A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning

Junho Yim[1] Donggyu Joo[1] Jihoon Bae[2] Junmo Kim[1]
[1]School of Electrical Engineering, KAIST, South Korea
[2]Electronics and Telecommunications Research Institute
{junho.yim, jdg105, junmo.kim}@kaist.ac.kr
{baejh}@etri.re.kr

Abstract

We introduce a novel technique for knowledge transfer, where knowledge from a pretrained deep neural network (DNN) is distilled and transferred to another DNN. As the DNN maps from the input space to the output space through many layers sequentially, we define the distilled knowledge to be transferred in terms of flow between layers, which is calculated by computing the inner product between features from two layers. When we compare the student DNN and the original network with the same size as the student DNN but trained without a teacher network, the proposed method of transferring the distilled knowledge as the flow between two layers exhibits three important phenomena: (1) the student DNN that learns the distilled knowledge is optimized much faster than the original model; (2) the student DNN outperforms the original DNN; and (3) the student DNN can learn the distilled knowledge from a teacher DNN that is trained at a different task, and the student DNN outperforms the original DNN that is trained from scratch.

1. Introduction

Over the past several years, various deep neural network (DNN) models have provided state-of-the-art performance in many tasks, ranging from computer vision [8, 23] to natural language processing [1, 19]. Recently, several studies on the knowledge transfer technique have been conducted [11, 20]. Hinton et al. [11] first proposed the concept of knowledge distillation (KD) in the teacher–student framework by introducing the teacher's softened output. Although the KD training achieved improved accuracy over several datasets, this method has limitations such as difficulty with optimizing very deep networks. To improve the performance of the KD training for deeper networks, Romero et al. [20] devised a hint-based training approach that uses the pretrained teacher's hint layer and student's guided layer. Thanks to the additional hint-based training, the trained deep student network showed better accuracy with fewer parameters compared to the original wide teacher network.

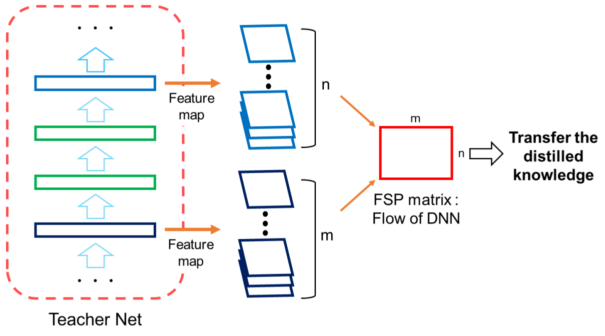

Figure 1. Concept diagram of the proposed transfer learning method. The FSP matrix, which represents the distilled knowledge from the teacher DNN, is generated by the features from two layers. By computing the inner product, which represents the direction, to generate the FSP matrix, the flow between two layers can be represented by the FSP matrix.

The knowledge transfer performance is very sensitive to how the distilled knowledge is defined. The distilled knowledge can be extracted by various features in the pretrained DNN. Considering that a real teacher teaches a student the flow for how to solve a problem, we defined high-level distilled knowledge as the flow for solving a problem. Because a DNN uses many layers sequentially to map from the input space to the output space, the flow of solving a problem can be defined as the relationship between features from two layers.

Gatys et al. [6] used the Gramian matrix to represent the texture information of the input image. Because the Gramian matrix is generated by computing the inner product of feature vectors, it can contain the directionality between features, which can be thought of as texture information. Similar to Gatys et al. [6], we represented the flow of solving a problem by using Gramian matrix consisting of the inner products between features from two layers. The key difference between the Gramian matrix in [6] and ours is that we compute the Gramian matrix *across* layers, whereas the Gramian matrix in [6] computes the inner products between features *within* a layer. Figure 1 shows the concept diagram of our proposed method of transferring distilled knowledge. The extracted feature maps from two layers are used to generate the flow of solution procedure (FSP) matrix. The student DNN is trained to make its FSP matrix similar to that of the teacher DNN.

Distilling the knowledge is a useful technique for various tasks. In this study, we verified the usefulness of the proposed distilled knowledge by using it to perform three tasks. The first was fast optimization. A DNN that understands the flow of solving a problem can be a good initial weight for solving a main task and can learn faster than a normal DNN. Fast optimization is a very useful technique. Researchers have focused on achieving fast optimization not only by using advanced learning rate scheduling techniques [13, 27, 4] but also by finding good initial weights [5, 9, 18, 20]. Our approach is based on the initial weight method, so we only compared it with other initial weight methods. We compared the number of training iterations and performance of our scheme with various other techniques.

