FEATURE MATTERS: A STAGE-BY-STAGE APPROACH FOR TASK INDEPENDENT KNOWLEDGE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNNs) become deeper and deeper in recent years, making the study of model acceleration imperative. It is a common practice to employ a shallow network, called *student*, to learn from a deep one, which is termed as *teacher*. Prior work made many attempts to transfer different types of knowledge from teacher to student, however, there are two problems remaining unsolved. Firstly, the knowledge used by existing methods is highly dependent on task and dataset, limiting their applications. Secondly, there lacks an effective training scheme for the transfer process, leading to degradation of performance. In this work, we argue that *feature* is the most important knowledge from teacher. It is sufficient for student to just learn good features regardless of the target task. From this discovery, we further present an efficient learning strategy to mimic features stage by stage. Extensive experiments demonstrate the importance of features and show that the proposed approach significantly narrows down the gap between student and teacher, outperforming the state-of-the-art methods.

1 INTRODUCTION

Over the past few years, Convolutional Neural Networks (CNNs) have advanced various tasks in computer vision field, such as image classification (Hu et al., 2018), object detection (Lin et al., 2017b), semantic segmentation (Chen et al., 2018), *etc*. However, along with the architecture growing deeper (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016; Huang et al., 2017), the great success of CNN is at the cost of large computational power, which can not be afforded by most devices in practice. Some lightweight models are presented by recent work (Howard et al., 2017) to reduce the computing cost especially for mobile devices, but the performance drops severely compared with the state-of-the-art methods. Accordingly, it is crucial to balance the trade-off between efficiency and capability of a CNN model.

To tackle this problem, knowledge distillation is introduced in Hinton et al. (2014) for model acceleration. The core idea is to train shallow networks (student) to mimic deep ones (teacher) following two folds. First, teacher employs a very deep model to achieve satisfying performance by excavating information (knowledge) from labeled data. Second, student learns the knowledge from teacher with a shallow model to speed up without losing much accuracy. Accordingly, the main challenges, corresponding to the above two steps respectively, lie in (1) what kind of knowledge should be transferred to student, and (2) how to transfer the knowledge from teacher to student as much as possible.

For the first issue, previous work usually make student to learn from both teacher and original labeled data. There are mainly two problems in doing so. On one hand, it is very sensitive to tasks and datasets. The hyper-parameters, *e.g.* the loss weights to balance these two objective functions, require careful adjustment, or otherwise, it may cause severe performance degradation. On the other hand, the purposes to learn from teacher and to learn from ground-truth are not always consistent with each other. For example, teacher model may eliminate some label errors during the training process. In this case, trying to minimize the loss to mimic teacher as well as the loss from target task may cause confusions to student. Furthermore, prior work has also characterized various types of knowledge from teacher model for student to learn, such as attention map (Zagoruyko

& Komodakis, 2017), information flow (Yim et al., 2017), *etc*. However, all types of knowledge are manually defined, which may not fully conform with the information contained in the teacher network. In other words, teacher is trained independently from these handcraft definitions, but is required to guide the student with such knowledge, which may cause some ambiguities. For the second issue, previous approaches do not solve the problem caused by the gaps between the learning abilities of student and teacher. Intuitively, student model has much less parameters compared to teacher, resulting in lower representation capability. Training it from scratch may always lead to poor performance.

In this paper, we address these weaknesses by proposing a task independent knowledge transfer approach, where student is trained to mimic features from teacher stage by stage. Here, for simplicity, we do not distinguish between feature and feature map. It has two appealing properties.

First, we isolate the knowledge contained in teacher model from the information provided by ground-truth. This goal is achieved with two phases. In the first phase, student learns knowledge by mimicking the output features of teacher, while in the second phase, student is trained with task dependent objection function based on the features from first phase. In this way, student can focus on acquiring information from only one source in each phase, making the transfer process more accurate. Separating these two phases apart also makes our method more generic to various tasks. Besides, we directly treat features as knowledge in this work. Since teacher model just uses features for inference in practice, they are expected to contain the compete information extracted by teacher from training data.

