24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Feature-level Ensemble Knowledge Distillation for Aggregating Knowledge from Multiple Networks

SeongUk Park[1] and **Nojun Kwak**[2]

Abstract. Knowledge Distillation (KD) aims to transfer knowledge in a teacher-student framework, by providing the predictions of the teacher network to the student network in the training stage to help the student network generalize better. It can use either a teacher with high capacity or an ensemble of multiple teachers. However, the latter is not convenient when one wants to use feature-map-based distillation methods. In this paper, we empirically show that using several non-linear transformation layer cope well with multiple-teacher setting compared to other kinds of feature-map-level distillation methods. Comprehensively, this paper proposes a versatile and powerful training algorithm named FEature-level Ensemble knowledge Distillation (FEED), which aims to transfer the ensemble knowledge using multiple teacher networks. In this study, we introduce a couple of training algorithms that transfer ensemble knowledge to the student at the feature-map-level. Among the feature-map-level distillation methods, using several non-linear transformations in parallel for transferring the knowledge of the multiple teachers helps the student find more generalized solutions. We name this method as parallel FEED, and experimental results on CIFAR-100 and ImageNet show that our method has clear performance enhancements, without introducing any additional parameters or computations at test time. We also show the experimental results of sequentially feeding teacher's information to the student, hence the name sequential FEED, and discuss the lessons obtained. Additionally, the empirical results on measuring the reconstruction errors at the feature map give hints for the enhancements.

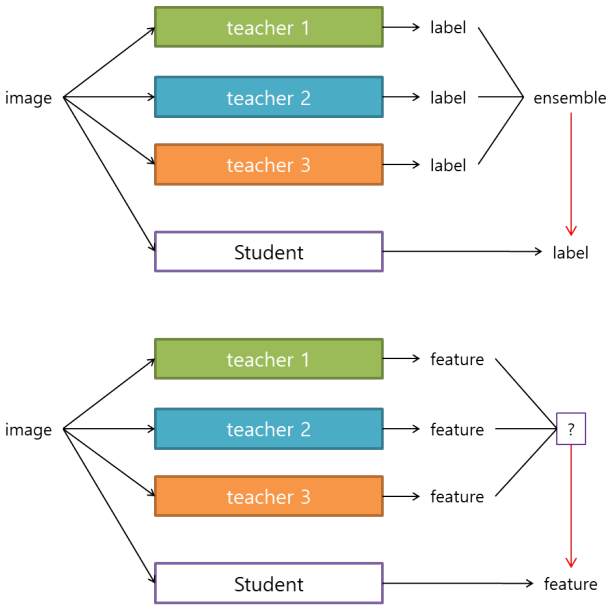


Figure 1. Our problem formulation. The figure at the top shows how the KD is trained using ensemble of networks as the teacher. The figure at the bottom shows that a problem arises when we want to distill ensemble information directly in feature level rather than using label information.

1 Introduction

Recent successes of convolutional neural networks (CNNs) have led to the use of deep learning in real-world applications. In order to manipulate these deep learning models, deep CNNs are trained using multi-class datasets to find manifolds separating different classes well. To meet this need, deep and parameter-rich networks have emerged that have the power to find manifolds for a large number of classes. However, these deep CNNs suffer from the problem of overfitting due to their great depth and complexity, which results in a drop of performance at the test time. In fact, even a small ResNet applied for a dataset such as CIFAR-100 [14] will easily overfit with the converged train losses, whereas the test accuracy is significantly lower. These phenomena have led to the need for learning DNN models with appropriate regularization to allow them to generalize better. Regularizing a model to achieve high performance for new inputs is a technique that has been used since the era of early machine learning.

[1] Seoul National University, South Korea, swpark0703@snu.ac.kr
[2] Seoul National University, South Korea, nojunk@snu.ac.kr

Model ensemble [4] is one of the popular regularization methods, which has been used as a way of alleviating the problem of overfitting in a single model. However, it has drawbacks in that it requires multiple models and inputs should be fed to each of them at the test time. Many studies proposed ideas to transfer knowledge of a teacher to a compact student [1, 3, 22, 26]. For a solution to this problem, [11] proposed *Knowledge Distillation* (KD), which trains a student network using soft labels from an ensemble of multiple models or a teacher network with high capacity. They obtained meaningful results in the speech recognition problem, and KD has become one of the representative methods of *knowledge transfer*. They aim performance improvements of a weak student network by giving various forms of knowledge of expert teacher networks. It is also categorized as one family of model compression since it helps the student network achieve higher accuracy with a fixed number of parameters.

The recent knowledge transfer algorithms can be approximately categorized in two viewpoints. The first is whether to use an ensemble model or a single high-capacity model as a teacher. The second is whether the teacher delivers predicted labels or the information from feature maps. Whereas the methods that use teacher's predic-

tion can use both types of teachers, to the best of our knowledge, methods using feature-map-level information can only use a single high-capacity model as a teacher. For example, studies of *Factor Transfer* (FT) [13], *Attention Transfer* (AT) [31], and *Neuron Selectivity Transfer* (NST) [12] set the student network as a shallow one with a small number of parameters, and set the teacher network as a deeper and more powerful one instead of an ensemble expert. One of the drawbacks of methods with a high-capacity teacher model is that high-capacity model may be hard to obtain [15].

