Self-supervised Knowledge Distillation Using Singular Value Decomposition

Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song(✉)

Inha University, Incheon, Republic of Korea
lsh910703@gmail.com, kdhht5022@gmail.com, bcsong@inha.ac.kr

Abstract. To solve deep neural network (DNN)'s huge training dataset and its high computation issue, so-called teacher-student (T-S) DNN which transfers the knowledge of T-DNN to S-DNN has been proposed. However, the existing T-S-DNN has limited range of use, and the knowledge of T-DNN is insufficiently transferred to S-DNN. To improve the quality of the transferred knowledge from T-DNN, we propose a new knowledge distillation using singular value decomposition (SVD). In addition, we define a knowledge transfer as a self-supervised task and suggest a way to continuously receive information from T-DNN. Simulation results show that a S-DNN with a computational cost of 1/5 of the T-DNN can be up to 1.1% better than the T-DNN in terms of classification accuracy. Also assuming the same computational cost, our S-DNN outperforms the S-DNN driven by the state-of-the-art distillation with a performance advantage of 1.79%. code is available on https://github.com/sseung0703/SSKD_SVD.

Keywords: Statistical methods and learning · Optimization methods
Recognition: detection · Categorization · Indexing · Matching

1 Introduction

Recently, DNN has overwhelmed other machine learning methods in the research fields such as classification and recognition [1,2]. As a result of the development of general-purpose graphics processing unit (GP-GPU) with high computational power, DNNs with huge complexity can be implemented and verified, resulting in DNNs that are superior to human recognition capabilities [3–5]. On the other hand, it is still challenging to operate DNN on a mobile device or embedded system due to limited memory and computational capability. Recently, various lightweight DNN models have been proposed to reduce memory burden and computation cost [6,7]. However, these small-size models have less performance than state-of-the-art models like ResNext [5]. Another problem is that not only the conventional DNN but also the lightweight DNN model requires huge data in learning.

As a solution to these two problems, Hinton et al. [8] defined the concept of knowledge distillation and presented a teacher-student (T-S) DNN based

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11210, pp. 339–354, 2018.
https://doi.org/10.1007/978-3-030-01231-1_21

on it. Then several knowledge distillation techniques have been studied [9,10]. For example, in [10], Yim et al. proposed a method to transfer the correlation between specific feature maps generated by T-DNN as the knowledge of T-DNN to the S-DNN. In this case, the S-DNN learns in two stages: the first stage that initializes the network parameters using the transferred knowledge, and the second stage that learns the main task.

However, the existing T-S knowledge distillation approaches have several limitations as follows: (1) They do not yet extract and distill rich information from the T-DNN. (2) In addition, the structure of T-S-DNN is very limited. (3) Finally, since the knowledge from the T-DNN is learned only for the purpose of initializing the parameters of the S-DNN, it gradually disappears as the learning of the next main task progresses.

In order to solve this problem, this paper approaches two perspectives. The first is a proper manipulation of knowledge for smaller memory and lower computation. So we gracefully compress the knowledge data by utilizing singular value decomposition (SVD), which is mainly applied to dimension reduction of features [11–13] in signal processing domain. We also analyze the correlation between compressed feature maps through a radial basis function (RBF) [14,15], which is often used for kernelized learning. As a result, knowledge distillation using SVD and RBF can distill the information of T-DNN more efficiently than conventional techniques, and can transfer regardless of the spatial resolution of feature maps. Second, the training mechanism [16–18] through self-supervised learning, which learns to create labels by itself, ensures that the transferred knowledge does not vanish and is continuously used. That is, it can figure out the vanishing problem of T-DNN knowledge. In addition, self-supervised learning can be expected to provide additional performance improvement because it allows for more powerful regularization [8].

The experimental results show that when the visual geometry group (VGG) model [19] is applied to the proposed network, T-DNN with 64.4% accuracy for CIFAR-100 can improve the performance of S-DNN with 1/5 computation cost of T-DNN by 65.1%. In addition to VGG, state-of-the-art models such as MobileNet [7] and ResNext [5] are also applied to the proposed knowledge distillation method, confirming similar effects and proving that the proposed method can be generalized. Finally, we introduced self-supervised learning to continuously deliver the T-DNN's knowledge. As a result, we confirmed that the performance of the S-DNN is further improved by a maximum of 1.2%, and finally the performance of the S-DNN becomes superior to the T-DNN by 1.79%.

