)–7(d) are our proposed methods with MobileNet model. The combination of cross-one layer and cross-two layers are named as "P₁(Cross two layers)". Additionally, the combination of cross-one layer, cross-two layers, and cross-three layers is termed as "P₁(Cross three layers)". Finally, the combination of cross-one layer, cross-two layers, cross-three layers, and cross-four layers is named as "P₁(Cross four layers)".

The results of MobileNet model are shown in Table 6. Compared with the baseline, the methods of "P₁(Cross two layers)" and "P₁ (Cross three layers)" achieve a low performance in the testing result. Furthermore, "P₁ (Cross three layers)" has increased performance of 3.38% compared with FSP [6].

2) INFLUENCE OF ADDING KL DIVERGENCE

The illustration of adding KL Divergence as our second-order loss function is shown in Fig. 8. We believe that the student's model can get full of knowledge from the teacher's model by being similar to the teacher's distribution. The combination of FSP [6] and adding KL Divergence is named as "P₂ (KL Divergence)". First, let us see the result of VGG models as shown in Table 7. Additionally, the result of S-DNN is set as the baseline. We have two competitors. The first is the proposed Hinton [7] approach and our basis FSP [6]. It is demonstrated that adding KL Divergence yields a

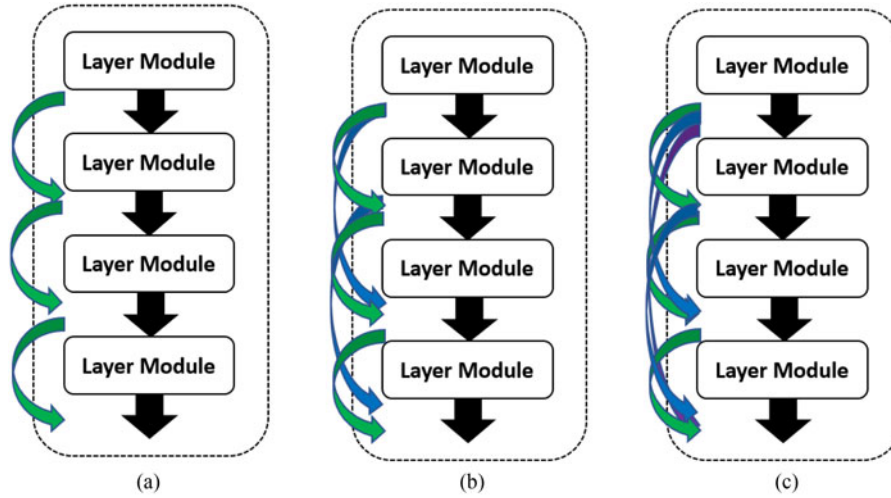


Fig. 6. (a) Cross one layer. (b) Cross two layers. (c) Cross three layers.

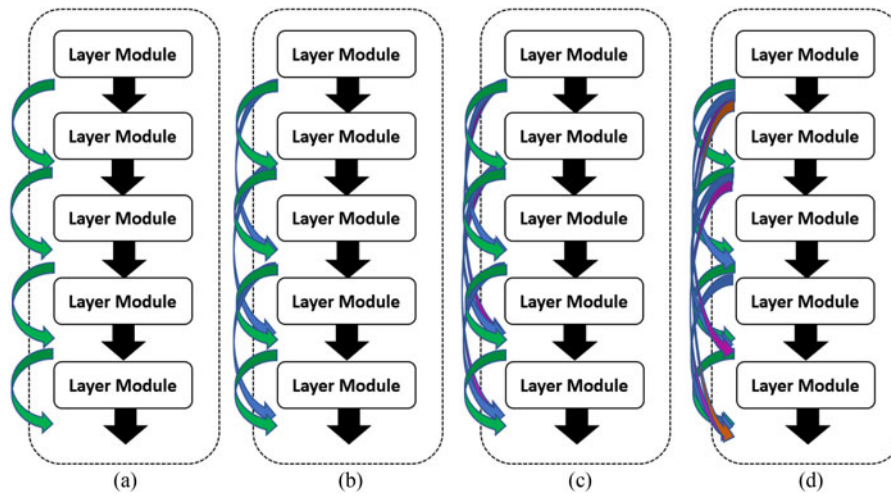


Fig. 7. (a) Cross-one layer. (b) Cross-two layers. (c) Cross-three layers. (d) Cross-four layers.

Table 7. Differential of adding KL Divergence (VGG-11>9) with CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	66.01%	65.75%	66.54%	66.10%	3.00%
S-DNN{Baseline}	63.06%	62.79%	63.45%	63.10%	0.00%
Hinton [7]	64.89%	64.81%	64.84%	64.84%	1.74%
FSP [6]	64.51%	64.18%	64.53%	64.40%	1.30%
P2(KL Divergence)	66.04%	66.09%	66.02%	66.05%	2.95%

Table 8. Differential of adding KL Divergence (ResNet-32->8) with CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	68.80%	69.43%	68.79%	69.00%	4.93%
S-DNN{Baseline}	64.34%	63.68%	64.20%	64.07%	0.00%
Hinton [7]	67.83%	67.97%	67.72%	67.84%	3.77%
FSP [6]	65.30%	65.48%	65.65%	65.47%	1.40%
P2(KL Divergence)	67.12%	67.08%	66.84%	67.01%	2.94%

1.54% increase over the competitor FSP [6]. Furthermore, the experimental results of ResNet are shown in Table 8. It indicates that adding KL Divergence yields a 1.54% increase over the competitor FSP [6].