On the contrary, the ones which use an ensemble of networks can transfer ensemble knowledge and also have advantages of the peer-teaching framework [7, 11, 15]. Also, [33] showed that using the same type of network for transferring output-level knowledge can improve the performance of a network. However, the methods that deliver knowledge at the feature-map level also have advantages that they can give more specific information to the student compared to the methods that only rely on the output predictions of the teacher.

To make full advantages of both the ensemble teacher method and the feature map transfer method, we propose a new framework that delivers knowledge of multiple networks at the feature-map-level (See Fig. 1). In this paper, we train a new student network using multiple teachers that share the same architecture with the student network, named parallel FEED. We also explore a variant, named sequential FEED, which recursively trains a new teacher. Comparisons with other algorithms are provided with analysis that provides lessons. Our main contributions are threefold:

- In various settings, we empirically show that for high-capacity student, feature-map-based methods that give more specific knowledge are stronger than label-based methods, which give more abstract knowledge.
- We propose parallel FEED, a method that allows multiple teacher networks to be used for knowledge distillation at the feature-map level with non-linear transformations.
- For qualitative analysis, by utilizing the autoencoder reconstruction losses, we provide hints for the performance enhancements of our FEED.

The paper is organized as follows. First, we briefly explain the related works in the area of knowledge distillation. Then our main proposed method is described and a variant version implemented to compare with other methods is proposed. Next, we verify our proposed methods with experiments. Experimental results from our proposed training methods are compared with AT, KD, FT, BAN (Born Again Neural Network)[7] on CIFAR-100. The ImageNet dataset is also used to check the feasibility of our method on a large-scale dataset. Qualitative analysis is also provided, which is followed by discussion and conclusion.

2 Related Works

Many researchers studied the ways to train models other than using a purely supervised loss. In the early times of these studies, *Model Compression* [3] studied the ways to compress information from ensemble models in one network. Ba [1] showed that shallow feed-forward nets can learn the complex functions previously learned by deep neural nets, by minimizing L_2 loss between logits.

More recently, Hinton [11] proposed *Knowledge Distillation* (KD), which uses softened softmax labels from teacher networks in training the student network by minimizing the following loss:

$$\mathcal{L}_{KD} = (1-\alpha)\mathcal{L}_{CE}(y, \sigma(\mathbf{s})) + \alpha T^2 \mathcal{L}_{KL}\left(\sigma\big(\frac{\mathbf{s}}{T}\big), \sigma\big(\frac{\mathbf{t}}{T}\big)\right), \quad (1)$$

where the $\sigma(\cdot)$ is the softmax function, T is a temperature value that controls the softened logit. α is hyper-parameter that controls the weight between two terms. The vectors **s** and **t** are predicted output logits of the student network and the teacher network, respectively, and y is the ground-truth label. $\mathcal{L}_{CE}$ is the cross-entropy loss that is commonly used in classification problems, and $\mathcal{L}_{KL}$ is the Kullback-Leibler divergence loss.

This motivated researchers to develop many variants of it to various domains, and many researchers studied ways to better teach the student [6, 18, 19, 24, 25]. We introduce some recent research works that are potentially related to our proposed method.

Peer teaching framework: Rather recently, many papers adapt peer-teaching framework that use the same kind of network for both the teacher and the student. Geras [8] try to transfer knowledge between two networks that have almost the same number of parameters. BAN [7] shows that using the exact same architecture for the teacher and the student boosts the performance of the student network even without softening the labels. BAN uses a simpler loss term without softening both logits, which KD does, and does not even assign weights to the two terms as follows:

$$\mathcal{L}_{BAN} = \mathcal{L}_{CE}\big(y, \sigma(\mathbf{s})\big) + \mathcal{L}_{KL}\big(\sigma(\mathbf{t}), \sigma(\mathbf{s})\big). \quad (2)$$

They train the student recursively to enhance the performance further. The n^{th} student network becomes the $(n+1)^{th}$ teacher network to train the next student network. The better teacher network will teach the student network better.

Also, studies such as DML [33] and ONE [15] use the same kind of network for on-line training of peer networks with mutual KL-divergence losses.

Feature-map-based methods for knowledge transfer: Contrary to the methods that try to use labels from the teacher network, there exist studies that distill useful information directly from feature maps in various forms.

AT [31] tried to transfer the attention map of the teacher network to the student network, and got meaningful results in knowledge transfer and transfer learning tasks. Their loss term is:

$$\mathcal{L}_{AT} = \mathcal{L}_{CE}\big(y, \sigma(\mathbf{s})\big) + \beta \sum_{l=1}^{L} \| \frac{f(A_l^t)}{\|f(A_l^t)\|_2} - \frac{f(A_l^s)}{\|f(A_l^s)\|_2} \|_2, \quad (3)$$

where β is a hyperparameter that depends on the number of elements, and l denotes the l^{th} group [29] in the network. A_l^t, A_l^s are attention maps obtained from the teacher network and the student network, and $f(A) = (1/N)\sum_{n=1}^{N} a_n^2$, where N is the number of channels, and a_n is the spatial map from the n^{th} channel.

Yim [28] introduced another knowledge transfer technique for faster optimization and applied it also to transfer learning. Shen[23] tried to combine knowledge trained from different domains, and You [30] utilized ensemble of orderings of samples to teach the student network, which is very novel. Huang [12] tried to match the features of the student and teacher networks by devising a loss term MMD (Maximum Mean Discrepancy).