Second, instead of training all parameters of student together, we divide the transfer process into different stages and only train a sub-network at one time. Student network has far more limited representation ability than teacher, resulting in the huge difficulty to mimic the final features directly. To alleviate such obstacle, we let the student to learn from teacher gradually. In other words, both teacher network and student network are separated into sequential parts. Then, in each stage, one part of student will be trained to mimic the output of the corresponding part of teacher with all previous parts fixed. In doing so, the gap between learning powers between student and teacher is narrowed down. As long as each stage is well trained, they will finally collaborate to achieve appealing results.

To summarize, the contributions of this work are as follows:

- We demonstrate the effectiveness of mimicking features directly in task independent knowledge transfer.
- We present a stage-by-stage training strategy to learn features accurately and efficiently.
- We show experimentally that our approach surpasses the state-of-the-art methods on various tasks with higher performance and stronger stability.

2 Related Work

There have been several attempts in the literature to reduce computational cost by model compression. Network pruning was proposed to find a balance between performance and storage capacity by removing redundant structures of the network. Molchanov et al. (2017) and Li et al. (2017) introduced different criteria to evaluate the importance of neurons and filter out the insignificant channels to reduce the network size. Besides, quantization (Courbariaux et al., 2016), which uses fewer bits for each neuron, and low-rank approximation (Zhang et al., 2015), which factorizes a huge matrix with several small matrices, are also widely applied for model acceleration. In this work, we focus on the other model acceleration technique, knowledge transfer, where a shallow student model is trained to gain information from a deep teacher model.

The preliminary view of knowledge transfer was adopted in Ba & Caruana (2014), which makes a shallow network to learn from a deep network by using data labeled by the deep one. Hinton et al. (2014) introduced the concept of Knowledge Distillation (KD), which describes the process of compression as a student learning from a teacher and trying to get similar outputs as teacher. To achieve this goal, the student model is trained with not only the original classification label, but also the class probabilities produced by teacher. Both of these two methods consider the soft label prediction as the key knowledge from teacher model. However, such soft target is not well defined

in other tasks, limiting their applications. In addition, KD is very sensitive to the temperature in the softmax function, and the performance drops severely when the number of classes is very small.

To solve the above problem, Romero et al. (2015) presented FitNets by introducing an intermediate layer as hint for student. Instead of merely learning the soft probabilities, student can also get feature-level knowledge by mimicking the output feature map of the hint layer. Furthermore, FitNets is formulated as a two-stage training, where the student network can get a better initialization in the first stage. Different from FitNets, stages in our framework are trained separately, *i.e.* only parameters of a sub-network will be updated in one stage while others will be fixed. In this way, we can prevent the transferred knowledge vanishing and speed up the mimicking process as well.

Inspired by FitNets, various types of knowledge have been proposed to assist student to get better results. Zagoruyko & Komodakis (2017) attempted to transfer spatial attention map, which is defined as the statistics of feature maps across the channel dimension. Huang & Wang (2017) proposed to learn feature map through Maximum Mean Discrepancy (MMD), which can be regarded as a sample-based metric to measure the distance between two probability distributions. Yim et al. (2017) used Flow of Solution Procedure (FSP) to describe the information flow of a CNN model, which computes the Gram matrix of features from two layers. Similarly, Lee et al. (2018) applied Singular Value Decomposition (SVD) to derive the correlation between two features and forced student to learn such distilled information.

However, all the above methods, no matter single-stage or multiple-stages, learning from probabilities or features, using MMD or SVD as the description of the relationship between features, do not treat knowledge transfer as a task independent problem. They combine their proposed methods with the original task loss together, which leads to three main problems: (1) Poor stability, since they are sensitive to the hyper-parameters, such as the weights to balance different loss terms; (2) Huge limitations, as some specific designs hinder their migration to other tasks; (3) Low performance, because the knowledge from teacher and the information from original data are not always aligned with each other. Moreover, existing methods lack an effective solution to close the enormous gap of learning abilities between student and teacher.