On ImageNet64*64, the experimental results of MobileNet models are shown in Table 9. We have two competitors, the first is the proposed Hinton [7] approach and our basis FSP [6]. It is shown that adding KL Divergence that obtains a 2.59% increase over the competitor FSP [6].

3) OFFLINE ENSEMBLE

In this section, we will discuss the influence of the number of multiple pre-trained teachers as shown in Fig. 9. Figure 9(a) uses one pre-trained teacher as FSP [6]. Figures 9(b) and 9(c) represent two pre-trained teachers and three pre-trained teachers named as “P₂ (two teachers)” and “P₃ (three teachers)”, respectively. First, let us see the result of VGG model in Table 10. Additionally, the result of S-DNN is set as the baseline. From Table 10, we find that using more pre-trained teachers with stochastic mean can increase the Top-1 accuracy. It is shown that offline ensemble yields a

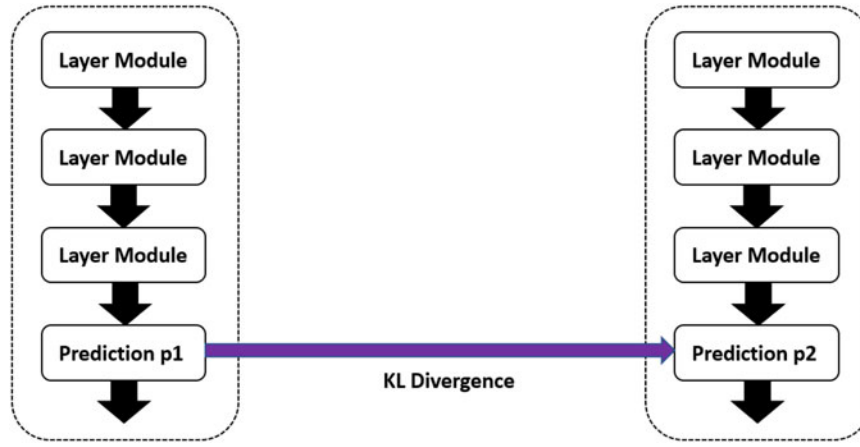


Fig. 8. Illustration of using KL Divergence.

Table 9. Differential of adding KL Divergence (MobileNet-16->9) with ImageNet64*64. T-DNN: MobileNet-16, S-DNN: MobileNet-9.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	53.48%	53.33%	53.72%	53.51%	7.69%
S-DNN{Baseline}	45.73%	45.86%	45.89%	45.82%	0.00%
Hinton [7]	49.23%	49.10%	49.12%	49.15%	3.33%
FSP [6]	46.38%	46.15%	46.65%	46.39%	0.57%
P2(KL Divergence)	48.63%	48.14%	48.48%	48.41%	2.59%

Table 12. Different numbers of teachers (MobileNet-16->9) with ImageNet64*64. T-DNN: MobileNet-16, S-DNN: MobileNet-9.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	53.48%	53.33%	53.72%	53.51%	7.69%
S-DNN{Baseline}	45.73%	45.86%	45.89%	45.82%	0.00%
FSP [6]	46.38%	46.15%	46.65%	46.39%	0.57%
P3(2 teachers)	49.25%	49.52%	49.29%	49.44%	3.2%
P3(3 teachers)	49.13%	49.54%	49.36%	49.34%	3.52%

Table 10. Different numbers of teachers (VGG-11->6) with CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	66.01%	65.75%	66.54%	66.10%	3.00%
S-DNN{Baseline}	63.06%	62.79%	63.45%	63.10%	0.00%
FSP [6]	64.51%	64.18%	64.53%	64.40%	1.30%
P3(2 teachers)	66.15%	65.94%	65.90%	66.00%	2.90%
P3(3 teachers)	66.20%	65.97%	66.02%	66.06%	2.96%

Table 11. Different numbers of teachers (ResNet-32->8) with CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
T-DNN	68.80%	69.43%	68.79%	69.00%	4.93%
S-DNN{Baseline}	64.34%	63.68%	64.20%	64.07%	0.00%
FSP [6]	65.30%	65.48%	65.65%	65.47%	1.40%
P3(2 teachers)	66.27%	66.01%	66.15%	66.14%	2.07%
P3(3 teachers)	66.71%	66.69%	66.96%	66.78%	2.71%

1.66% increase over the competitor FSP [6]. In addition, let us see the deeper model ResNet. From Table 11, we can also find that using more teachers can increase the Top-1 accuracy. It is demonstrated that offline ensemble obtains a 1.31% increase over the competitor FSP [6]. Finally, let us see the result of MobileNet models as shown in Table 12. It is demonstrated that adding KL Divergence yields a 2.59% increase over the competitor FSP [6].