FT [13] uses additional paraphraser and translator networks, which help training the student network and got meaningful results. Their loss terms are:

$$\mathcal{L}_{rec} = \|x - P(x)\|_2^2, \quad (for\ the\ paraphraser) \quad (4)$$

$$\mathcal{L}_{student} = \mathcal{L}_{CE}(y, \sigma(\mathbf{s})) + \beta \| \frac{F_T}{\|F_T\|_2} - \frac{F_S}{\|F_S\|_2} \|_1, \quad (5)$$

where $P(\cdot)$ is the autoencoder-based paraphraser network and x is the input feature map for the paraphraser. F_T and F_S are the output of the paraphraser and the translator, respectively.

3 Proposed Training Algorithm

In deep CNNs, mainly due to the curse of dimensionality, the data points that lie on the data space are very sparse. For example, CIFAR datasets that contains the number of 50,000 training images and has 3,072 dimensions, so distances between each samples are very far. Necessarily, decision boundaries that determine the borders dividing classes are multitudinous, because finding boundaries that fit well to a training dataset is relatively an easy task. Even if the networks with the same architecture are trained, the learned decision boundaries cannot be the same. This is why ensemble methods usually perform better than a single model despite their structural equality. Goodfellow [9] also state that different models will not make all the same errors on the test dataset.

Consider the conditions that determine the training procedure of CNNs. They include the structure of the CNN and the choice of an optimizer, the seed of random initialization, the sequence of mini-batches, and the types of data augmentations. If one makes the same conditions for two different CNNs, their training procedure will be identical. However, we usually determine only the structure of the CNN, usually keeping others to be random. Consequently, two networks with the same structure are highly unlikely to learn the same decision boundaries.

Additionally, Kim [13] state that they resolve the 'inherent difference' between two networks. Among the inherent differences, minimizing the differences in the structure of CNNs can help better learn the knowledge of the teacher network. This has a thread of connection with that of BAN [7], which shows that using the same architecture for both the student and the teacher is actually beneficial. This motivation provides us chances to produce several modified versions of existing methods. In this section, we explain the feature-level ensemble training algorithms that are used for boosting the performance of a student network without introducing any additional calculations at the test time. The proposed method is named as FEED which is an abbreviation for the *FEature-level Ensemble knowledge Distillation*. We propose pFEED (*parallel FEED*), which we handle as our main method, that use non-linear transformation layers to distill ensemble knowledge into the student network. We also introduce sFEED (*sequential FEED*), motivated by BAN [7], which adapts the sequential training with the use of nonlinear transformations. The use of the nonlinear layers, rather than using a simple distance metric, had been explored previously in FitNet [20] and FT [13].

3.1 Parallel FEED

Assuming that distilling knowledge at the feature map level and distilling knowledge from an ensemble of multiple networks both have their advantages, we wanted to make cooperation of these two kinds of methods. Tackling this problem, we propose a training structure named *pFEED* to transfer the ensemble knowledge at the feature map level. Let us denote the number of the teacher network as N. We use different non-linear transformation layers for each teacher, which means that if there are N teachers, there are N different non-linear layers. The final feature map of the student network is fed into the non-linear layers and its output is trained to mimic the final feature map of the teacher network. In this way, we take advantages of both the ensemble model and the feature-based method. Our method is illustrated in Figure 2. If we use N different teachers, the loss term

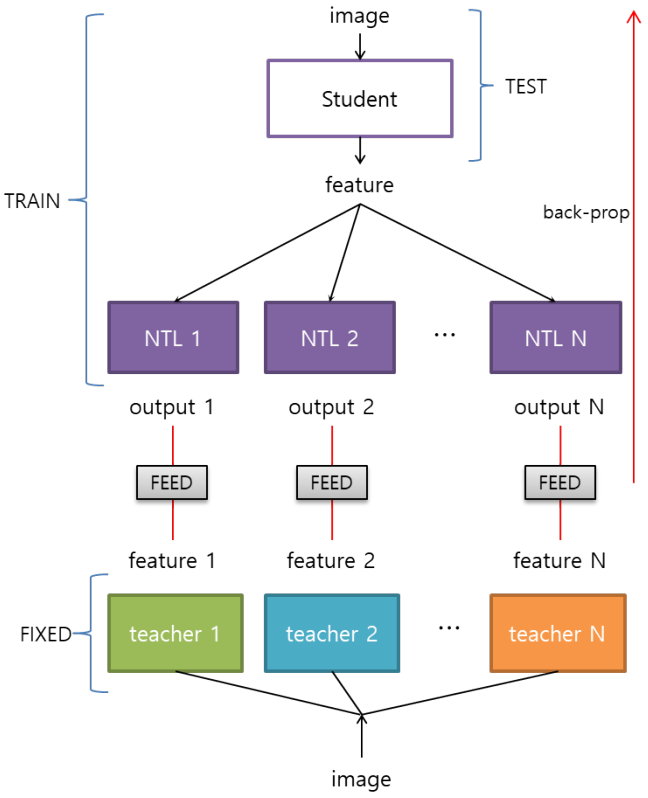


Figure 2. Illustration of our proposed method, parallel FEED. NTL is an abbreviation of Nonlinear Transformation Layer, and one NTL is allocated to each teacher network. All teacher networks are fixed during the training, and the student network and NTL networks are trained simultaneously.