On the contrary, we propose to transfer the knowledge from network level instead of task level, leading to a much more general solution which can be easily integrated into different scenarios. Besides, we introduce a stage-by-stage training strategy to reduce the representation capability difference between shallow and deep models in each stage. Similar to our method, a recent work (Wang et al., 2018) mentioned a progressive block-wise training scheme. However, in their work, when training a particular block, the latter blocks of student should be replaced by those of teacher temporarily to make sure the original task loss can be applied. Accordingly, all blocks should be computed at each stage, which is very time consuming. Besides, the block conversion between teacher and student may also cause some confusions to student. By contrast, our approach is much more efficient and accurate.

There is also some work introducing reinforcement learning (Ashok et al., 2018) and adversarial network (Belagiannis et al., 2018) to model compression problem. However, training these networks is still not trivial for now, *e.g.* how to design the networks, how to update the parameters, how to formulate the reward in reinforcement learning, or how to establish the competition in adversarial nets. Even so, we still would like to appreciate their contributions.

3 Proposed Method

Figure 1 illustrates our method where student mimics features from teacher stage by stage. Before going into details, we will introduce two observations to explain our motivation intuitively.

Observation 1. In general, most deep CNN models can be regarded as two parts, which are feature extraction and feature application. More specifically, given an image, the first part will interpret it as a high dimensional feature, and then the second part will take such representation as input and output a prediction for some specific tasks. Taking classification task as an example, after characterizing an input sample as a feature vector, the classification model engages a fully-connected (fc) layer followed by a softmax function to predict the probabilities of this sample for different classes. Similarly, this is also applicable to detection model, such as RetinaNet (Lin et al., 2017b), which uses a backbone structure to extract feature map as well as subnets to predict bounding boxes.

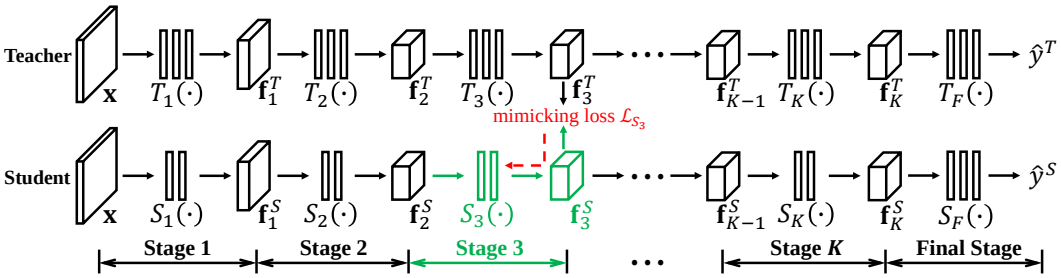

Figure 1: Overview of the proposed stage-by-stage knowledge transfer approach. Both teacher and student models are divided into K stages, where the number of rectangular bars indicates the depth of the sub-network. Student is trained to mimic the output feature of teacher at each stage and the output feature of the K-th stage can be finally applied to different tasks. Note that we do not transfer the knowledge in the final stage, since teacher and student share the same network structure in this stage. When training a particular stage, as shown in green color, all parameters in previous well-trained stages are fixed. Red dashed arrow shows the backward propagation. Better viewed in color.

For simplicity, both feature and feature map are called features in this work, no matter it is a 1D vector, 2D matrix or a 3D tensor.

Observation 2. In the two parts mentioned above, the only difference between student and teacher is the ability to extract features from images, because they share the same structure in how to use these features. For example, in classification task, the fully-connected layer of student has the same number of parameters as that of teacher, resulting in equal capability in converting features to probabilities. In other words, if the student could produce identical feature as teacher, it should be able to achieve as promising performance as teacher. In this way, when transferring knowledge from teacher to student, we should focus on the feature extraction part instead of the entire network. Another advantage in doing so is that student gains all the information from teacher and does not need to learn from the labelled data, making it a generic solution which is independent from tasks.