4) COMBINATION OF PROPOSED METHODS

By analyzing prior work, we want to try a combination of proposed methods. The combination of crossing one layer, adding KL Divergence and offline ensemble is shown in Tables 13 and 14.

First, let us see the result of VGG with the combination of crossing three layers, adding KL Divergence and offline ensemble. From Table 13, we can see that by adding the proposed method increases the Top-1 accuracy. It is shown that the combination of proposed method gets an increase of 2.27% than FSP [6]. Second, let us see the deeper ResNet model with the combination of proposed methods. From Table 14, we can see that by adding the proposed method to increase the Top-1 accuracy. It is shown that the combination of proposed method gets an increase of 2.98% than FSP [6]. Finally, from Table 15 we can see that by adding proposed method to increase the Top-1 accuracy, the combination of proposed method get 4.04% increase than FSP [6].

D) Comparison with other work

With the same compression on S-DNN, it can be seen that our proposed method got the state-of-the-art results on VGG and ResNet models compared with the competitors [6, 7, 10, 11]. As we can see in Table 16, the result of our proposed method achieves a 66.67% Top-1 accuracy with a 2.08x compression rate and 3.5x computation rate. Additionally, the result of our proposed method achieves a 68.45% Top-1 accuracy with a 6.11x compression rate and 5.27x computation rate as shown in Table 17. Furthermore, we can see

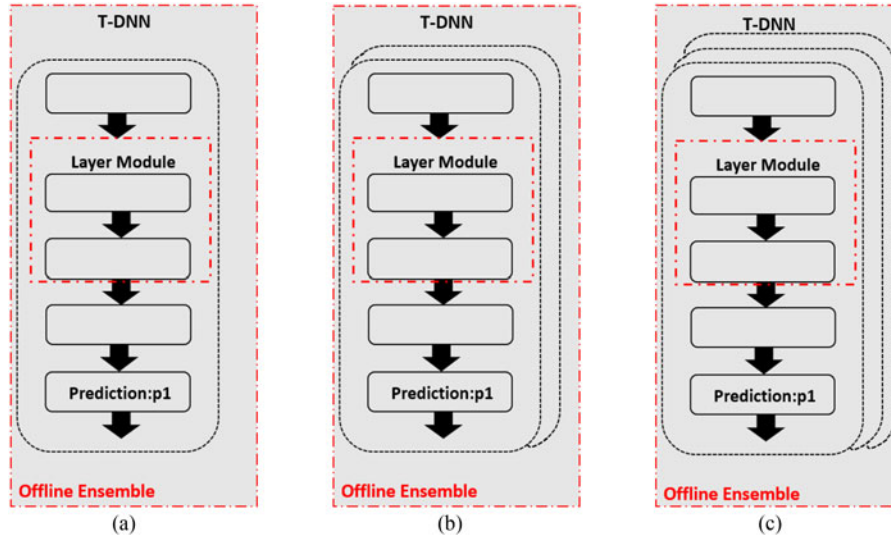


Fig. 9. (a) One pre-trained teacher. (b) Two pre-trained teachers. (c) Three pre-trained teachers.

Table 13. Combination of proposed methods (VGG-11->6) with CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6. P1: cross-three layers. P2: KL Divergence. P3: three pre-trained teachers.

FSP [6] (Baseline)	Cross-layer matrix	KL Divergence	Offline ensemble	Top-1 accuracy	Differential Accuracy
✓				64.40%	0.00%
✓	✓			64.80%	0.40%
✓	✓	✓		66.10%	1.70%
✓	✓	✓	✓	66.67%	2.27%

Table 14. Different proposed method of cross matrix (ResNet-32->8) with CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8. P1: cross-three layers. P2: KL Divergence. P3: three pre-trained teachers.

FSP [6] (Baseline)	Cross-layer matrix	KL divergence	Offline ensemble	Top-1 accuracy	Differential Accuracy
✓				65.47%	0.00%
✓	✓			65.60%	0.13%
✓	✓	✓		66.13%	0.66%
✓	✓	✓	✓	68.45%	2.98%

Table 15. Combination of proposed methods (MobileNet) with ImageNet64*64. T-DNN: MobileNet-16, S-DNN: MobileNet-9. P1: cross-three layers. P2: KL Divergence. P3: three pre-trained teachers.

FSP [6] (Baseline)	Cross-layer matrix	KL Divergence	Offline ensemble	Top-1 accuracy	Differential Accuracy
✓				46.39%	0.00%
✓	✓			49.12%	2.73%
✓	✓	✓		49.60%	3.21%
✓	✓	✓	✓	49.86%	3.47%

Table 16. Computation, parameters, and average Top-1 accuracy comparison with VGG-11 and VGG-6 on CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6.