2 Related Works

2.1 Knowledge Distillation

Knowledge transfer is a technique for transferring information from a relatively complex and deep model, i.e., T-DNN to a smaller DNN model, i.e., S-DNN, ultimately increasing the performance of the S-DNN [8]. FitNet [9] first introduced the two-stage method to re-train the main task of the S-DNN after transferring

knowledge of the T-DNN. The S-DNN could have much better initial parameters by learning knowledge distilled from the T-DNN than random initialization. Yim et al. [10] defined the knowledge transferred from the T-DNN to the S-DNN as changes of feature maps rather than layer parameters. They determined a certain layer group in the network and defined the correlation between input and output feature maps of the layer group as a Gram matrix so that the feature correlations of the S- and T-DNN become similar. However, the knowledge defined by the above techniques still lacks information, and knowledge transfer through initialization is still limited.

2.2 SVD and RBF

SVD is mainly used for dimension reduction or for extracting important information from feature maps [11–13]. In [11], Alter et al. showed that it is possible to abstract the information of a dataset by using SVD. Lonescu et al. defined the gradient according to the chain rule for SVD, and proved that end-to-end learning is realizable even in DNN using SVD [13]. They also showed that pooling high-level information in the feature map is very effective in the feature analysis tasks such as recognition and segmentation. RBF is a function that re-maps each feature in a viewpoint of distance from the center so that the feature has a high dimension. RBF can be used for various kernelized learning or RBF network (RBFN) [14,15]. In particular, analyzing features with RBF such as Gaussian function makes it possible to analyze noisy data more robustly. If these two methods can be combined well, it will be possible to extract important information effectively from fuzzy and noisy data. The proposed knowledge distillation method efficiently extracts core knowledge from a given feature map using SVD and effectively computes the correlation between two feature maps using RBF.

2.3 Training Mechanism

Self-supervised learning generates labels and learns them by itself. Recently, various self-supervised learning tasks have been studied [16–18] because they can effectively initialize the network model. In [18], a method to learn various self-supervised tasks at a time by bundling them into a multi-task has been proposed and proved to be more efficient than conventional methods. On the other hand, semi-supervised learning is another learning scheme that uses labeled and unlabeled data at the same time when labeling data is insufficient. In order to solve the fundamental problem of the lack of a training-purpose dataset, various studies on semi-supervised learning have been actively conducted [20,21].

We will introduce the above-mentioned self-supervised learning as a more efficient transfer approach than parameter initialization through knowledge transfer in the existing T-S-DNNs.

3 Method

This section details the proposed knowledge transfer method. Inspired by the idea of [10], we derive a correlation between two feature maps extracted from T-

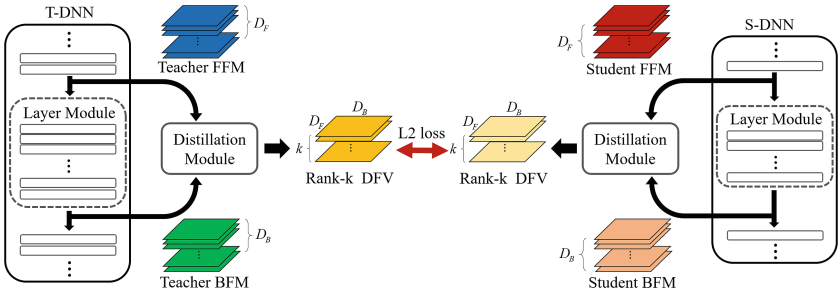

Fig. 1. The concept of the proposed knowledge distillation-based network.

DNN, and transfer it as knowledge. Figure 1 illustrates the proposed knowledge distillation based network. First, both the T-DNN and the S-DNN are composed of a predetermined convolutional layer and a fully-connected layer depending on the purpose. For example, VGG [19], MobileNet [7], ResNext [5], etc. can be adopted as DNN. Then, to extract the feature map characteristic inherent to each DNN, we specify two particular layer points in the DNN and sense the corresponding two feature maps. The layers between the two points are defined as a layer module. The feature map that is sensed at the input of the layer module is called the front-end feature map (FFM) and the feature map that is sensed at the output is called the back-end feature map (BFM). For example, in MobileNet, the layer module can consist of several depth-wise separable convolutions. Let the depths of FFM and BFM be D_F and D_B, respectively. On the other hand, several non-overlapping layer modules may be defined in each DNN for robust distillation. In this paper, the maximum number of layer modules in each DNN is G.

Now we can get the correlation between FFM and BFM of a certain layer module through the distillation module. The distillation module outputs the distillation feature vectors (DFV) having the size of $k \times D_F \times D_B$ from two inputs of FFM and BFM. See Sect. 3.1.