3.1 Feature Transfer

As discussed before, given an input image $\mathbf{x}$, a CNN model $M(\cdot)$ makes inference through two steps. The first step is called feature extractor $M_E(\cdot)$, which encodes the image to feature space with $\mathbf{f} = M_E(\mathbf{x})$. Then, the second step, which is termed as final stage $M_F(\cdot)$ in this work, aims at predicting the ground-truth y based on the feature from first step by producing $\hat{y} = M_F(\mathbf{f})$. Therefore, we have

$$M = M_F \circ M_E, \tag{1}$$

where $\circ$ indicates the function composition.

Then the model can be trained with objective function

$$\min_{\Theta_M} \mathcal{L}_M = \phi(y, M(\mathbf{x})), \tag{2}$$

where Θ_M, consisting of Θ_{M_E} and Θ_{M_F}, is the trainable parameters of the entire model. ϕ is the task-related energy function, such as softmax cross-entropy loss in classification task, bounding box regression loss in detection task, *etc*.

To transfer knowledge, we employ a shallow student model $S(\cdot)$ to learn from a deep teacher model $T(\cdot)$, which has been well trained with Eq.(2). However, unlike existing methods where student is trained to acquire information from teacher and minimize the original task loss in Eq.(2) simultaneously, we divide the training process into two phases, feature learning and task adaption, corresponding to the above two steps respectively.

In the first phase, student tries to mimic the output feature $\mathbf{f}^T = T_E(\mathbf{x})$ from teacher with

$$\min_{\Theta_{S_E}} \mathcal{L}_{S_E} = d(\mathbf{f}^T, S_E(\mathbf{x})), \tag{3}$$

where $d(\cdot, \cdot)$ is a metric to measure the distance between two features. Actually, there are many choices of $d(\cdot, \cdot)$, such as l_1 distance, l_2 distance, KL divergence, JS divergence, *etc*. But the attempts on different losses are out of the scope of this work, and we just use the l_2 distance in our experiments.

In the phase of task adaption, student learns to apply features for different tasks by minimizing Eq.(2). However, different from the training of teacher, we fix the parameters Θ_{S_E} to prevent the transferred knowledge vanishing, and only Θ_{S_F} is updated. As mentioned before, $S_F(\cdot)$ and $T_F(\cdot)$ have the same learning ability, thus there is no need to use teacher to guide student in this step any more. In other words, we transfer knowledge from teacher to student in the first step, and train student how to apply the features for some certain tasks in the second step.

3.2 Stage-by-Stage Training Scheme

From discussion above, it is crucial for our method that student can learn similar features as teacher. However, considering the wide difference between the representation capabilities of these two models, the goal is not that easy to achieve through simple end-to-end learning. To tackle this problem, we break them down into multiple stages, as shown in Figure 1. It is much easier for student to mimic the output of teacher in each stage. Taking teacher model as an instance, we have

$$\mathbf{f}_0^T = \mathbf{x}, \tag{4}$$

$$\mathbf{f}_i^T = T_i(\mathbf{f}_{i-1}^T) \quad i = 1, 2, \cdots, K, \tag{5}$$

$$T_E = T_1 \circ T_2 \circ \cdots \circ T_K, \tag{6}$$

where K is the total number of stages, excluding the final task adaption stage. $\mathbf{f}_i^T$ is the output feature of the i-th stage in teacher network, while $\mathbf{f}_0^T$ is the initial feature, *i.e.* the image fed into the network. $T_i(\cdot)$ is the sub-network of teacher model in each stage, and the feature encoder $T_E(\cdot)$, which is defined in Section 3.1, can be considered as a composition of a set of sub-networks. Similarly, student feature encoder is also divided into K stages.