	FLOPs [M]	Parameters [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	212.8	7.93	66.01%	65.75%	66.54%	66.10%
S-DNN	60.71	3.8	63.06%	62.79%	63.45%	63.10%
Hinton [7]	60.71	3.8	64.89%	64.81%	64.84%	64.84%
FSP [6]	60.71	3.8	64.51%	64.18%	64.53%	64.40%
SSKD_SVD [10]	60.71	3.8	66.51%	66.57%	66.50%	66.52%
RKD [11]	60.71	3.8	62.41%	62.38%	62.33%	62.37%
Our method	60.71	3.8	66.62%	66.60%	66.78%	66.67%

Table 17. Computation, parameters, and average Top-1 accuracy comparison with ResNet-32 and ResNet-8 on CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	FLOPs [M]	Parameters [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	1101.7	7.4	68.80%	69.43%	68.79%	69.00%
S-DNN	191.79	1.21	64.34%	63.68%	64.20%	64.07%
Hinton [7]	191.79	1.21	67.83%	67.97%	67.72%	67.84%
FSP [6]	191.79	1.21	65.30%	65.48%	65.65%	65.47%
SSKD_SVD [10]	191.79	1.21	65.96%	66.32%	66.11%	66.13%
RKD [11]	191.79	1.21	66.84%	66.78%	66.05%	66.55%
Our method	191.79	1.21	68.53%	68.41%	68.41%	68.45%

Table 18. Computation, parameters, and average Top-1 accuracy comparison with ResNet-32 and ResNet-8 on CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	FLOPs [M]	Parameters [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	117.59	2.97	53.48%	53.33%	53.72%	53.51%
S-DNN	57.32	1.86	45.73%	45.86%	45.89%	45.82%
Hinton [7]	57.32	1.86	49.23%	49.10%	49.12%	49.15%
FSP [6]	57.32	1.86	46.38%	46.15%	45.65%	46.39%
SSKD_SVD [10]	57.32	1.86	45.61%	44.91%	44.97%	45.16%
RKD [11]	57.32	1.86	48.98%	48.41%	48.41%	48.79%
Our method	57.32	1.86	49.90%	49.80%	49.90%	49.86%

Table 19. Classification results after knowledge distillation (ResNet-50->10) with CIFAR-100 dataset.

ResNet	Baseline(T-DNN)	Our proposed
Compression	100%	27.81%
Computation	100%	26.84%
Top-1 accuracy	70.47%	73.92%
Inference time[ms]	188.9	66.3

in Table 18 that the result of our proposed method achieves a 49.86% Top-1 accuracy with a 1.59x compression rate and 2.05x computation rate.

V. DISCUSSION

In this section, we show what our proposed methods meet limitations. After that, we will compare solutions with the previous proposed methods and discuss why some of our proposed new ideas make our S-DNN get better performance with the same compression.

A) Constraints of our proposed method

FSP [6] proposed a certain layer group in the network and defined the correlation between input and output feature maps of the layer group as Gramian matrix. The limitation is that the T-DNN and S-DNN should have the same channels, as shown in Fig. 10. If the Gramian matrices of T-DNN and S-DNN do not have the same dimension, $m_T = m_S$ and $n_T = n_S$, then they cannot be calculated as loss function. Furthermore, we find that huge difference layers between T-DNN and S-DNN may cause the difficulty in transferring knowledge, as shown in Fig. 11. Hence, we define “the T-DNN and S-DNN should have the same channels” and “the huge difference layers between T-DNN and S-DNN” as constraint 1 and constraint 2, respectively.

B) Solutions

How to make the Gramian matrices of T-DNN and S-DNN with same dimension is the key to solving constraint 1. As a result, we propose to use 1×1 convolution layers to forcibly tune output channels of T-DNN. In Fig. 12, the color in orange layers are the additional convolutional 1×1 layers. Because the additional convolutional layer only works in the training procedure, the parameters of T-DNN and S-DNN are the same with the original ones. Moreover, we propose two-step compression to solve constraint 2 as shown in Fig. 13. We believe that using “Teacher” T-DNN to teach a temporary neural network model “Temporal”, then use the “Temporal” to teach the final target neural network model “Student”.

Based on our proposed method, we do not have the experiments on greatly reducing the depth of the student model. There are two reasons. First, if we greatly reduce the depth of the student model, the image sizes after passing the student module with reduced depth are different from that of the original student model, the issue of how to align with the teacher model needs to be resolved. Second, to remove the number of Res-blocks to three in some Res-blocks could make the top-1 accuracy drop heavily since the CNN model may not learn enough details from the feature maps.

As a result, we do not change the structure of ResNet. We decrease the student model to the lowest number of the ResNet layers to create the smallest size and computation of the ResNet-10 as shown in Fig. 14.

C) Experiments and results

Using the same environment details of CIFAR-100 datasets, we set the batch size to 64 during training and training epochs to 200. We use ResNet-50 as the teacher’s model and ResNet-10 as the student’s model and ResNet-18 as the