The second task was to improve the performance of a small network, which is a shallow network with fewer parameters. Because a small network learns distilled knowledge from the teacher network, it is more powerful than using the student network alone without help from the teacher network. We compared the performance of the original network and a network using various knowledge transfer techniques.

The third task was transfer learning. Although a new task may provide only a small dataset, transfer learning can take advantage of a deep and heavy DNN pretrained with a huge dataset [2]. Because our proposed method has the advantage of being able to transfer the distilled knowledge to a small DNN, the small network can perform similarly to a large DNN that uses a normal transfer learning method.

Our paper makes the following contributions: 1. We propose a novel technique to distill knowledge. 2. This approach is useful for fast optimization. 3. Using the proposed distilled knowledge to find the initial weight can improve the performance of a small network. 4. Even if the student DNN is trained at a different task from the teacher DNN, the proposed distilled knowledge improves the performance of the student DNN.

2. Related Work

Knowledge Transfer Deep networks with many parameters usually perform well in computer vision tasks. The depth of most architectures is being increased to improve performance. When deep learning first began, Alexnet [16] had only five convolution layers. However, the recent well-known network GoogleNet [23] has 22 convolution layers, and the residual network [8] has 152 layers.

A deep network with many parameters requires heavy computation for both training and testing. These deep networks are difficult to use in real-life applications because a normal computer cannot handle this work, let alone mobile devices. Therefore, many researchers have been trying to make networks smaller while maintaining the performance level. A typical way is to distill knowledge from trained deep networks and transfer it to a small network that can be used without large storage and heavy computation. Recently, Hinton et al. [11] introduced the model compression method based on the concept of dark knowledge. It uses a softened version of the final output of a teacher network to teach information to a small student network. With this teaching procedure, a small network can learn how a large network studied given tasks in a compressed form. Romero et al. [20] used not only the final output but also intermediate hidden layer values of the teacher network to train the student network and showed that using these intermediate layers can improve the performance of deeper and thinner student networks. Net2Net [3] also uses a teacher–student network system with a function-preserving transform to initialize the parameters of the student network according to the parameters of the teacher network.

Fast Optimization A deep convolutional neural network (CNN) takes a relatively long time to reach its global optimum or a good local optimum. It is easy to train small datasets such as MNIST [17] or CIFAR10 [15]. In cases of big datasets like the ILSVRC datasets [21], however, a big network can take a few weeks for training. Therefore, fast optimization has become another important subject of research recently. There are several different approaches to fast optimization, such as finding good initial weights or reaching the optimal point with a different technique than the standard stochastic gradient descent (SGD) method.

In the early days, initialization by Gaussian noise with a zero mean and unit variance became very popular. Other various initialization techniques such as Xavier initialization [7] are also used widely. However, these simple initializations are poor at training very deep networks. Therefore, some new techniques [18, 22, 14] have appeared that are based on mathematical approaches. With good initialization, if training starts at an appropriate location, then the parameters can rapidly reach the global optimum.

Optimization algorithms have also evolved with the development of deep learning. Conventionally, the SGD algorithm is widely used as a baseline. However, using SGD can make it difficult to escape from many saddle points. Because of this problem, several other algorithms have been suggested [13, 27, 4]. These algorithms help with getting out of saddle points and reaching the global optimum quickly.

Transfer Learning Transfer learning is a simple technique of modifying the parameters of an already trained network to adapt to a new task. Typically, input-side layers that play the role of feature extraction are copied from a pretrained network and kept frozen or fine-tuned, whereas a top classifier for the new task is randomly initialized and then trained at a slow learning rate. Fine-tuning often outperforms training from scratch because the pretrained model already has a great deal of information. For example, many researchers [19, 28, 1, 2] have recently used a model pretrained with the ILSVRC dataset to extract visual features from an image and fine-tuned the model to improve the final accuracy for the VQA [1] and CUB200 [25] tasks. Many other tasks such as detection and segmentation also use this ImageNet pre-trained model for the initial values of the model because the ILSVRC dataset can be helpful for generalization. Our approach also uses this fine-tuning technique with our own good initialization method.

3. Method

The main concept of our proposed method is how to define the important information of the teacher DNN and transfer the distilled knowledge to the other DNN. This section is divided into four parts to describe our main concept. Sec. 3.1 presents the useful distilled knowledge that we used in this study. Sec. 3.2 introduces the mathematical expression of our proposed distilled knowledge. Based on the carefully designed distilled knowledge, we define the loss term in Sec. 3.3. Finally, Sec. 3.4 presents the whole learning procedure of the student DNN.