Finally, we propose a novel training mechanism so that the knowledge from the T-DNN does not disappear in the 2nd stage, i.e., main-task learning process. We improve self-supervised learning, which was presented in [8], to enable more effective transfer of knowledge. See Sect. 3.2.

3.1 Proposed Distillation Module

In general, DNNs generate feature maps through multiple layers to suit a given task. In the distillation method of [10], the correlation between feature maps obtained from DNN is first defined as knowledge. The proposed method also accepts the idea of [10] and distillates the knowledge using correlation between feature maps. However, feature maps that are produced through multiple convolution layers are generally too large to be used as they are not only computationally expensive, but also difficult to learn. An intuitive way to solve this problem

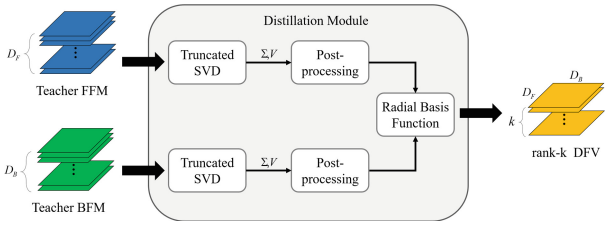

Fig. 2. The proposed knowledge distillation module.

is to reduce the spatial dimensions of the feature maps. We introduce SVD to effectively remove spatial redundancy in feature maps and obtain meaningfully implied feature information in the process of reducing feature dimensions. This section describes in detail how to generate DFV, i.e., knowledge for distillation using SVD.

Figure 2 shows the structure of the proposed knowledge distillation module. Suppose that the input and output feature maps of the layer module defined in T-DNN, i.e., FFM and BFM are inputs to this distillation module. First, we eliminate the spatial redundancy of feature maps by using truncated SVD. Then, the right-hand singular vectors V obtained from the truncated SVD and the singular value matrix are post-processed for easy learning, and then k feature vectors are obtained. Finally, the correlation between feature vectors obtained from FFM and BFM is computed by RBF to obtain a rank-k DFV.

Truncated SVD. As shown in Fig. 3(a), the first step of the distillation module is the truncated SVD which is used to compress the feature map information and lower the dimension simultaneously. Prior to applying SVD, preprocessing is performed to convert the 3D feature map information of $H \times W \times D$ into a 2D matrix M having $(H \times W) \times D$ size. Then M can be a factorization of the form $U\Sigma V^T$ by SVD. V^T is the conjugate transpose of V. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of M, respectively. The non-zero singular values of M (found on the diagonal entries of Σ) are the square roots of the non-zero eigenvalues of both $M^T M$ and MM^T. On the other hand, U and V decomposed through SVD have different information [11]. U is the unique pattern information of each feature of M, and V can be interpreted as global information of the feature set. And Σ has the scale or energy information of the singular value. Since we aim to obtain compressed feature information, we use only V having global information of the feature map and its energy Σ.

To minimize memory size as well as computational cost, we use truncated SVD. Truncated SVD refers to an SVD that decomposes a given matrix by only a pre-determined rank k. That is, V and Σ have dimensions of $k \times D$ and $k \times 1$, respectively. In this case, since the difference between the re-composed matrix and the original matrix is minimized, the information of the given matrix M can

be maintained as much as possible. As a result, FFM and BFM are compressed with minimal loss of information as shown on Fig. 3(a).

On the other hand, in order to apply the chain rule by back propagation to the truncated SVD part in the learning process, the gradient of M must be defined. So, we modify the gradient defined in [13]. Note that the proposed scheme uses only V and Σ among decomposed vectors, unlike [13]. Since Σ is simply used as a scale factor, it is not necessary to obtain its gradient. Therefore, only the gradient for V is obtained and the gradient of M is re-defined as in Eqs. (1)–(2).

$$\nabla(M) = \begin{cases} UE^T - U\left(E^T V\right)_{diag} V^T \\ \quad -2U\left(K \circ \left(\Sigma^T V^T E\right)\right)_{sym} \Sigma^T V^T, \text{HW} \leq D \\ 2U\Sigma\left(K^T \circ \left(V^T \nabla(V)\right)\right)_{sym} V^T, \quad otherwise \end{cases} \tag{1}$$

$$E = \nabla(V)\,\Sigma^{-1}, K = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, i \neq j, (1 \leq i, j \leq k) \\ 0, \quad otherwise \end{cases} \tag{2}$$

where $(A)_{sym} = \frac{1}{2}\left(A^T + A\right)$and $(A)_{diag}$ is a function that makes all off-diagonal components zero. Also $\circ$ indicates Hadamard product, and σ stands for diagonal component of Σ. We do not need to perform unnecessary operations on $\nabla(\Sigma)$ and $\nabla(U)$, and since the dimension of each matrix is low, the computation cost can be minimized as a whole.