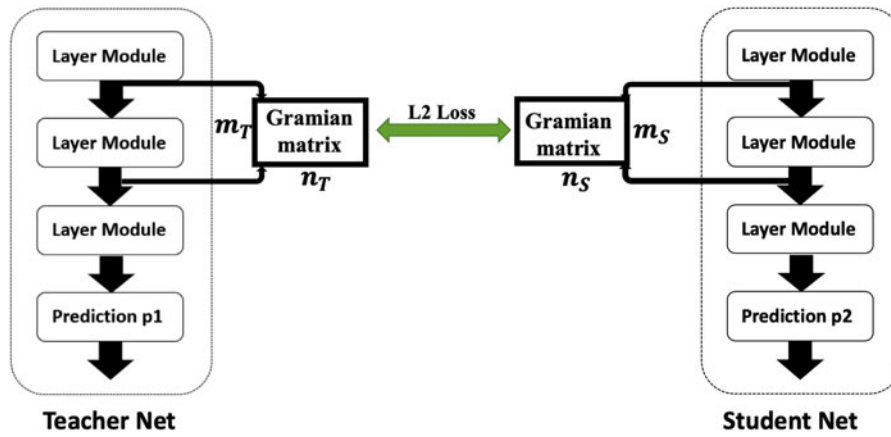


Fig. 10. Limitation of FSP [6]. m_T, n_T represent the dimension of T-DNN Gramian matrix and m_S, n_S represent the dimension of S-DNN Gramian matrix.

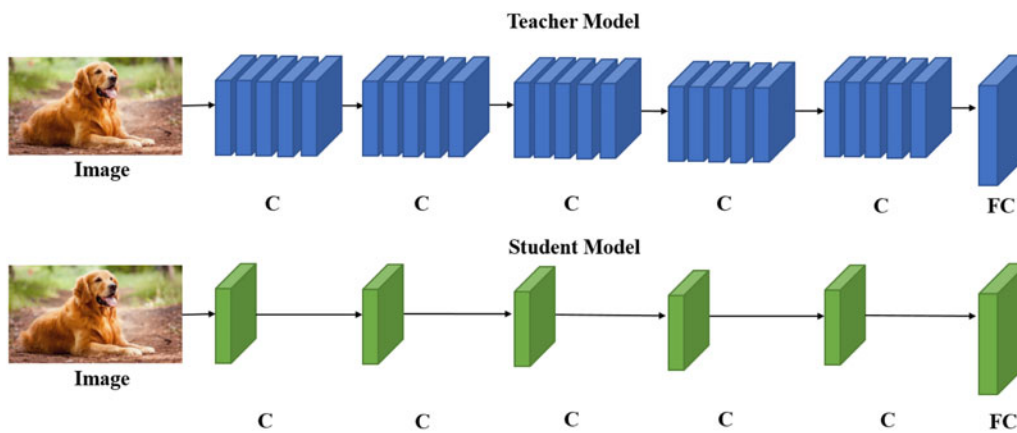


Fig. 11. Illustration of huge layer number difference. C, convolutional layer; FC, fully-connected layer.

temporal model, as shown in Fig. 14. Experimental results show that the student’s model increases the Top-1 accuracy by 3.45% and decreases 72.19% of parameters and 73.16% computation compared to T-DNN and reduces inference time from 188.9 ms to 66.3 ms. The experimental results of ResNet are shown in Table 19. Compared with ResNet-10

in Table 20, by using 1×1 convolutional layers and our proposed method, ResNet-18 can obtain a 6.20% differential accuracy. Furthermore, by using two-stage KD, we could increase the differential accuracy by a further 6.54%. We believe that using two-stage knowledge distillation can prevent the loss of KD.

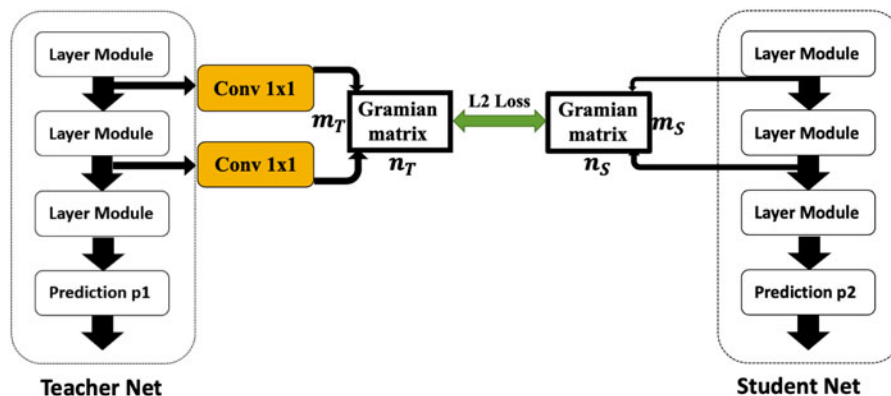


Fig. 12. Using 1×1 convolutional layers to decrease channels.