3.1. Proposed Distilled Knowledge

The DNN generates features layer by layer. Higher layer features are closer to the useful features for performing a main task. If we view the input of the DNN as the question and the output as the answer, we can think of the generated features at the middle of the DNN as the intermediate result in the solution process. Following this idea, the knowledge transfer technique proposed by Romero et al. [20] lets the student DNN simply mimic the intermediate result of the teacher DNN. However, in the case of the DNN, there are many ways to solve the problem of generating the output from the input. In this sense, mimicking the generated features of the teacher DNN can be a hard constraint for the student DNN.

In the case of people, the teacher explains the solution process for a problem, and the student learns the flow of the solution procedure. The student DNN does not necessarily have to learn the intermediate output when the specific question is input but can learn the solution method when a specific type of question is encountered. In this manner, we believe that demonstrating the solution process for the problem provides better generalization than teaching the intermediate result.

3.2. Mathematical Expression of the Distilled Knowledge

The flow of the solution procedure can be defined by the relationship between two intermediate results. In the case of a DNN, the relationship can be mathematically considered by the direction between features of two layers. We designed the FSP matrix to represent the flow of the solution process. The FSP matrix $G \in \mathbb{R}^{m\times n}$ is generated by the features from two layers. Let one of the selected layers generate the feature map $F^1 \in \mathbb{R}^{h\times w\times m}$, where h, w, and m represent the height, width, and number of channels, respectively. The other selected layer generates the feature map $F^2 \in \mathbb{R}^{h\times w\times n}$. Then, the FSP matrix $G \in \mathbb{R}^{m\times n}$ is calculated by

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

where x and W represent the input image and the weights of the DNN, respectively. We prepared residual networks with 8, 26, 32 layers that were trained with the CIFAR-10 dataset. There are three points in the residual network for the CIFAR-10 dataset where the spatial size changes. We selected several points to generate the FSP matrix, as shown in Figure 2.

3.3. Loss for the FSP Matrix

In order to help the student network, we transfer distilled knowledge from the teacher network. As described before, the distilled knowledge is represented in the form of an FSP matrix that contains information about the flow of the solution procedure. We can assume that there are n FSP matrices G_i^T, $i = 1, \ldots, n$, which are generated by the teacher network, and n FSP matrices G_i^S, $i = 1, \ldots, n$, which are generated by the student network. In this study, we only considered a pair of FSP matrices between the teacher and student networks (G_i^T, G_i^S), $i = 1, \ldots, n$ with the same spatial size. We took the squared L2 norm as the cost function for each pair. The cost function of transferring the distilled knowledge task is defined as

$$\begin{aligned} &L_{FSP}(W_t, W_s) \\ &= \frac{1}{N}\sum_x \sum_{i=1}^{n} \lambda_i \times \|(G_i^T(x;W_t) - G_i^S(x;W_s)\|_2^2, \quad (2) \end{aligned}$$

where λ_i and N represent the weight for each loss term and the number of data points, respectively. We assumed that whole loss terms are equally significant. Therefore, we used the same λ_i for all experiments.