Therefore, truncated SVD is a key element of the proposed knowledge distillation module because it effectively reduces the dimension of the feature map. As a result, the proposed knowledge distillation functions to fit the small size network.

Post-processing. Truncated SVD products, V and Σ contain enough FFM and BFM information, but are difficult to use directly because of the following two problems. First, since SVD decomposes a given matrix in decreasing order of energy, the order of singular vectors with similar energy can be reversed. Second, because each element of the singular vector has a value of [−1,1], singular vectors with the same information but the opposite direction may exist. So, even with similar feature maps, the results of decomposing them may seem to be very different.

Therefore, the corresponding singular vectors of T-DNN and S-DNN are post-processed differently based on T-DNN because T-DNN delivers its information to S-DNN. First, post-processing for T-DNN is described in Fig. 3(b). The singular value of T-DNN Σ_T is normalized so that the square sum becomes 1. Normalization is performed by multiplying a normalized Σ_T with singular vector of T-DNN V_T as shown in Eq. (4) to obtain a set of compressed feature vectors F_T as shown in Eq. (3).

$$F_T = \{f_{T,i} | 1 \leq i \leq k\} \tag{3}$$

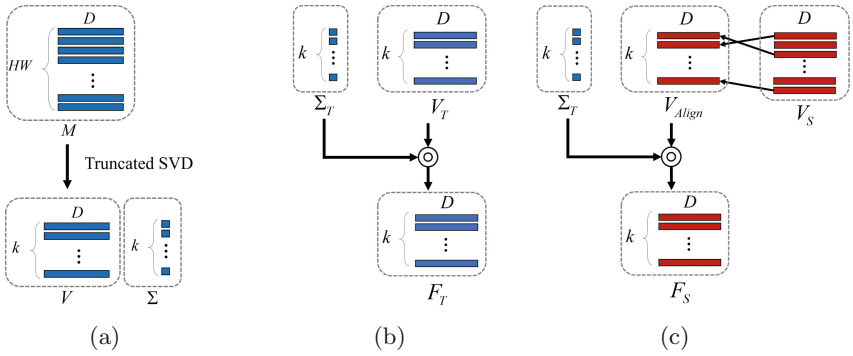

Fig. 3. (a) Truncated SVD (b) post-processing of T-DNN (c) post-processing of S-DNN

$$f_{T,i} = \frac{\sigma_{T,i}}{||\Sigma_T||_2} v_{T,i} \tag{4}$$

where $\sigma_{T,i}$ is the i-th singular value of T-DNN and $v_{T,i}$ is the corresponding singular vector. Since the singular value means the energy of the corresponding singular vector, each singular vector is learned in order of importance.

Next, a singular vector of S-DNN is post-processed as shown in Fig. 3(c). First, we align the student singular vectors based on the teacher singular values. So the student singular vector with the most similar information to the teacher singular vector is aligned in the same order.

Here, the similarity between singular vectors is defined as the absolute value of cosine similarity, which determines the similarity degree through the angles between two vectors so that the similarity between the vectors with opposite directions can be accurately measured. This process is described in Eqs. (5–6).

$$s_j = \underset{j}{\operatorname{argmax}} \left(|v_{T,i} \cdot v_{S,j}| \right), (1 \leq i \leq k), (1 \leq j \leq k+1) \tag{5}$$

$$v_{Align,i} = sign\left(v_{T,i} \cdot v_{S,s_j}\right) v_{S,s_j} \tag{6}$$

Here $v_{S,j}$ indicates the j-th vector of the S-DNN's V and $v_{Align,i}$ is the i-th vector of the aligned version of the S-DNN's V. Note that for effective alignment, the student feature map decomposes one more vector. Also, the singular vectors of S-DNN are normalized by the singular values of T-DNN, so that a singular vector of higher importance is further learned. This is shown in Eqs. (7–8).

$$F_S = \{f_{S,i} | 1 \leq i \leq k\}, \tag{7}$$

$$f_{S,i} = \frac{\sigma_{T,i}}{||\Sigma_T||_2} v_{Align,i} \tag{8}$$

Thus, because of the post-processing, noisy and randomly decomposed singular vector information can be used effectively.

Computing Correlation Using Radial Basis Function. This section describes the process of defining knowledge by the correlation of the feature vectors obtained in the previous section. Since the derived feature information from a singular vector is generally noisy, noise-robust methods are required. Therefore, we employ Gaussian RBF, which is a frequently used kernel function for analyzing noisy data [14,15], as a way to obtain the correlation.