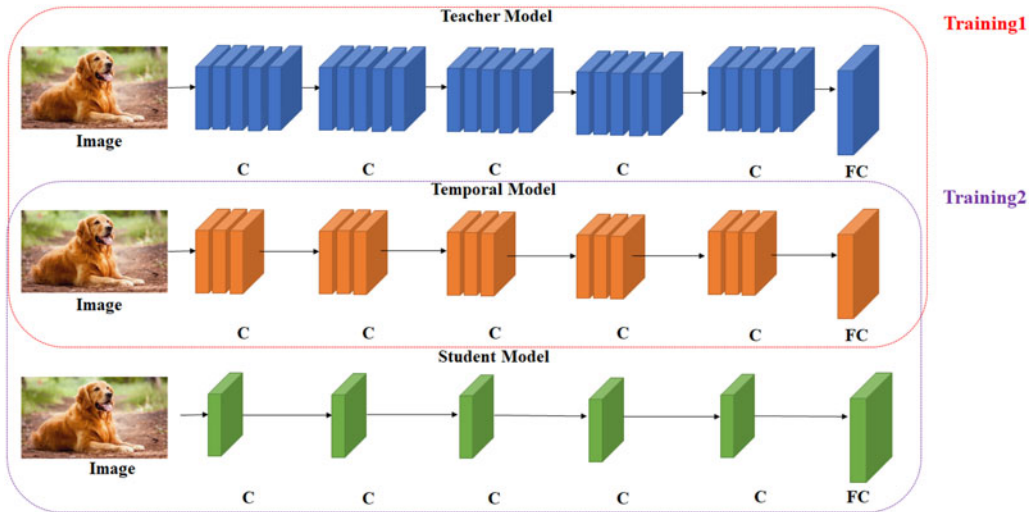


Fig. 13. The illustration of two-stage knowledge distillation. C, convolutional layer; FC, fully connected layer.

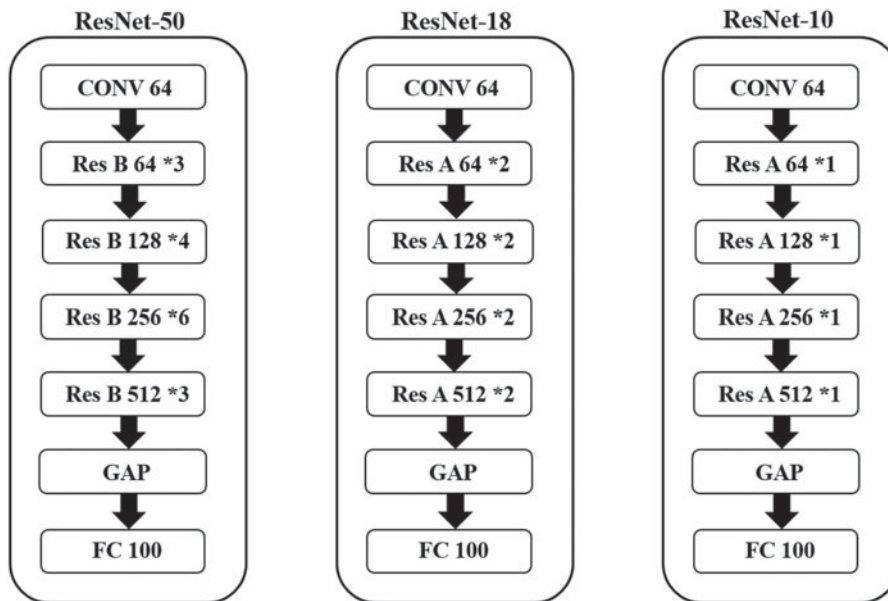


Fig. 14. ResNet-50/ResNet-18 /ResNet-10.

Table 20. Methods of adding 1×1 convolution to solve the limitation of proposed method and multi-steps compression with ResNet models and CIFAR-100. T-DNN1: ResNet-50. T-DNN2: ResNet-18. S-DNN: ResNet-10.

	FLOPs [M]	Params [M]	Exp1.	Exp2.	Exp3.	Average	Differential Accuracy
ResNet-50	924.46	17.08	70.56%	70.37%	70.49%	70.47%	3.09%
ResNet-18	550.90	11.02	68.78%	69.21%	70.05%	69.34%	1.96%
ResNet-10{Baseline}	248.42	4.75	66.93%	67.16%	68.07%	67.38%	0.00%
1x1(50 ->18)	550.90	11.02	73.67%	73.40%	73.68%	73.58%	6.20%
1x1(50 ->10)	248.42	4.75	72.51%	72.62%	73.04%	72.72%	5.34%
1x1(50->18->10)	248.42	4.75	73.75%	74.05%	73.98%	73.92%	6.54%

VI. CONCLUSION

In this paper, we propose a method using cross-layer knowledge distillation with KL divergence and offline ensemble to extract more knowledge from T-DNN to S-DNN to improve

the Top-1 accuracy of image classification. Moreover, we use a 1×1 convolutional layer to tune the dimension of Gramian matrix to solve the limitation of our proposed method and we further propose a method, two-stage KD, to avoid the loss of knowledge transfer.