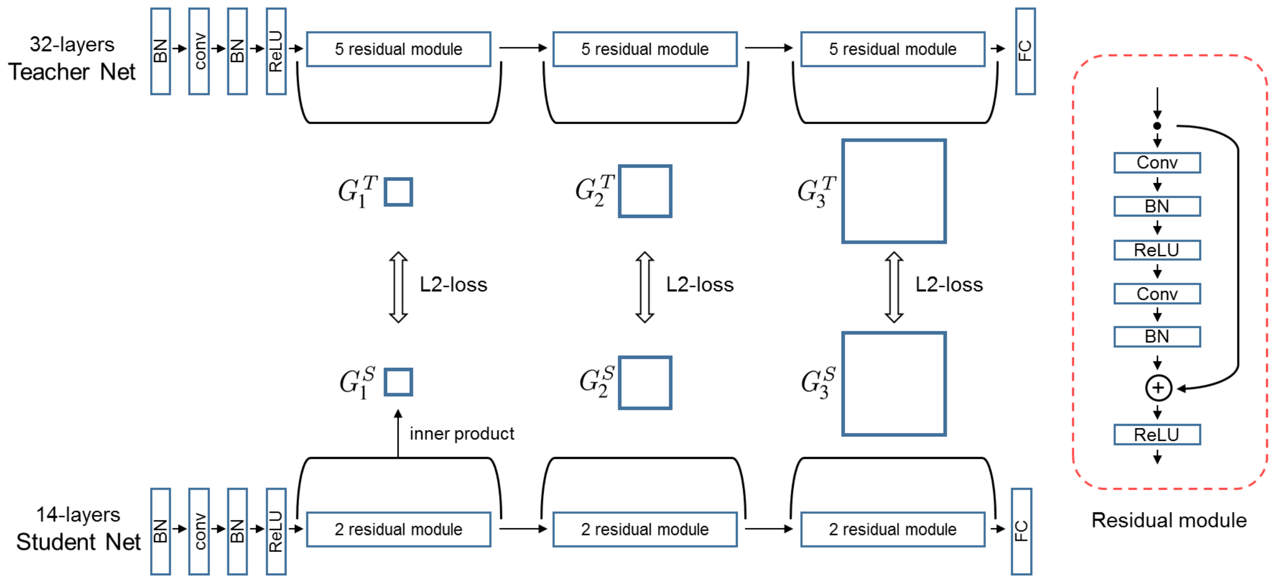

Figure 2. Complete architecture of our proposed method. The numbers of layers of the teacher and student networks can be changed. The FSP matrices are extracted at the three sections that maintain the same spatial size. There are two stages of our proposed method. In stage 1, the student network is trained to minimize the distance between the FSP matrices of the student and teacher networks. Then, the pretrained weights of the student DNN are used for the initial weight in stage 2. Stage 2 represents the normal training procedure.

3.4. Learning Procedure

Our transfer method uses the distilled knowledge generated by the teacher network. To clearly explain what the teacher network represents in our paper, we define two conditions. First, the teacher network should be pretrained by some dataset. This dataset can be the same or different from the one that the student network will learn. The teacher network uses a different dataset from that of the student network in the case of a transfer learning task. Second, the teacher network can be deeper or shallower than the student network. However, we consider a teacher network that is the same or deeper than the student network.

The learning procedure contains two stages of training. First, we minimize the loss function L_{FSP} to make the FSP matrix of the student network similar to that of the teacher network. The student network that went through the first stage is now trained by the main task loss at the second stage. Because we used the classification task to verify the effectiveness of our proposed method, we can use the softmax cross entropy loss L_{ori} as the main task loss. The learning procedure is explained below in Algorithm 1.

```
Algorithm 1 Transfer the distilled knowledge
Stage 1: Learning the FSP matrix
Weights of the student and teacher networks: $W_s$, $W_t$
  1: $W_s = \arg\min_{W_s} L_{FSP}(W_t, W_s)$
Stage 2: Training for the original task
  1: $W_s = \arg\min_{W_s} L_{ori}(W_s)$
```

4. Experiments

We conducted three experiments to verify the effectiveness of our proposed knowledge transfer technique. For all experiment settings, we used a deep residual network [8] for the base architecture. Interestingly, the deep residual network has shortcut connections to make an ensemble structure [24]. Furthermore, the shortcut connections allow training of much deeper networks. Because of these two reasons, many researchers use the residual network for various tasks. Figure 2 shows the base architecture of the deep residual network. There are several sections that maintain the same spatial size of feature maps by using the zero padding. For example, the deep residual network in this figure consists of three sections. Although there are no constraints on how to select the two layers to make the FSP matrix, we selected the first and last layers in a section. Furthermore, because the FSP matrix can be generated by two layer features with the same spatial size, we used the max pooling layer to make the same spatial size if the sizes of two layer features are different.

We used three representative tasks to verify the usefulness of the proposed knowledge transfer technique. By learning the flow of the solution procedure, the student network can study a task faster than usual, as discussed in Sec. 4.1. Furthermore, the FSP matrix generated by the teacher network allows the student network to outperform a student network that is trained alone, as described in Sec. 4.2. We considered the case where the teacher and student networks are trained by the same dataset for the same task. Sec. 4.3 expands on the ideas for application by dealing with the transfer learning task.

For all experiments, we compared the proposed method with the existing knowledge transfer method, FitNet [20]. For the first stage of FitNet, hint-based training was implemented by minimizing the L2 loss between outputs of the two layers during 35 000 iterations, where the hint and guided layer were set to the middle layer of each DNN. The learning rate started with 1e-4 initially. Then, it was changed to 1e-5 after 25 000 iterations. To ensure a fair comparison of the recognition accuracy, the FitNet in the second stage also had the same learning rate policy and training iterations as the proposed method. At this stage, the softening factor tau was set to 3, and the value of lambda in the KD loss function was linearly decreased from 4 to 1.

4.1. Fast optimization

Because recent DNNs have become deeper to increase performance, the training procedure takes many days [26, 8]. Furthermore, although a DNN takes a long time to train, many researchers use an ensemble of DNNs to outperform the performance of single DNN [23]. In this case, if we use an ensemble of n DNNs, training takes n times longer. Because of this, interest in the fast optimization technique has been rising in recent years.