is as follows:

$$\mathcal{L}_{student} = \mathcal{L}_{CE}(y, \sigma(\mathbf{s})) + \beta \sum_{n=1}^{N} \mathcal{L}_{FEED_n}, \tag{6}$$

$$\mathcal{L}_{FEED_n} = \| \frac{x_n^T}{\|x_n^T\|_2} - \frac{NTL_n(x^S)}{\|NTL_n(x^S)\|_2} \|_1. \tag{7}$$

Here, $\mathcal{L}_{CE}$ is the cross-entropy loss, y is the ground-truth label, $\mathbf{s}$ is the predicted logits, and $\sigma(\cdot)$ denotes the softmax function. $\mathcal{L}_{FEED_n}$ is the FEED loss from the n^{th} teacher network, x_n^T is the output feature map obtained from the n^{th} teacher network, and x^S is the output feature map obtained from the student network. $NTL_n(\cdot)$ is the n^{th} nonlinear transformation layer used for adapting the student with the n^{th} teacher network. Each $NTL_n(\cdot)$ is composed of three convolution layers with the kernel size of 3 to expand the size of receptive field, so that the student can flexibly merge the knowledge attained from different teachers. The feature maps are normalized by its own size as in (7). This normalizing term was previously used in AT. β is used to scale the $\mathcal{L}_1$ distance loss to match the scale of $\mathcal{L}_{CE}$, as also described in AT.

3.2 Sequential FEED

BAN [7] used cross-entropy loss combined with KD loss without softening the softmax logits. They use a trained student network as a

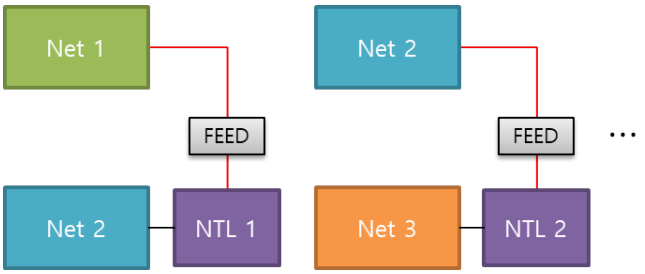


Figure 3. The sequential FEED. We adapted the setting of BANs where the teacher network and the student network have the same architecture.The trained student network is used as a teacher network for the next stage. Dots mean that we repeat this procedure.

new teacher, which is used to train a new student and do this recursively. We take this architectural advantage of using the same type of network recursively because it is a suitable model for accumulating and assembling knowledge. By performing knowledge transfer several times recursively, it may also ensemble knowledge of many training sequences. We applied the FEED recursively and named this framework as *sFEED (sequential FEED)*. The training procedure of sFEED is illustrated in Figure 3.

If the student network is trained standalone, it would perform similarly to the teacher network. However, from the view of knowledge ensemble, since the teacher network delivers feature-level knowledge different from that of the student network, the student network will benefit from it.

4 Experiments

Focus: Here, we make an assumption that labels are very abstract information, whereas feature-maps are more detailed information. For low-capacity student, giving more abstract knowledge is beneficial, since it lacks capability. However, for high-capacity giving more detailed knowledge is beneficial, since the student has enough capability. We will show with experiments that the results match our assumption.

Firstly, we want to briefly show with experiments that using nonlinear transformations for the output feature map with a distance metric is helpful for teaching the student network, rather than just using a simple distance metric without any adaptation layer. Next, we will report the score of pFEED, which use multiple pairs of nonlinear transformation layers with each teacher networks, and will compare ours with KD and AT in a similar setting. Finally, sFEED will be compared with FT, KD, and BAN. The algorithms will be experimented following the BAN's sequential training schedules.

We show the classification results on CIFAR-100 [14] on which many networks show lower test accuracy than train accuracy, so that many studies do experiments on it to show their regularization power. On this dataset, our results are compared with feature map based methods and label based methods with corresponding settings. Second, we explore the feasibility of our algorithm on Imagenet [21], a commonly used large dataset and analyze the results quantitatively. In the remaining section, we show some analysis of our algorithm. The implementation details are on the supplementary material.

We chose three types of CNNs to check the applicability of our algorithms on CIFAR-100: ResNet [10], Wide ResNet [32], and ResNext [27]. For ResNets, we chose ResNet-56 and ResNet-110 which have fewer number of parameters compared to recent CNNs. WRN28-10 is a model that controls the widen factor, with much more number of parameters. WRN28-10 achieves the best classification accuracy on CIFAR-100 among the WRNs reported in [32]. The ResNext29-16x64d also achieves the best classification accuracy on CIFAR-100 in [27]. This type of CNNs controls the cardinality of CNNs, and it has much more parameters compared to other models. For ImageNet, we used ResNet-34 to confirm the feasibility of our method on large scale datasets.

4.1 Effectiveness of standalone feature-map-level distillation losses

We compare the classification results using our FEED loss and two other kinds of loss terms, AT and simple $\mathcal{L}_1$ loss, in training the student at the feature map level. The results are shown in Table 1. The $\mathcal{L}_1$ means we simply use $\mathcal{L}_1$ loss at the final feature maps. AT use attention maps attained from feature maps to give information. From these results, we can see that using nonlinear transformation layers are helpful in delivering information at the feature-map level. Interestingly, AT and $\mathcal{L}_1$ beats a single FEED model in ResNext29-16x64d model.

Table 1. Test classification error on Cifar-100 dataset. The numbers on the scratch column are the baseline errors of each network, trained by the pure cross-entropy loss, which are the scores of the teachers. The numbers on scratch* columns are the reported errors on their original papers.