REFERENCES

- [1] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L.: Imagenet: a large-scale hierarchical image database, in *IEEE Conf. on Computer Vision and Pattern Recognition, 2009. CVPR 2009*. IEEE, Florida, UAS, 2009, pp. 248–255.
- [2] Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A.: The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, **88** (2) (2010), 303–338.
- [3] Cordts, M.; *et al.*: The cityscapes dataset for semantic urban scene understanding, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE, Las Vegas, USA, 2016, pp. 3213–3223.
- [4] Choy, C.B.; Xu, D.; Gwak, J.; Chen, K.; Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, in *European Conf. on Computer Vision*. Springer, Amsterdam, the Netherlands, 2016, pp. 628–644.
- [5] Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R.: Indoor segmentation and support inference from rgb-d images, in *European Conf. on Computer Vision*. Springer, Florence, Italy, 2012, pp. 746–760.
- [6] Yim, J.; Joo, D.; Bae, J.; Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, Hawaii, 2017*, pp. 4133–4141.
- [7] Hinton, G.; Vinyals, O.; Dean, J.: Distilling the knowledge in a neural network, preprint [arXiv:1503.02531](https://arxiv.org/abs/1503.02531), 2015.
- [8] Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y.: Fitnets: hints for thin deep nets, preprint [arXiv:1412.6550](https://arxiv.org/abs/1412.6550), 2014.
- [9] Chen, T.; Goodfellow, I.; Shlens, J.: Net2net: accelerating learning via knowledge transfer, preprint [arXiv:1511.05641](https://arxiv.org/abs/1511.05641), 2015.
- [10] Lee, S.H.; Kim, D.H.; Song, B.C.: Self-supervised knowledge distillation using singular value decomposition, in *European Conf. on Computer Vision*. Springer, Munich, Germany, 2018, pp. 339–354.
- [11] Park, W.; Kim, D.; Lu, Y.; Cho, M.: Relational knowledge distillation, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Long Beach, California, 2019*, pp. 3967–3976.
- [12] Zhang, Y.; Xiang, T.; Hospedales, T.M.; Lu, H.: Deep mutual learning, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, Utah, 2018*, pp. 4320–4328.
- [13] Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G.E.; Hinton, G.E.: Large scale distributed neural network training through online distillation, 2018.
- [14] Lan, X.; Zhu, X.; Gong, S.: Knowledge distillation by on-the-fly native ensemble, *arXiv preprint arXiv:1806.04606*, 2018.
- [15] Krizhevsky, A.; Sutskever, I.; Hinton, G.E.: Imagenet classification with deep convolutional neural networks, in *26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3–6, 2012*, Lake Tahoe, Nevada, United States, 2012, pp. 1097–1105.
- [16] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [17] Ren, S.; He, K.; Girshick, R.; Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks, in *Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015*, Montreal, Quebec, Canada, 2015, pp. 91–99.
- [18] Redmon, J.; Farhadi, A.: Yolo9000: better, faster, stronger, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Honolulu, Hawaii, USA, 2017*, pp. 7263–7271.
- [19] Liu, Y.; Chen, K.; Liu, C.; Qin, Z.; Luo, Z.; Wang, J.: Structured knowledge distillation for semantic segmentation, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Long Beach, California, USA, 2019*, pp. 2604–2613.
- [20] Ji, S.; Xu, W.; Yang, M.; Yu, K.: 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern. Anal. Mach. Intell.*, **35** (1) (2013), 221–231.
- [21] Wang, L.; Yoon, K.-J.: Knowledge distillation and student-teacher learning for visual intelligence: a review and new outlooks, *IEEE Trans. Pattern Anal. Mach. Intell.* (2021). doi:10.1109/TPAMI.2021.3055564.
- [22] Wang, X.; Zhang, R.; Sun, Y.; Qi, J.: Kdgan: Knowledge distillation with generative adversarial networks, in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018. [Online]. Available: <https://proceedings.neurips.cc/paper/2018/file/019d385eb67632a7e958e23f24bd07d7-Paper.pdf>.
- [23] Bae, J.-H.; Yeo, D.; Yim, J.; Kim, N.-S.; Pyo, C.-S.; Kim, J.: Densely distilled flow-based knowledge transfer in teacher-student framework for image classification. *IEEE Trans. Image. Process.*, **29**, (2020), 5698–5710.
- [24] Hanson, S.J.; Pratt, L.Y.: Comparing biases for minimal network construction with back-propagation, in *Advances in Neural Information Processing Systems 2, NIPS Conference*, Denver, Colorado, USA, 1989, pp. 177–185.
- [25] Han, S.; Pool, J.; Tran, J.; Dally, W.: Learning both weights and connections for efficient neural network, in *Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015*, Montreal, Quebec, Canada, 2015, pp. 1135–1143.
- [26] Han, S.; Mao, H.; Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149), 2015.
- [27] Tung, F.; Mori, G.: Clip-q: Deep network compression learning by in-parallel pruning-quantization, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, Utah, USA, 2018*, pp. 7873–7882.
- [28] Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P.: Pruning filters for efficient convnets, preprint [arXiv:1608.08710](https://arxiv.org/abs/1608.08710), 2016.
- [29] Hsiao, T.-Y.; Chang, Y.-C.; Chou, H.-H.; Chiu, C.-T.: Filter-based deep-compression with global average pooling for convolutional networks. *J. Syst. Arch.*, **95**, (2019), 9–18.
- [30] Deng, L.; Li, G.; Han, S.; Shi, L.; Xie, Y.: Model compression and hardware acceleration for neural networks: a comprehensive survey. *Proc. IEEE*, **108** (4) (2020), 485–532.
- [31] Sze, V.; Chen, Y.-H.; Yang, T.-J.; Emer, J.S.: Efficient processing of deep neural networks: a tutorial and survey. *Proc. IEEE*, **105** (12) (2017), 2295–2329.
- [32] Denil, M.; Shakibi, B.; Dinh, L.; De Freitas, N.; *et al.*: Predicting parameters in deep learning, in *27th Annual Conference on Neural Information Processing Systems 2013*, Lake Tahoe, Nevada, USA, pp. 2148–2156.
- [33] Denton, E.L.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation, in *28th Annual Conference on Neural Information Processing Systems 2014*, Montreal, Canada, 2014, pp. 1269–1277.
- [34] Jaderberg, M.; Vedaldi, A.; Zisserman, A.: Speeding up convolutional neural networks with low rank expansions, preprint [arXiv:1405.3866](https://arxiv.org/abs/1405.3866), 2014.
- [35] Zhang, X.; Zou, J.; He, K.; Sun, J.: Accelerating very deep convolutional networks for classification and detection. *IEEE Trans. Pattern. Anal. Mach. Intell.*, **38** (10) (2015), 1943–1955.
- [36] Basu, S.; Varshney, L.R.: Universal source coding of deep neural networks, in *DCC 2017 Data Compression Conference*, Snowbird, Utah, USA, 2017, pp. 310–319.