On the other hand, feature vectors obtained by applying the proposed SVD and post-processing to FFM and BFM are basically discrete random vectors independent of each other. Thus, we define the correlation between feature vector sets obtained from FFM and BFM as a point-wise L_2 distance as in Eq. (10), and the rank-k DFV are completed by applying Gaussian RBF to the computed correlation as in Eq. (9) for the dimension extension.

$$DFV = \left\{ \exp\left(-\frac{d_{m,n,l}}{\beta} \right), 1 \le m \le D_F, 1 \le n \le D_B, 1 \le l \le k \right\} \tag{9}$$

$$d_{m,n,l} = \left\| f_{m,l}^{FFM} - f_{n,l}^{BFM} \right\|_2^2 \tag{10}$$

β in Eq. (9) is a hyper-parameter for smoothing DFV and it should be properly selected for noise-robust operation.

As mentioned above, the correlation between feature maps composed of noisy and fuzzy data can be effectively obtained through SVD and RBF. Therefore, the distillated knowledge from T-DNN by the proposed scheme can be a very effective guidance for S-DNN. Also, unlike the existing technique, DFV can transfer knowledge regardless of feature map size and therefore it causes consistent performance. The experimental results are discussed in Sect. 4.2.

3.2 Training Mechanism

The remaining step is to learn to improve the performance of S-DNN by transferring distilled knowledge of T-DNN, i.e., DFV, to S-DNN. We need to learn that the S-DNN imitates the T-DNN with the DFV as an intermediary, so we define the L_2 loss function $L_{transfer}\left(DFV_T, DFV_S\right)$ of the knowledge pair of T-DNN and S-DNN as Eq. (11).

$$L_{transfer}\left(DFV_T, DFV_S\right) = \sum_{g}^{G} \frac{\left\| DFV_T^{(g)} - DFV_S^{(g)} \right\|_2^2}{2} \tag{11}$$

where G is the maximum number of layer modules defined in the proposed T-S-DNN. In this case, all layer modules are assumed to have the equivalent importance, and are trained without additional weighting. If S-DNN is initialized by transferring knowledge of T-DNN to S-DNN through learning based on Eq. (11), the learning performance of the main task of S-DNN can be improved (see Sect. 4.2).

However, even though learning the main task of S-DNN after initialization as described above, there is still a problem that the knowledge of T-DNN gradually disappears as learning progresses and the performance improvement is limited. So we introduce self-supervised learning to train both main task and transfer task at the same time. Since the knowledge of T-DNNs learned by S-DNN is a label generated by T-DNN, self-supervised learning is possible using this characteristic. As a result, the final loss function for learning the parameter of S-DNN Θ_S is defined as Eq. (12).

$$L_{total}\left(\Theta_S\right) = L_{main}\left(\Theta_S\right) + L_{transfer}\left(DFV_T, DFV_S\right) \tag{12}$$

As described above, when the main task and the transfer task are learned together by a multi-task learning, it is possible to continuously transfer knowledge of T-DNN to further improve the performance.

On the other hand, if the distillation loss is much larger than the main task loss, the gradient of knowledge transfer becomes too large and the above multi-task learning may not work properly. To solve this problem, it is necessary to limit the effect of the distillation task. So we introduce a gradient clipping [22] to limit the gradient of knowledge transfer.

In general, the threshold for clipping is constant, but we define the L_2-norm ratios of the main task and the transfer task as shown in Eq. (13), and clip the gradient of the knowledge transfer adaptively using this. In addition, since randomly initialized S-DNN is different from T-DNN, it is difficult to follow T-DNN fast. Therefore, we use a sigmoid function as shown in Eq. (14) to design the clipped gradient to grow smoothly as learning progresses.

$$\tau = \frac{\left\|\nabla\left(\Theta_S\right)_{main}\right\|_2}{\left\|\nabla\left(\Theta_S\right)_{trans}\right\|_2} \tag{13}$$

$$\nabla\left(\Theta_S\right)_{trans}^{clipped} = \begin{cases} \frac{1}{1+\exp(-\tau+p)}\nabla\left(\Theta_S\right)_{trans}, & \nabla\left(\Theta_S\right)_{trans} < \nabla\left(\Theta_S\right)_{main} \\ \nabla\left(\Theta_S\right)_{trans} & otherwise \end{cases} \tag{14}$$

In Eq. (14), p means the current epoch. Therefore, the proposed self-supervised learning method can concentrate more on the learning of the main task while learning the two tasks of different nature at the same time. In other words, rich knowledge distillated from T-DNN can be continuously transferred to S-DNN without vanishing. In addition, since the proposed self-supervised learning method has the effect of hard regularization of S-DNN, the performance of S-DNN can be improved without over-fitting (see Sect. 4.3).