Model Type	scratch*	scratch	$\mathcal{L}_1$	AT	FEED
ResNet-56	-	28.18	27.16	26.60	26.02
ResNet-110	-	26.97	25.42	25.70	25.25
WRN28-10	19.25	19.09	17.94	17.86	17.68
ResNext29-16x64d	17.31	17.32	16.46	16.51	16.80

4.2 Parallel FEED

In this experiment on pFEED, our main experiment, we used the same type of networks that were used on previous experiments. For all four types of CNNs, we compared the classification results with the result of KD because we designed our training algorithm with the intention of distilling more ensemble-like knowledge from multiple teachers. We also experimented AT to empirically show that it would not be easy for a feature-map-based knowledge transfer methods to fully utilize multiple teacher networks. We modulated AT to use multiple teachers, by simply incrementally adding same loss terms for each teacher network. We did not change the β values for weights of each loss terms, similar to pFEED.

The results are in Table 2. The 'Scratch' column shows the performance of the base networks, which are used in KD for model ensemble, and also used as teachers in pFEED and AT. For all experiments, pFEED consistently got higher accuracy compared with both KD and AT and produces the closest results to those of the network ensemble.

Comparison with KD: It is worth noting that the performance of KD is almost equivalent to pFEED for small networks, because giving abstract knowledge performs well with smaller networks. However, when it comes to the networks with a larger number of parameters, pFEED shows better accuracy compared to KD. This result matches the assumption that distilling in the feature-map level will provide more **detailed** information to the student. In contrast, KD works quite well for small networks, because it gives ensemble labels, which are rather **abstract**. These labels are useful for small net-

Table 2. Test classification error (%) on CIFAR-100 dataset. All scores of other methods are our reproduction. In the 5x$\mathcal{L}_1$, 5xAT and 5xFT columns, we used five $\mathcal{L}_1$, AT and FT losses each for training one student. The scores of the Ens column are the performance of label ensemble of 5 scratch models. We trained pFEED 5 times and averaged the results.

Model Type	Scratch (5 mean)	KD	5x$\mathcal{L}_1$	5xAT	5xFT	5xRegressor	pFEED	Ens	Parameters
ResNet-56	28.18 ($\pm$ 0.17)	**24.69**	27.29	27.02	25.41	25.00	24.74 ($\pm$ 0.12)	22.45	0.85M
ResNet-110	26.97 ($\pm$ 0.16)	23.50	28.94	25.23	23.81	23.18	**22.98** ($\pm$ 0.20)	21.20	1.73M
WRN28-10	19.09 ($\pm$ 0.13)	18.30	23.75	17.73	17.30	17.26	**16.86** ($\pm$ 0.14)	16.59	36.5M
ResNext29-16x64d	17.32 ($\pm$ 0.08)	16.64	16.31	17.42	16.30	16.17	**15.70** ($\pm$ 0.08)	15.66	68.1M

works that should focus on key information for accuracy improvement.

Comparison with feature-map-based methods: Compare the results on Table 2 with Table 1. As shown by the scores of $\mathcal{L}_1$ and AT, using multiple teachers with these methods did not make a meaningful difference compared to the case when a single teacher network was used. This is an example of our statement, that it is not easy for existing feature-map-based knowledge transfer methods to fully utilize multiple teachers to boost the performance of a single student network.

Comparison with 5 regressors: Here, we compose an experiments of using 5 regressors from FitNets, which use single 1×1 convolution layer for the regressor, which is proposed to adapt the channel size. Since the student and teacher have the same channel sizes, following the FitNet, one can omit the regressors, which becomes the setting of 5x$\mathcal{L}_1$. However, as the result shows, Regressors which have non-linearity boosts the ensemble-feature distillation performance. Furthermore, with stacking more of this non-linearity for increasing receptive field size is beneficial, which can be found with the result of pFEED.

Imagenet: The results of pFEED for ImageNet is on Table 3. We also find some accuracy improvements on ImageNet dataset, but did not have enough resources to train models with larger parameters, nor did we could experiment on larger models or other methods. Though we reported scores of five scratch models, only the first three teacher networks had been used for training of the student. We could get decent results, but interestingly, improvements are not as strong as those of sFEED on ImageNet that will be shown later.

Table 3. Validation classification error (%) of pFEED on Imagenet dataset.

Model Type	Scratch (5 runs)					pFEED
ResNet-34(Top-1)	26.45	26.59	26.40	26.77	26.64	**25.27**
ResNet-34(Top-5)	8.54	8.72	8.63	8.68	8.61	**7.79**

4.3 Sequential FEED

This section contains results of 4 types of methods (FT, KD, BAN, and sFEED) experimented in small and large capacity networks. Note that the pFEED is our main proposed method for good performance, and the purpose of this subsection is to further delve into each method to speculate the characteristics and differences of each method. The classification results of different methods on CIFAR-100 can be found in Table 4.3. The word 'Stack' in Table 4.3 is the number of recursions that the student model is trained. We only experimented up to 5 times since all of them achieve fairly good enough accuracy compared to baseline models. We reported the performance of sFEED because it has a more similar framework to BANs, though the pFEED, our main proposed method, outperforms sFEED. The BAN-N in [7] would be the identical setting to the Stack-(N-1) in Table 4.3. We additionally experimented FT with the identically structured teacher and student networks. FT basically uses a large teacher network with a paraphraser, which is trained as an autoencoder. It is trained to extract key information called 'factor' from the teacher network in an unsupervised manner, so it gives more abstract information.