We first prepared the teacher DNN with the normal training procedure. The teacher DNN was used to train the student DNNs with the learning procedure described in Sec. 3.4. By using one teacher network, we generated multiple student networks. The goal of the proposed fast optimization technique is to reach a similar performance with the ensemble of student networks as that of the teacher network by using less training time than the normal training procedure.

4.1.1 CIFAR-10

The CIFAR-10 dataset [15] contains 50 000 training images with 5000 images per class and 10 000 test images with 1000 images per class. The CIFAR-10 dataset comprises 32×32 pixel RGB images with 10 classes. However, we padded 4 pixels on each side to make the image size 40×40 pixels. Randomly cropped 32×32 pixel images were used for training, and the original 32×32 pixel images were used for testing.

We used a residual network with 26 layers for the teacher DNN, which provided 92% accuracy for the CIFAR-10 dataset as reported in [8]. Furthermore, we used the same structure of the teacher DNN for the student DNN. The teacher network was trained according to the parameters described below. For all experiments, we used a batch size of 256. The learning rate started with 0.1, was changed to 0.01 and 0.001 at 32 000 and 48 000 iterations, respectively, and terminated at 64 000 iterations. We used a weight decay

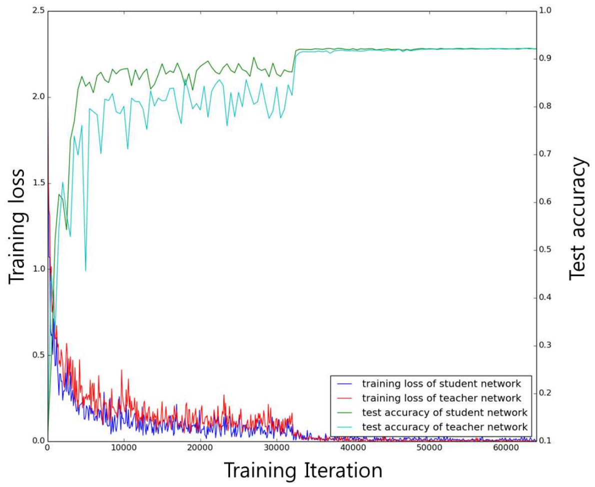

Figure 3. Analysis of the optimization speed and test accuracy. We compared the teacher DNN and student DNN that learned the distilled knowledge (i.e., the FSP matrix).

of 0.0001 and momentum of 0.9 with the MSRA initialization [9] technique and BN [12].

A student network with the same structure as the teacher network was used in stage 1 to set the initial weights as described in algorithm 1. We used learning rates of 0.001, 0.0001, and 0.00001 until 11 000, 16 000, and 21 000 iterations, respectively. We used a weight decay of 0.0001 and momentum of 0.9. We then trained the student DNN by using the normal procedure and the initial weights provided at the end of stage 1. Note that we trained several student networks in stage 2 by using the same initial weights provided by the student network trained in stage 1. As the result of stage 1 is copied to many student networks as initial weights, stage 1 is an efficient way of initializing many student networks. One potential drawback of sharing the same initial weights across all student networks is that the networks can be more correlated than if the student networks are independently initialized.

Figure 3 represents the test accuracy and change in the training loss over time. The student network showed faster optimization than the teacher network. The student network was three times faster at reaching the saturation region than the teacher network. Because we used the MSRA initialization technique for the teacher network, which is not a naive initialization method but a high performance method, we believe that the FSP matrix provides good distilled knowledge for initializing the weights of the student network.

We trained student networks with one-third of the original number of iterations in stage 2 to demonstrate the fast optimization. In stage 2, we used learning rates of 0.1, 0.01, and 0.001 until 11 000, 16 000, and 21 000 iterations, which are less than one-third the original number of iterations. From the results in Table 1, we observed that using one-third of the iterations was enough for the student networks with the proposed method. Although the student networks used fewer iterations, the proposed method outperformed the FitNet as well as the original teacher network.

We also experimented with the FitNet method of taking three losses applied to three intermediate layers as well as one loss applied to the middle layer only. It turned out that the one-loss FitNet outperformed the three-loss FitNet as in Table 1.