4 Experimental Results

In order to evaluate the performance of the proposed knowledge distillation method, we performed the following three experiments. First, we verified the effectiveness of the proposed knowledge itself. To do this, we conducted experiments on so-called small network enhancement that improves the performance

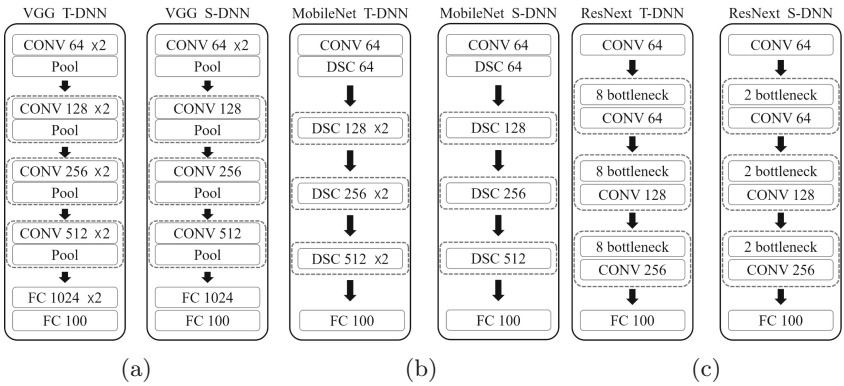

Fig. 4. A pair of T-DNN and S-DNN for an experiment to evaluate small network enhancement. (a) VGG, (b) MobileNet, (c) ResNext. Here dotted boxes indicate layer modules.

of a relatively small S-DNN using T-DNN knowledge (see Sect. 4.2). Second, we examined the performance of the training mechanism proposed in Sect. 3.2 (see Sect. 4.3). Here, the comparison target was Yim et al.'s two-stage approach [10]. Section 4.3 also demonstrates that the proposed method can transfer knowledge robustly even when there is no hard constraint on image information. Third, the performance of the proposed method according to the number of DFVs is experimentally examined in Sect. 4.4.

4.1 Experiment Environments

We implemented the proposed method using Tensorflow [23] on a computer with specification of the Intel Core i7-7700 CPU@3.60 GHz x8, 16 GB RAM, and GeForce GTX 1070. We used CIFAR100 [24]. The CIFAR100 dataset consists of color images with a small size of 32×32, with 50,000 training data and 10,000 test data divided into 100 categories or labels. The augmentations used here are random shift, random rotation, and horizontal flip. The proposed method was tested under the same conditions as [10], and the average of three equivalent experimental results was used as the final result to increase the reliability of the results.

4.2 Small Network Enhancement

In order to verify the effect of knowledge transfer only, we first showed the result of learning in two-stage approach as in [10]. That is, the self-supervised learning of Sect. 3.2 was not used in this experiment. We compared the proposed method and the state-of-the-art knowledge distillation method [10]. In addition, the results of T-DNN alone and S-DNN alone were also shown. All the methods were learned with the CIFAR100 dataset. We employed VGG, MobileNet,

ResNext as the DNN to apply to the proposed method. The T-S-DNNs constructed using these are shown in Fig. 4.

Although VGG is somewhat poorer than the state-of-the-art CNN models in terms of ratio of accuracy and parameter size, it is widely used because of its simple structure and ease of implementation. We used a modified version of the T-DNN for CIFAR100 by removing the last three convolutional layers from the VGG network proposed in [19]. The S-DNN consists of only one convolutional layer with the same filter depth as shown in Fig. 4(a). Here, the layer module is defined as a convolutional layer with the same filter depth.

MobileNet is a CNN with small parameter size and computational cost designed for use in mobile or embedded environments. The MobileNet case shows that the proposed method is capable of improving performance even for small networks. As shown in Fig. 4(b), T-DNN was constructed by removing the last four depth-wise separable convolutional layers (DSC) proposed in [7] to fit CIFAR100. The S-DNN is composed by using the DSC of the same filter depth only once. Here, the layer module is defined by the DSC of the same filter depth.

Finally, ResNext is a network where the convolution layer was divided into several bottleneck layers. Through experiments using ResNext, we show that the proposed method can transfer knowledge effectively even in networks with very complex structures. We used the network proposed in [5] as the T-DNN and the S-DNN is constructed by partially reducing the bottleneck layers. Here, the layer module is defined by combining the bottleneck layer and one convolutional layer (see Fig. 4(c)).