Under such separation, we propose to train the student model stage by stage with

$$\min_{\Theta_{S_i}} \mathcal{L}_{S_i} = d(\mathbf{f}_i^T, S_i(\mathbf{f}_{i-1}^S)), \tag{7}$$

where $d(\cdot, \cdot)$ is same with that in Eq.(3). Taking after the training strategy for final stage mentioned in Section 3.1, we fix all parameters in previous stages when training a new stage, to prevent the transferred knowledge vanishing. That is to say, only a subset of parameters will be updated at each stage in this scheme. Therefore, even though we have more stages than the conventional end-to-end training, the computational cost of training process almost remains the same, making our method not only accurate but also efficient.

Note that how to break down teacher and student models is worth studying in this framework, but is also out of the scope of this work. We just use the down-sampling layers of the network (or up-sampling layers in some particular structure) as the breakpoints.

4 Experiments

To evaluate the performance of our proposed method, we carry out various experiments on different datasets and different tasks. Section 4.1 briefly introduces the basic settings used in our experiments. Section 4.2 conducts a series of comparative experiments to verify the importance of learning good features in CNN model. Section 4.3 demonstrates the efficiency of the novel stage-by-stage feature transfer scheme. Section 4.4 explores our approach on classification and detection tasks and compares with other state-of-the-art knowledge transfer methods.

4.1 Experimental Settings

We evaluate our feature transfer method on several standard datasets, including CIFAR-100 (Krizhevsky & Hinton, 2009), ImageNet (Krizhevsky et al., 2012) and COCO (Lin et al., 2014). Among them, CIFAR-100 and ImageNet are used for classification task, while COCO is for

Table 1: Experiments on features with ImageNet dataset.

Experiment	Model	top1	top5
student	ResNet-18	69.572	89.244
student (fixed feature)	ResNet-18	69.952	89.242
teacher	ResNet-34	73.554	91.456
teacher (fixed feature)	ResNet-34	73.532	91.439
teacher (fixed first block)	ResNet-34	73.547	91.458
KD (Hinton et al., 2014)	ResNet-18	70.759	89.806
KD (fixed feature)	ResNet-18	70.752	89.837

detection task. For classification task, a pre-trained ResNet-34 (He et al., 2016) is employed as teacher model and a shallower network, ResNet-18, is adopted as student model. We also present experimental results on detection task with RetinaNet following settings in Lin et al. (2017b). ResNet-101 (Lin et al., 2017a) and ResNet-50 are applied as teacher student models respectively.

To further validate the effectiveness of our method, a set of comparison experiments have been done with state-of-the-art knowledge transfer approaches, including KD (Hinton et al., 2014), FitNets (Romero et al., 2015), AT (Zagoruyko & Komodakis, 2017) and NST (Huang & Wang, 2017). We set $T = 4$ and $\lambda = 16$ in KD following the settings in (Hinton et al., 2014). For FitNets, AT and NST, the transfer loss is calculated with 4 outputs of each residual block in classification task, and with 4 feature maps fed into Feature Pyramid Network (FPN) in detection task.

4.2 Preliminary Experiments on Features

In this section, we set up a series of experiments to show the importance of features and the sufficiency of training a model with fixed well-learned features. We take classification task on ImageNet as an example. As discussed in Observation 1 of Section 3, the whole network can be divided into two parts, which are feature extraction part before fully-connected (fc) layer and the softmax-activated fc layer as the classifier part.

Given a model, student or teacher, we firstly train the entire network with task-related objective function. Then we randomly re-initialize the fc layer, fix the feature extraction part, and merely train fc layer with same loss. As shown in the first four lines in Table 1, as long as the network can learn good enough features for follow-up task, no matter shallow (student) or deep (teacher), it will always achieve nearly the same results as end-to-end training. We also experiment on relaxing the fixed part to the first residual block and start training from the second block. It turns out that there is no real distinction in where to break up the model. This is also the basis for the feasibility of our proposed stage-by-stage feature transfer scheme.