The proposed method can decompose the entire network into several modules, and each module's behavior is captured by its FSP matrix. If the student's FSP matrix of a module is similar to that of the teacher network, it implies that the module in the student network behaves similarly to the corresponding module in teacher network. Further, each module can be trained independently in that the module can be trained from the correlations between inputs and outputs of that module alone, even though the other modules are not fully trained. In contrast, the three-loss FitNet's upper modules, which are trained by matching only the outputs of the module without considering the relation between the input and the output, are less efficiently trained until the modules below in the student network are sufficiently trained so that the input to the upper module begins to be meaningful. This explains why the one-loss FitNet outperformed the three-loss FitNet. For the three-loss FitNet, the network had four modules. The second and third modules would be difficult to train by the intermediate results. In addition, FSP is less restrictive than FitNet. If the student network and teacher network have the same intermediate feature maps, they will have the same FSP matrix. However, the converse is not true, which allows diversity in feature maps given the same FSP matrix.

As both teacher networks and student networks are of the same architecture, one can also transfer knowledge by directly copying weights. We also compared the proposed method and the knowledge transfer by copying weights. To this end, we simply trained three copies of one teacher network for additional 21 000 iterations, which is equivalent to copying the weights from a single teacher network and starting from there. This did not provide a better result than student*. As given in Table 1 (Teacher ‡), the individual performances were slightly better than the original teacher performance, but they provided a poor ensemble performance. FSP is less restrictive than copying the weights and allows for better diversity and ensemble performance.

In addition, the ensemble of student networks with fewer iterations provided a similar performance as the ensemble of teacher networks, but FitNet did not. Although the ensemble performance of the student networks was close to the ensemble performance of the teacher networks, there was an obvious loss in gain achieved by the former (92.14 $\rightarrow$ 93.26) compared to the gain achieved by the latter (91.75 $\rightarrow$ 93.48). This is because the student networks were more closely correlated from sharing initial weights.

	Net 1	Net 2	Net 3	*Avg*	*Ensemble*	*#Iter*
Teacher	91.61	91.56	92.09	91.75	93.48	192k
Teacher *	90.47	90.83	90.62	90.64	92.6	63k
Teacher ‡	91.84	92.26	92.01	92.04	92.71	63k
1 loss FitNet [20]*	91.69	91.85	91.64	91.72	92.98	98k
3 loss FitNet [20]*	88.90	89.35	89.02	89.09	89.92	98k
Student *	92.28	92.08	92.07	92.14	93.26	84k
Student *†	92.28	91.89	92.08	92.08	93.67	126k

Table 1. Recognition rates (%) on CIFAR-10. The symbol * indicates that each network was trained with 21 000 iterations, which is less than one-third of the iterations for the original case, which used 64 000 iterations. Student * was trained with 21 000 iterations in stage 1, whose results are copied to net 1, net 2, and net 3, and each student network was trained with 21 000 iterations in stage 2 to result in total 84 k iterations. The symbol ‡ represents the teacher network trained with 21 000 iterations, which started from the one of the teacher network trained with 64 000 iterations. The symbol † indicates the student network that learned the randomly shuffled FSP matrix in stage 1. In the case of Student * †, each net was trained with 21 000 iterations in stage 1 and 21 000 iterations in stage 2.

Interestingly, we developed a very simple but effective way to train less correlated student networks using the same single teacher network. The idea is that we can generate multiple FSP matrices that are essentially equivalent but apparently different. By using apparently different FSP matrices for the student networks instead of sharing the same FSP matrix, we can reduce the correlation among student networks. The FSP matrix was generated by features from two selected layers. Note that we can permute feature channels in the teacher network to obtain an equivalent teacher network that behaves essentially the same way. This means that the rows or columns of the FSP matrix can be shuffled without affecting the transfer of the distilled knowledge. The different FSP matrices obtained by row and column shuffling can be used in stage 1 to generate multiple student networks with different initial weights. Then, after stage 2, the resulting student networks would be less correlated, and an improved ensemble can be obtained. As indicated in Table 1, despite the fewer iterations, the ensemble of student networks using a randomly shuffled FSP matrix outperformed even the ensemble of teacher networks.

In terms of the training times instead of the number of iterations, the original model took 16 s/100 iterations, while the proposed model took 35 s/100 iterations for stage 1. Therefore, in terms of the total learning time, it took 8.6 h to train three teacher DNNs with the original method and take 4.84 h to train three student DNNs with the proposed method. The latter is 1.78 times faster. However, by learning more efficiently (e.g., storing the FSP matrices and using it directly instead of calculating the FSP matrix every time (took 19 s/100 iterations)) student * and student *† could be trained 2.18 and 1.39 times faster, respectively.