The weight of each network was determined by He's initialization [3] and L_2 regularization. Decay parameter was set to 10^{-4}. Batch size was set to 128, and stochastic gradient descent (SGD) [25] was used for optimization, and Nesterov accelerated gradient [26] was applied. The initial learning rate was set to 10^{-2} and the momentum was set to 0.9. During a total of 200 epochs, the networks were learned and the learning rate was reduced to 1/10 per 50 epochs. Both stages used the same hyper-parameters. The hyper-parameter of the proposed method k was set to is 1. In other words, only one DFV is used and β of RBF is experimentally fixed to 8.

The experimental results are shown in Table 1, and it can be seen that the proposed method is always better than [10]. In the case of VGG, the proposed method has an outstanding performance improvement of 3.68% compared to S-DNN. It also shows about 0.49% better performance than [10] and 0.61% higher performance than T-DNN alone. In case of Mobilenet, the proposed method improves the performance by about 2% over S-DNN, and 1.62% over [10] and 0.3% over T-DNN. This shows that the proposed method is more suitable for small networks than [10]. In the case of ResNext, the proposed method improves the performance of S-DNN by only 1.43%, which is lower than that of VGG or MobileNet, but has a performance advantage over 1.83% than [10]. This result shows that the proposed method works well in a state-of-the-art network with a complicated structure such as ResNext. Therefore, the proposed method effec-

Table 1. Comparison of the proposed algorithm with [10] for three different networks. Here, FLOPS indicates the sum of the numbers of addition, multiplication, and condition. Params indicates the sum of weights and biases.

Network	Model	FLOPs	Params	Accuracy
VGG	T-DNN	576.3M	10.9M	64.44
	S-DNN	121.3M	3.8M	61.37
	[10]	121.3M	3.8M	64.54
	Proposed	121.3M	3.8M	**65.05**
MobileNet	T-DNN	98.4M	2.3M	57.85
	S-DNN	37.8M	0.82M	56.15
	[10]	37.8M	0.82M	56.53
	Proposed	37.8M	0.82M	**58.15**
ResNext	T-DNN	547.3M	0.66M	66.58
	S-DNN	247.6M	0.34M	64.00
	[10]	247.6M	0.34M	63.60
	Proposed	247.6M	0.34M	**65.43**

Table 2. Sensitivity of the proposed network to spatial resolution of feature map.

Network	Model	FLOPs	Params	Accuracy
VGG	T-DNN	576.3M	10.9M	64.44
	S-DNN	15.6M	3.8M	54.17
	Proposed	15.6M	3.8M	61.15

tively compresses knowledge of T-DNN and transfers the compressed knowledge regardless of network structure.

On the other hand, we constructed another VGG-based S-DNN to show that the proposed method can transfer knowledge regardless of the resolution of feature maps. In the convolutional layer of the S-DNN used above, the padding was not performed and the size of the feature map was reduced by setting the stride of the convolutional layer to 2 instead of pooling. This dramatically reduces the spatial resolution of the feature map as it passes through the convolution layer. The hyper-parameters used for learning were the same as before.

Since knowledge transfer using [10] is impossible in this T-S-DNN structure, Table 2 shows only the results of the proposed method. We can see that the performance of S-DNN with FLOPS of about 0.03 times that of T-DNN is improved by about 6.98%. Therefore, the proposed method can effectively transfer the knowledge of T-DNN regardless of the spatial resolution of the feature map, and is effective for practical applications requiring small size DNNs.

Table 3. Performance evaluation according to training mechanism.

Model	Mechanism	Accuracy
[10]	2 stage	64.54
	1 stage	64.89
Proposed	2 stage	65.05
	1 stage	**65.54**

4.3 Training Mechanism

In this section, we evaluate the training mechanism proposed in Sect. 3.2. The network used for learning is the VGG-based T-S-DNN used in Sect. 4.2. The hyper-parameters are the same as those used in Sect. 4.2.

Table 3 shows the experimental results. The performance improvement was 0.35% when the proposed training mechanism was applied to [10], and the performance improved by 0.49% when the proposed training mechanism was applied together with the proposed knowledge distillation technique. This is because S-DNN is regularized continuously without vanishing of knowledge of T-DNN. In addition, since the number of epochs required for learning is reduced by half compared with the conventional two stage structure, the learning time can be shortened significantly. Therefore, using both the knowledge distillation technique and the training mechanism, the performance improvement is expected to be about 4.17% higher than that of the S-DNN alone. In addition, the proposed method can improve performance up to 1% than [10] and 1.1% over T-DNN. Since the computation cost of S-DNN amounts to only 1/5 of that of T-DNN, we can see that S-DNN is well regularized by the proposed method.