To further prove whether this conclusion is applicable for knowledge transfer, we conduct the same experiment on the classic KD method (Hinton et al., 2014). As shown in the last two lines in Table 1, after the transfer process, even the feature is fixed, KD can still achieve competitive results. It suggests that learning distilled knowledge just help student to get a better feature representation than learning from scratch. Under this conclusion, if student can learn an identical feature from teacher, then the left part of the network should have the ability to get to the same performance as teacher, even without the supervision from teacher. Therefore, we argue that it is much more efficient to make student directly mimic the output features of teacher.

4.3 Stage-by-Stage Learning

To show the effectiveness of training with multiple stages, we apply experiments on CIFAR-100 by using different number of stages. ResNet-18 and ResNet-34 are adopted as student model and teacher model respectively, and the network is divided into 4 blocks using down-sampling layers as breakpoints. We train four independent student models with $1 \sim 4$ stages respectively, in addition to a final stage to fine-tune the classifier layer. Here, one stage indicates an end-to-end feature learning, two-stages strategy trains the first block independently from other parts, and so on and so force.

Table 2: Experiments on number of training stages with CIFAR-100 dataset.

Method	Model	top1	top5
student	ResNet-18	68.062	89.598
teacher	ResNet-34	73.045	90.545
1 stage	ResNet-18	70.371	89.100
2 stages	ResNet-18	71.223	90.000
3 stages	ResNet-18	72.321	90.795
4 stages	ResNet-18	**72.768**	**91.396**

When training a new stage, all previous stages are fixed and only the parameters in the sub-network of current stage will be updated. Besides the number of stages, we keep all other hyper-parameters same.

Table 2 summarizes the results. It is not hard to tell that all these four models achieve higher accuracy than the baseline of student model, which is the first line in Table 2. Furthermore, along with the number of training stages increasing, the performance gets better and better. Training with 4 stages, which is the final version of our method, achieves 4.7% improvement in top1 accuracy compared with the baseline model, and 2.4% compared with the single stage model. This benefits from the narrowed gap between student and teacher in each stage. In this way, it is much easier to transfer knowledge between these two models. Consequently, for teachers with more layers or with more complicated structures, we can add more stages to improve the performance of student by following this strategy.

4.4 Knowledge Transfer on Different Tasks

In this section, we compare our approach with other state-of-the-art knowledge transfer methods on different tasks, including image classification and objective detection. Note that we use exactly the same training settings, *e.g.* learning rate and optimizer type, when performing experiments on different tasks (or datasets), to verify that our method can be generally applied for various problems.

4.4.1 Image Classification

For classification task, we apply CIFAR-100 and ImageNet as validation datasets to evaluate how our framework can fit with various numbers of classes. With the same setting in Section 4.3, we employ ResNet-18 and ResNet-34 as student and teacher respectively, and use 4 stages in the training process.

Evaluation on CIFAR-100. We firstly start with CIFAR-100 dataset which consists of 50K training images and 10K testing images from 100 classes. Table 3 shows the comparison results. Our proposed stage-by-stage feature transfer scheme surpasses other work in both top1 accuracy and top5 accuracy. We even achieve similar performance as teacher model, suggesting that student in our approach collects as much information from teacher as possible.

Evaluation on ImageNet. We also conduct large-scale experiments on ImageNet dataset, which includes over 1M training images and 50K testing images collected from 1,000 categories. As shown in Table 4, our method also achieves the best results. In addition, by comparing Table 3 and Table 4, we can tell that other methods perform inconsistently on different datasets. For example, although learning attention map in AT method (Zagoruyko & Komodakis, 2017) works well on ImageNet, the accuracy drops a lot when it is applied to CIFAR-100. Similarly, FitNets (Romero et al., 2015) achieves good transfer on CIFAR-100, but fails on ImageNet. That is because the design of these methods is highly sensitive to the hyper-parameters, making them share discrepancies on different datasets. Compared to them, our method, which focuses on transferring knowledge from network to network independently from tasks, shows much stronger stability.

Table 3: Comparison results of image classification task on CIFAR-100 dataset.