Comparison of Label-based-methods and Feature-map-based methods: For the smaller networks such as ResNet-56 and ResNet 110, methods using labels performed better than the feature-map-based methods, but for networks with larger sizes, feature-map-based methods showed higher accuracy. FT uses more abstract knowledge (using paraphraser) compared to sFEED, so it did not perform well as sFEED for larger networks. However, it performed better for smaller-sized networks. KD uses more abstract knowledge compared to BAN, because KD softens the labels, and it achieves higher accuracy for smaller networks, but BAN showed better accuracy for ResNext.

The results of sFEED for ImageNet is on Table 5. For the base model, we simply used the pre-trained model that Pytorch supplies, and could achieve the desired result that the performance of Top-1 and Top-5 accuracy improves at each Stack. The sFEED with Stack-5 achieves better performance compared to pFEED. The performance of pFEED in Table 3 trained with only three teachers is close to that of sFEED with Stack-3, which is a reasonable comparison.

4.4 Qualitative Analysis

Reconstruction Loss: Suppose that the reason for the accuracy gains shown in the previous tables is that the student learns the ensemble knowledge that contains information with high complexity. But how can one actually distinguish whether the networks learn complex information or not? Here, we adopt a convolutional autoencoder. The convolutional autoencoders uses 3 convolution and 3 transposed convolution layers, and trained with a $\mathcal{L}_1$ reconstruction loss (for pFEED with WRN and ResNext, we use mse loss to cope with the gradient explosions that occur by the high reconstruction loss). Then the code layer can be interpreted as a latent vector z.

Let us denote the input of autoencoder as x. The increase in the complexity of feature representation is equivalent to the increase in the complexity of x. Since the number of parameters in the autoencoder is fixed, the size of z also should be fixed. Consequently, as the complexity of x increases, $p(x|z)$ decreases, resulting in an increase of the reconstruction loss. Reconstruction errors were used as a criterion for feature selection or PCAs in [2, 5, 16, 17], where they use linear models. In our experiment, we use an arbitrary autoencoder composed of convolution layers with nonlinear activation.

For sFEED and pFEED, we recorded the average training reconstruction losses of the autoencoders normalized by the size of the autoencoder and plotted the curve on Figure 6. In Fig. 6, St1 through St4 on sFEED are the autoencoders trained based on the student net-

Table 4. Test classification error (%) of sFEED on CIFAR-100 dataset. The model's scores on the **Scratch** column are from our implementation. The parameters are counted in Millions.

	Model Type	Scratch	Distillation Type	Algorithm	Stack-2	Stack-3	Stack-4	Stack-5
CIFAR-100	ResNet-56	28.03	Label	BAN	25.85	25.52	**25.30**	25.45
				KD	25.55	25.14	**24.98**	24.98
			Feature	FT	26.33	25.70	25.91	**25.18**
				sFEED	26.02	26.00	25.59	**25.33**
	ResNet-110	27.14	Label	BAN	24.67	**23.81**	23.98	24.06
				KD	24.35	24.01	23.73	**23.66**
			Feature	FT	24.90	24.50	24.34	**24.13**
				sFEED	25.25	**24.33**	24.58	24.40
	WRN28-10	19.00	Label	KD	18.47	**18.43**	18.57	19.05
			Feature	FT	18.23	18.05	18.14	**17.84**
				sFEED	17.68	17.50	17.52	**17.27**
	ResNext29-16x64d	17.31	Label	BAN	16.59	**16.42**	16.43	16.48
				KD	**16.87**	16.90	16.88	-
			Feature	FT	16.80	16.76	**16.47**	16.48
				sFEED	16.80	16.47	16.22	**15.94**

Table 5. Validation classification error (%) of sFEED on Imagenet dataset. The model's scores on the **Scratch*** column are the same as the scores reported on the Pytorch implementation.

Model Type	Scratch*	Stack-2	Stack-3	Stack-4	Stack-5
ResNet-34(Top-1)	26.45	25.60	25.30	25.18	**25.00**
ResNet-34(Top-5)	8.54	8.08	7.86	**7.73**	7.83

works of Scratch (St1) through Stack-4 (St4) in Table 4.3. The autoencoders of pFEED in Figure 6 are trained based on one of the scratch teacher networks and following student network of pFEED and KD in Table 2. As expected, as the knowledge is transferred, the reconstruction loss becomes larger, which indicates that the student network learns more difficult knowledge and thus the classifier accuracy increases. This trend matches the results on the tables. In Figure 6, the the trend of reconstruction losses reciprocally matches the accuracy trend of results in Table 4.3 of sFEED. (Especially, Stack 2 and 3 have similar errors, and likewise, St2 and St3 are similar). The big difference in reconstruction loss between 'Scratch' and 'pFEED' in Figure 6 also corresponds to the high performance increase in ResNet-56 row of Table 2. A better teacher network would learn more complex and detailed features because it has to contain powers to distinguish important but different details from each image to form better decision boundaries. It is worth noting that the curves for KD and pFEED shows opposite aspect, even though they both succeed in enhancing their performance. Here, our assumptions also holds: First, training the student with multiple teacher's feature map will help the student learn detailed features. Second, teacher's labels are abstract information, but will help the student learn key information. Thus, the tendency of KD in Figure 6 is opposite to pFEED. The curves for other type of networks also show consistent aspect, and more examples are handled in the Table 6.