	Net 1	Net 2	Net 3	*Avg*	*Ensemble*	*#Iter*
Teacher	64.06	64.19	64.21	64.15	69.3	192k
Teacher *	61.29	61.26	61.41	61.32	67.2	63k
FitNet [20]*	62.85	62.46	62.35	62.55	67.6	98k
Student *	64.66	64.64	64.65	64.65	68.8	95k

Table 2. Recognition rates (%) on CIFAR-100. The symbol * indicates that the network was trained with one-third of the iterations for the original case, which used 64 000 iterations.

4.1.2 CIFAR-100

The CIFAR-100 dataset uses 50 000 training images with 500 images per class and 10 000 test images with 100 images per class. The CIFAR-100 dataset contains 32 $\times$ 32 pixel RGB images with 100 classes. Because of the small number of images per class with 100 classes, we used a residual network with 32 layers and four times as many channels as the one described in Sec. 4.1.1.

We did not use augmentation methods unlike the CIFAR-10 case to make the various experiment settings. The teacher and student networks used the same parameters as those described in Sec. 4.1.1. The only difference was that we used learning rates of 0.001, 0.0001, and 0.00001 until 16 000, 24 000, and 32 000 iterations, respectively, in stage 1.

Table 2 presents the recognition rates for different settings. Each setting was performed three times. The second column from the right shows the performance of the ensemble of three DNNs. Considering the difference in accuracy between the prepared deep residual network with 32 layers with an average of 64.15% accuracy and the same network trained with one-third of the original iterations with an average of 61.32% accuracy, we can conclude that the number of iterations is important for high performance. However, even though the student network used fewer iterations for training, the student network that used the distilled knowledge from the teacher network generated a similar performance as the original teacher network.

We compared the other distilled knowledge transfer method FitNet with our proposed method. As given in Table 2, the student network with FitNet outperformed the teacher network with fewer iterations. However, when an ensemble of three networks was used, the teacher network with fewer iterations and student network with FitNet had similar accuracies. There was not that much improvement. In terms of the performance and number of iterations, the proposed method was more efficient than the existing method of FitNet, as presented in Table 2.

4.2. Performance improvement for the small DNN

Recently, many researchers have used a very deep neural network with a huge number of parameters for high performance. For example, one residual network uses more than 1000 layers for the classification task [10]. The wide residual network [26] increases the width of the residual network. However, there are many defects. Even if we use the DNN as an inference network, we have to prepare a high-performance system, which is very expensive. Furthermore, many iterations are needed to train a deep and wide neural network. It is also expensive. Thus, methods to improve the performance of a small DNN are very important.

	Accuracy
Teacher-original	91.91
Student-original	87.91
FitNet [20]	88.57
Proposed Method	88.70

Table 3. Recognition rates (%) on CIFAR-10. We used a residual DNN with 8 layers for the student DNN and 26 layers for the teacher DNN.

We conducted experiments to verify that our proposed method can be used with DNNs of different sizes. The goal of our proposed method is to improve the performance of a small student network by learning the distilled knowledge of a deep teacher network. Once again, we defined a small network as a shallow network with few weights. As shown in Figure 2, the teacher DNN was deeper than the student DNN. The student DNN was constructed by simply reducing the number of residual modules in the teacher DNN. Therefore, the student DNN used fewer parameters than the teacher DNN.

The learning procedure was the same as that described in Sec. 4.1. Because the student DNN and teacher DNN had the same number of channels, the sizes of the FSP matrices were the same. By minimizing the distance between the FSP matrices of the student network and teacher network, we found a good initial weight for the student network. Then, the student network was trained to solve the main task.

4.2.1 CIFAR-10

We used a residual network with 26 layers for the teacher DNN and a residual network with 8 layers for the student DNN. For the parameter settings and learning procedure, we use the same parameters as described in Sec. 4.1.1 but not the same training iterations for stage 2. The student DNNs used the same number of training iterations as the teacher DNN.

For fair comparison, we prepared a student DNN that was trained from scratch. As indicated in Table 3, the methods of transferring distilled knowledge outperformed the student DNN that used the original learning procedure. This means that distilled knowledge from the teacher DNN can be useful information for even a shallow student DNN. We can conclude that the proposed method is more useful than the existing method.

4.2.2 CIFAR-100

We also verified the network minimization ability of proposed method at the CIFAR-100 dataset. As a similar exper-

	Accuracy
Teacher-original	64.06
Student-original	58.65
FitNet [20]	61.28
Proposed Method	63.33

Table 4. Recognition rates (%) on CIFAR-100. We used a residual DNN with 14 layers for the student DNN and 32 layers for the teacher DNN.

imental setting to Sec. 4.1.2, we used residual networks with 32 and 14 layers for the teacher DNN and student DNN, respectively. For all experiments in this section, we used the full 64 000 iterations.