Method	Model	top1	top5
student	ResNet-18	68.062	89.598
teacher	ResNet-34	73.045	90.545
KD (Hinton et al., 2014)	ResNet-18	72.393	91.062
FitNets (Romero et al., 2015)	ResNet-18	71.662	90.277
AT (Zagoruyko & Komodakis, 2017)	ResNet-18	70.741	90.036
NST (Huang & Wang, 2017)	ResNet-18	70.482	89.241
ours	ResNet-18	**72.786**	**91.396**

Table 4: Comparison results of image classification task on ImageNet dataset.

Method	Model	top1	top5
student	ResNet-18	69.572	89.244
teacher	ResNet-34	73.554	91.456
KD (Hinton et al., 2014)	ResNet-18	70.759	89.806
FitNets (Romero et al., 2015)	ResNet-18	70.662	89.232
AT (Zagoruyko & Komodakis, 2017)	ResNet-18	71.237	90.146
NST (Huang & Wang, 2017)	ResNet-18	70.762	89.586
ours	ResNet-18	**71.361**	**90.496**

Table 5: Comparison results of object detection task on COCO benchmark.

Method	Backbone	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
student	ResNet-50	35.5	55.6	38.5	17.8	39.1	47.7
teacher	ResNet-101	38.7	58.7	41.9	21.5	42.3	49.9
FitNets (Romero et al., 2015)	ResNet-50	36.2	55.3	39.1	19.0	40.6	48.5
AT (Zagoruyko & Komodakis, 2017)	ResNet-50	35.9	54.7	38.2	19.3	39.6	46.6
NST (Huang & Wang, 2017)	ResNet-50	35.7	54.3	38.0	18.6	39.7	46.8
ours	ResNet-50	**36.5**	**55.6**	**39.4**	**19.5**	**40.7**	**48.8**

4.4.2 Object Detection

To further prove that our method is task independent, we also conduct experiments on detection tasks. As described in Section 4.1, we use RetinaNet (Lin et al., 2017b) as the detection framework, and employ ResNet-50 and ResNet-101 as the backbones of student and teacher respectively. The feature mimicking part is trained in the same way as classification task in Section 4.4.1. We do not compare with KD method in this task, since soft target can not be directly applied to bounding box regression in objection detection.

From the results in Table 5, we can see that our method outperforms the baseline model with 1.0% mean Average Precision (mAP) and also surpasses all the other methods, demonstrating the effectiveness of our approach.

Furthermore, by comparing the performance of each method on different tasks, we find that previous work is not as stable as ours, consistent with the conclusion in Section 4.4.1, where a same approach shows different performance on different datasets. This situation is caused by combining the knowledge transfer loss and the task loss together. Although teacher is trained with task loss as well, it is able to filter out some impractical information from original data. Such distillation requires strong learning ability, which the student model does not possess. This is also the reason why we require a teacher model to guide the student. However, mixing these two objective functions together will cause the confusion of the student, since it has no idea about which information, filtered by teacher or extracted from raw data, is more accurate. In other words, mimicking loss and target loss are not always consistent with each other. On the contrary, our method divides the whole process

into feature learning stage and final stage, which are trained with these two losses independently, leading to stronger stability.

5 CONCLUSION

This work presents a stage-by-stage knowledge transfer approach by training student to mimic the output features of teacher network gradually. Compared to prior work, our method pays more attention to the information contained in the model, regardless of what task the model is applied for, making it a generic solution for model acceleration. The progressive training strategy helps reduce the learning difficulties of student in each stage, and all stages cooperate together for a better result. Extensive experimental results suggest that our scheme can significantly improve the performance of student model on various tasks with strong stability.

REFERENCES

Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M. Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. In *ICLR*, 2018.

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

Vasileios Belagiannis, Azade Farshad, and Fabio Galasso. Adversarial network compression. *arXiv preprint arXiv:1803.10750*, 2018.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Workshop*, 2014.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *ECCV*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Tsung Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017a.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017b.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *IJCAI*, 2018.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *CVPR*, 2015.