4.5 Implementation Details

CIFAR-100: In the student network training phase, we used $l_1(p = 1)$ loss and the hyper-parameter β in FEED was set to 500 for ResNets and 2,000 for WideResNet and ResNext. We tried to set the training procedure to be the same as those of the original papers. For ResNets, we set the initial learning rate to 0.1 and decayed the learning rate with a rate of 0.1 at 80 and 120 epochs, and training finished at 160 epochs. For WideResNets, we set the initial learning rate to 0.1 and decayed the learning rate with a rate of 0.2 at 60, 120 and 160 epochs, and ended the training at 200 epochs. For ResNexts, we set the initial learning rate to 0.1 and decayed the learning rate with a rate of 0.1 at 150 and 225 epochs, and training finished at 300 epochs. For all the experiments, simple SGD is used as an optimizer, with the momentum of 0.9 and weight decay of 5×10^{-4}, and mini-batch size of 128. The ResNets and WideResNets were trained on single Titan XP and the ResNexts were trained on four 1080 ti GPUs. The same setting was applied for nonlinear transformation layers with 3 convolutions, since they were trained jointly with the student network.

Nonlinear transformation layers: The nonlinear transformation layers for FEED are very simple: just 3×3 convolution filters with leaky-ReLU with the slope of 0.1. The strides are just 1 with padding of 1, so the spatial size is fixed and the number of channels is also fixed for all three convolutions.

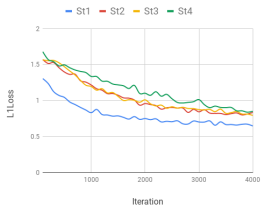
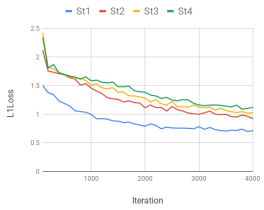
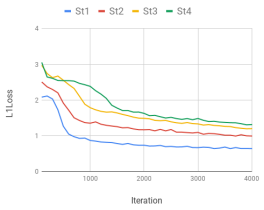
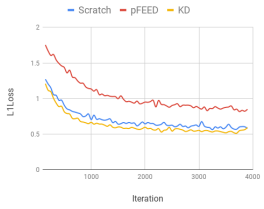
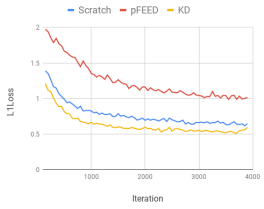
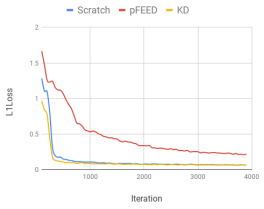
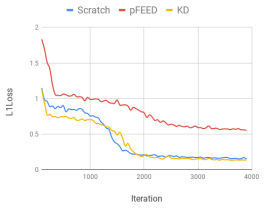
Autoencoder: All the autoencoders Figure 6 are trained for 10 epochs, with a learning rate of 0.1. All autoencoders have 3 convolution layers and 3 transposed convolutional layers for simple implementation.

ImageNet: The hyper-parameter β was set to 1,000. Following the training schedule of the Pytorch framework, the train starts with a learning rate of 0.1 and decays by the factor of 0.1 at 30 and 60 epochs, and finishes at 90 epoch, with a mini-batch size of 256. All other conditions are set to be the same as the setting of CIFAR-100.

The setting of β: For the ease of reproducing, the choice of β is important. Supposing that we use $\mathcal{L}_{FEED}$ with L_1, the loss scale of $\mathcal{L}_{FEED}$ depends on the number of nodes. Empirically, if one of the scales of either $\mathcal{L}_{CE}$ or the other is dominant, the accuracy diminishes compared to the even case. To deal with this, we approximately adjusted the scale of different losses, resulting in different βs in different networks. For AT, since it squeezes the feature map through the channel dimension, the number of elements does not change with $\beta = 500$.

Hyperparameters of KD: We set the Temperature T for softened softmax to 4 and α for scaling as 0.9, following the setting of AT and FT. The explanations for the two hyperparameters can be found in KD [11].

Table 6. Autoencoder reconstruction loss $\mathcal{L}_{rec}$(training) for 4 networks on sFEED and pFEED.

FEED Type	ResNet-56	ResNet-110	WRN28-10	ResNext29-16x64d
sFEED				
pFEED				

5 Discussion

Comparing the results in Section 4.1 and 4.2 shows that the single FEED loss is not greatly helpful, but with ensemble-teacher, pFEED is advantageous. Experiment in Section 4.2 with pFEED shows that allocating nonlinear transformations for each of teacher networks can extract ensemble knowledge from multiple teachers. On the other hand, AT and $\mathcal{L}_1$, which directly mimics the attention map, struggles to learn from multiple teacher networks. The error of pFEED comes closer to the actual model ensemble compared to KD, especially for the models with high capacity. Next, experiments in Section 4.3 compares sFEED with FT, KD, and BAN. Results give lessons to the choice of the algorithm that would be useful depending on the type of networks. The FT which extract key information from the teacher network performs better than sequential FEED for smaller networks and worse for larger networks. The KD and BANs, using labels which is even more abstract, perform better than sequential FEED for smaller networks. However, the result shows that sFEED with nonlinear transformation layers are more useful for networks with a larger capacity. Though not absolute, if one wants to use distillation for model compression with smaller networks, it would be beneficial to seek ones that use abstract information like labels. If one wants to use distillation for high performance where higher performance is needed, distillation methods that can give more detailed information can be useful. The analysis on the reconstruction error, where we utilize convolutional autoencoders, would be helpful to judge whether the network compactly learned its features.