4.4 Performance Evaluation According to the Number of DFVs

The number of DFVs to be transferred in the proposed knowledge distillation has a significant impact on overall performance. For example, using too many DFVs will not only increase cost, but also deliver noisy information, so we need to find an optimal number. In this experiment, we adopted the VGG-based T-DNN used in Sect. 4.2. We took into account two types of S-DNNs for this experiment: S-DNN with pooling and S-DNN with stride.

The experimental results of the proposed method were shown in Table 4. In general, performance was improved regardless of the number of DFVs, but in the case of S-DNN with pooling, we could observe that as the number of DFVs becomes too large, the accuracy rises and drops again. This is because the distillation of too much amount of knowledge may cause transfer of even unnecessary information as mentioned in Sect. 3. However, S-DNN with stride shows a slight increase in performance. This is because the performance of the S-DNN is relatively low compared to that of the T-DNN, so receiving additional knowledge will significantly improve performance. Therefore, a reasonable num-

Table 4. Performance comparison according to the number of DFVs.

VGG	Model	The number of DFVs					
		-	1	2	4	8	16
VGG	S-DNN w/pool	61.37	65.54	**66.33**	66.17	65.38	65.15
	S-DNN w/stride	54.17	61.28	61.54	61.63	61.82	62.00

ber of DFVs should be used depending on the available cost, and the number of DFVs required can be determined according to the structure of the network.

5 Conclusion and Future Work

We propose a novel knowledge distillation method in this paper. The existing knowledge transfer technique (1) was limited to a limited network structure, (2) the quality of knowledge was low, and (3) as the learning progresses, the knowledge of the T-DNN vanished rapidly. We have proposed a method to transfer very rich information by defining novel knowledge using SVD and RBF, which are frequently used in traditional machine learning, without any structural limitations of the network. In addition, self-supervised learning associated with multi-task learning have been applied so that it was able to continue to receive T-DNN's knowledge during the learning process, which could also lead to additional performance enhancement. Experimental results showed that the proposed method has a significant improvement of about 4.96% compared to the 3.17% improvement in terms of accuracy performance based on VGG network [10]. In the future, we will develop a semi-supervised learning scheme by extending self-supervised learning concept through proposed knowledge transfer.

Acknowledgements. This research was supported by National Research Foundation of Korea Grant funded by the Korean Government (2016R1A2B4007353).

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
4. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. 3 (2017)

5. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995. IEEE (2017)
6. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017)
7. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
9. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: FitNets: hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014)
10. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: fast optimization, network minimization and transfer learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
11. Alter, O., Brown, P.O., Botstein, D.: Singular value decomposition for genome-wide expression data processing and modeling. Proc. Natl. Acad. Sci. **97**(18), 10101–10106 (2000)
12. Zhang, Z., Ely, G., Aeron, S., Hao, N., Kilmer, M.: Novel methods for multilinear data completion and de-noising based on tensor-SVD. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3842–3849 (2014)
13. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2965–2973 (2015)
14. Kim, N., Byun, H.G., Kwon, K.H.: Learning behaviors of stochastic gradient radial basis function network algorithms for odor sensing systems. ETRI J. **28**(1), 59–66 (2006)
15. Wang, X.X., Chen, S., Harris, C.J.: Using the correlation criterion to position and shape RBF units for incremental modelling. Int. J. Autom. Comput. **3**(4), 392–403 (2006)
16. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016 Part IV. LNCS, vol. 9908, pp. 577–593. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_35
17. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016 Part VI. LNCS, vol. 9910, pp. 69–84. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_5
18. Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
20. Zhou, X., Belkin, M.: Semi-supervised learning. In: Academic Press Library in Signal Processing, vol. 1, pp. 1239–1269. Elsevier (2014)
21. Su, H., Zhu, J., Yin, Z., Dong, Y., Zhang, B.: Efficient and robust semi-supervised learning over a sparse-regularized graph. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016 Part VIII. LNCS, vol. 9912, pp. 583–598. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_35
22. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)
23. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). tensorflow.org

24. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
25. Kiefer, J., Wolfowitz, J.: Stochastic estimation of the maximum of a regression function. Ann. Math. Stat. **23**, 462–466 (1952)
26. Nesterov, N.: A method for unconstrained convex minimization problem with the rate of convergence o (1/k^2). Doklady AN USSR **269**, 543–547 (1983)