- [37] Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P.: Deep learning with limited numerical precision, in *The 32nd International Conference on Machine Learning (ICML 2015)*, France, 2015, pp. 1737–1746.
- [38] Dettmers, T.: 8-bit approximations for parallelism in deep learning, preprint [arXiv:1511.04561](https://arxiv.org/abs/1511.04561), 2015.
- [39] Hwang, K.; Sung, W.: Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1, in *SIPS 2014: IEEE Workshop on Signal Processing Systems*, Belfast, Ireland, UK, 2014, pp. 1–6.
- [40] Ji, Z.; Ovsianikov, I.; Wang, Y.; Shi, L.; Zhang, Q.: Reducing weight precision of convolutional neural networks towards large-scale on-chip image recognition, in *Independent Component Analyses, Compressive Sampling, Large Data Analyses (LDA), Neural Networks, Biosystems, and Nanoengineering XIII*, vol. 9496. International Society for Optics and Photonics, 2015, p. 94960A.
- [41] Kang, L.-W.: Special issue on deep learning based detection and recognition for perceptual tasks with applications. *APSIPA Trans. Signal Inform. Proc.*, **8**, (2019), e21.
- [42] Wang, H.-P.; Peng, W.-H.; Ko, W.-J.: Learning priors for adversarial autoencoders. *APSIPA Trans. Signal Inform. Proc.*, **9**, (2020), e4.
- [43] Pascanu, R.; Mikolov, T.; Bengio, Y.: On the difficulty of training recurrent neural networks, 2012.
- [44] Krizhevsky, A.; G. Hinton *et al.*: Learning multiple layers of features from tiny images, Citeseer, Tech. Rep., 2009.
- [45] Chrabaszcz, P.; Loshchilov, I.; Hutter, F.: A downsampled variant of imagenet as an alternative to the cifar datasets, preprint [arXiv:1707.08819](https://arxiv.org/abs/1707.08819), 2017.
- [46] Abadi, M.; *et al.*: Tensorflow: a system for large-scale machine learning, in *Proceedings of OSDI '16: 12th USENIX Symposium on Operating Systems Design and Implementation OSDI*, Savannah, GA, USA, 2016, pp. 265–283.
- [47] Kiefer, J.; Wolfowitz, J.; *et al.*: Stochastic estimation of the maximum of a regression function. *Annal Math. Stat.*, **23** (3) (1952), 462–466.
- [48] Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN USSR*, **269**, (1983), 543–547.

Hsin-Hung Chou received a B.S. degree in Communications, Navigation and Control Engineering from National Taiwan Ocean University, Taipei, Taiwan. He received M.S. degree in the Department of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan. His research topics focused on Deep Neural Network models compression with heuristic methods. He is now working in Cisco Taiwan and is a consulting engineer. He is focusing on 5G Core Network and importing 5GCN in manufacturing industries with automation tools to help customers digital transformation.

Ching-Te Chiu received her B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan. She received her Ph.D. degree from University of Maryland, College Park, Maryland, USA, all in electrical engineering. She is a Professor at the Computer Science Department and Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan. Currently, she is the Director of Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan. She was member of technical staff with AT&T, Lucent Technologies, Murry Hill, NJ, USA, and with Agere Systems, Murry Hill, NJ, USA. Her research interests include Machine Learning, Pattern Recognition, High Dynamic Range Image and Video Processing, Super Resolution, High Speed SerDes design, Multi-chip Interconnect, and Fault Tolerance for Network-on-Chip design. Dr. Chiu won the first prize award, the best advisor award, and the best innovation award of the Golden Silicon Award. She served as the Chair of IEEE Signal Processing Society, Design and Implementation of Signal Processing Systems (DISPS) TC. She is a TC member of the IEEE Circuits and Systems Society, Nanoelectronics and Gigascale Systems Group. She was the program chair of the first IEEE Signal Processing Society Summer School at Hsinchu, Taiwan 2011 and technical program chair of IEEE workshop on signal processing system (SiPS) 2013. She served as associate editor of IEEE Transactions on Circuits and Systems I and served as associate editor of *IEEE Signal Processing Magazine and Journal of Signal Processing Systems*.

Ms. Yi-Ping Liao Student at University of National Tsing Hua.