6 Conclusion

In this work, we proposed a couple of new network training algorithms referred to as *FEature-level Ensemble for knowledge Distillation* (FEED). With FEEDs, we can improve the performance of a network by trying to inject ensemble knowledge in the feature-map level to the student network. The first one, parallel FEED trains the student network using multiple teachers simultaneously. The second one, sequential FEED recursively trains the student network and incrementally improves performance. The qualitative analysis with reconstruction loss gives hints about the cause of accuracy gains. The main drawback is the training times needed for multiple teachers, which is an inherent characteristics of any ensemble methods, and pFEED causes bottleneck by feeding inputs to multiple teachers simultaneously. But, it does not affect the inference, which is beneficial without trade-offs in the test time. Devising a more train-efficient method will be our future work, together with an application to other domains other than classification tasks.

Acknowledgement

This work was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF-2017M3C4A7077582).

REFERENCES

[1] Jimmy Ba and Rich Caruana, 'Do deep nets really need to be deep?', in *Advances in neural information processing systems*, pp. 2654–2662, (2014).
[2] Christos Boutsidis, Michael W Mahoney, and Petros Drineas, 'Unsupervised feature selection for principal components analysis', in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 61–69. ACM, (2008).
[3] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil, 'Model compression', **2006**, 535–541, (08 2006).
[4] Thomas G Dietterich, 'Ensemble methods in machine learning', in *International workshop on multiple classifier systems*, pp. 1–15. Springer, (2000).
[5] Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel, 'An efficient greedy method for unsupervised feature selection', in *2011 IEEE 11th International Conference on Data Mining*, pp. 161–170. IEEE, (2011).
[6] Nicholas Frosst and Geoffrey Hinton, 'Distilling a neural network into a soft decision tree', *arXiv preprint arXiv:1711.09784*, (2017).
[7] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar, 'Born again neural networks', *arXiv preprint arXiv:1805.04770*, (2018).
[8] Krzysztof J Geras, Abdel-rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton, 'Blending lstms into cnns', *arXiv preprint arXiv:1511.06433*, (2015).
[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016. http://www.deeplearningbook.org.
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, (2016).
[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, 'Distilling the knowledge in a neural network', *arXiv preprint arXiv:1503.02531*, (2015).

[12] Zehao Huang and Naiyan Wang, 'Like what you like: Knowledge distill via neuron selectivity transfer', *arXiv preprint arXiv:1707.01219*, (2017).
[13] Jangho Kim, SeoungUK Park, and Nojun Kwak, 'Paraphrasing complex network: Network compression via factor transfer', *arXiv preprint arXiv:1802.04977*, (2018).
[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, 'Cifar-100 (canadian institute for advanced research)'.
[15] Xu Lan, Xiatian Zhu, and Shaogang Gong, 'Knowledge distillation by on-the-fly native ensemble', *arXiv preprint arXiv:1806.04606*, (2018).
[16] Jundong Li, Jiliang Tang, and Huan Liu, 'Reconstruction-based unsupervised feature selection: An embedded approach.', in *IJCAI*, pp. 2159–2165, (2017).
[17] Mahdokht Masaeli, Yan Yan, Ying Cui, Glenn Fung, and Jennifer G Dy, 'Convex principal feature selection', in *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 619–628. SIAM, (2010).
[18] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean, 'Efficient neural architecture search via parameter sharing', *arXiv preprint arXiv:1802.03268*, (2018).
[19] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He, 'Data distillation: Towards omni-supervised learning', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4119–4128, (2018).
[20] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, 'Fitnets: Hints for thin deep nets', *arXiv preprint arXiv:1412.6550*, (2014).
[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)*, **115**(3), 211–252, (2015).
[22] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell, 'Policy distillation', *arXiv preprint arXiv:1511.06295*, (2015).
[23] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song, 'Amalgamating knowledge towards comprehensive classification', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3068–3075, (2019).
[24] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo, 'Learning global additive explanations for neural nets using model distillation', *arXiv preprint arXiv:1801.08640*, (2018).
[25] Antti Tarvainen and Harri Valpola, 'Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results', in *Advances in neural information processing systems*, pp. 1195–1204, (2017).
[26] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson, 'Do deep convolutional nets really need to be deep and convolutional?', *arXiv preprint arXiv:1603.05691*, (2016).
[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, 'Aggregated residual transformations for deep neural networks', in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995. IEEE, (2017).
[28] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim, 'A gift from knowledge distillation: Fast optimization, network minimization and transfer learning', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, (2017).
[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, 'How transferable are features in deep neural networks?', in *Advances in neural information processing systems*, pp. 3320–3328, (2014).
[30] Shan You, Chang Xu, Chao Xu, and Dacheng Tao, 'Learning from multiple teacher networks', in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 1285–1294, New York, NY, USA, (2017). ACM.
[31] Sergey Zagoruyko and Nikos Komodakis, 'Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer', *arXiv preprint arXiv:1612.03928*, (2016).
[32] Sergey Zagoruyko and Nikos Komodakis, 'Wide residual networks', *arXiv preprint arXiv:1605.07146*, (2016).
[33] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu, 'Deep mutual learning', *arXiv preprint arXiv:1706.00384*, **6**, (2017).