Table 4 presents the recognition rates for different settings. Because we did not use any augmentation methods, the teacher DNN showed the 64% accuracy. Furthermore, the student DNN that used the normal learning method showed a 58.65% recognition rate. Surprisingly, the proposed method made the student network generate the similar performance to the teacher DNN. The existing knowledge distillation method (i.e., FitNet) also showed improved performance. However, when we compared the performance of the student network with two distilled knowledge methods and the student network with the original method, the proposed method with distilled knowledge clearly performed better than the existing ones.

4.3. Transfer Learning

In this section, we explain the applications to which the proposed methods can be applied. The teacher DNN and student DNN can learn not only the same task, but also different tasks. To deal with this problem, we focused on the transfer learning task. Transfer learning is widely used when the dataset is too small to generate useful features. In this case, most existing methods use a pretrained DNN that is trained by a huge dataset, such as ImageNet dataset [21]. However, the most important issue is that most existing methods directly use the pretrained DNN, which contains many layers and a huge number of weights. This means that a high-quality machine needs to be prepared to improve the performance with a small dataset. Therefore, because the distilled knowledge can be transferred to a small DNN, the knowledge transfer technique can be the effective solution for this problem.

We prepared a 34-layer residual DNN [8] that was pretrained with the ImageNet dataset. For the different task containing a small number of images, we used the Caltech-UCSD Birds (CUB) 200-2011 dataset [25]. The CUB 200-2011 dataset contains 11,788 images of 200 bird subordinates. Because of the small number of images per class, it is difficult to generate a high level of performance by using only this dataset. As given in Table 5, although we used the deep structure of the 34-layer residual DNN, the accuracy was very poor when we trained it from scratch.

	Accuracy
Teacher - fine tuning	77.72
Teacher - training from scratch	47.53
Student - training from scratch	47.73
FITNET [20]	70.19
Proposed Method	74.26

Table 5. Recognition rates (%) on CUB200. We used a residual DNN with 20 layers for the student DNN and 34 layers for the teacher DNN.

For the shallow DNN, we prepared a 20-layer residual DNN. The 34-layer residual DNN consisted of four parts that generated features of the same spatial size in the same part. The four parts contained three, four, six, and three residual modules. The prepared student DNN (20-layer residual DNN) contained two, two, three, and two residual modules, respectively. For all settings, we used learning rates of 0.1, 0.01, and 0.001 up to 10 000, 20 000, and 30 000 iterations. The fine-tuning technique usually uses small learning rates. However, because we found that the base learning rate of 0.1 was better than the learning rate of 0.001, we decided to report the results of 0.1.

Because proposed methods have to transfer the FSP matrix of the teacher DNN to the student DNN, we used learning rates of 0.1, 0.01, and 0.001 up to 11 000, 16 000, and 21 000 iterations for stage 1. In this stage, we extracted the FSP matrices at each part of the DNN. As reported in Table 5, the proposed method generated a high level of performance close to that of the teacher DNN with fine-tuning methods. Considering that the student DNN was 1.7 times shallower than the teacher DNN, we believe that proposed method is an effective technique for transferring the knowledge even to a different task.

5. Conclusion

We proposed a novel approach to generate distilled knowledge from the DNN. By determining the distilled knowledge as the flow of the solving procedure calculated with the proposed FSP matrix, the proposed method outperforms state-of-the-art knowledge transfer methods. We verified the effectiveness of our proposed method in three important aspects. The proposed method optimizes the DNN faster and generates a higher level of performance. Furthermore, the proposed method can be used for the transfer learning task.

Acknowledgement

This work was partly supported by the ICT R&D program of MSIP/IITP, 2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion and the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) (No. CRC-15-05-ETRI).

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015. 1, 3

[2] S. Branson, G. Van Horn, S. Belongie, and P. Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014. 2, 3

[3] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015. 2

[4] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 2

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010. 2

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 1, 2

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1, 2, 4, 5, 8

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. 2, 5

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 7

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[14] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015. 2

[15] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 2, 5

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[18] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 2

[19] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. *arXiv preprint arXiv:1511.05756*, 2015. 1, 3

[20] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *In Proceedings of ICLR*, 2015. 1, 2, 3, 5, 6, 7, 8

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 8

[22] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013. 2

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1, 2, 5

[24] A. Veit, M. Wilber, and S. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *arXiv preprint arXiv:1605.06431*, 2016. 4

[25] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011. 3, 8

[26] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5, 7

[27] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 2

[28] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015. 3