Neurocomputing 433 (2021) 154–161

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Residual error based knowledge distillation

Mengya Gao, Yujun Wang, Liang Wan *

Tianjin University, Hong Kong University, Tianjin University, China

ARTICLE INFO

Article history:
Received 17 November 2019
Revised 8 June 2020
Accepted 12 October 2020
Available online 28 November 2020
Communicated by Ivor Tsang

Keywords:
Model compression
Knowledge distillation
Residual learning

ABSTRACT

Knowledge distillation (KD) is one of the most popular ways for model compression. The key idea is to transfer the knowledge from a deep teacher model (*T*) to a shallower student (*S*). However, existing methods suffer from performance degradation due to the substantial gap between the learning capacities of *S* and *T*. To remedy this problem, this paper proposes Residual error based Knowledge Distillation (RKD), which further distills the knowledge by introducing an *assistant* model(*A*). Specifically, *S* is trained to mimic the feature maps of *T*, and *A* aids this process by learning the *residual error* between them. In this way, *S* and *A* complement with each other to get better knowledge from *T*. Furthermore, we devise an effective method to derive *S* and *A* from a given model without increasing the total computational cost. Extensive experiments show that our approach achieves appealing results on popular classification datasets, CIFAR-100 and ImageNet, surpassing state-of-the-art methods and keep strong robustness to adversarial samples.

1. Introduction

Recent years have witnessed the great success of convolutional neural networks (CNNs) on a variety of computer vision tasks [1–4]. However, the advantage of CNN is accompanied with very deep model structure [5–8], which requires extensive computing and memory costs, hindering it from being applied to many real-world applications. To this end, it becomes crucial to explore a way of reducing the model size but barely sacrificing the performance. Knowledge distillation (KD) [9,10] is a popular solution among existing model compression methods, including network pruning [11,12], parameter quantization [13], and low-rank approximation [14]. In general, KD starts with training a large model, called teacher (*T*), to achieve appealing performance, and then employs a smaller low-capacity one, termed as student (*S*), to learn knowledge from *T*. In this way, *S* is supposed to produce similar prediction as *T* but with faster speed and less memory consumption. Accordingly, it is critical for KD to make *S* gain as much information from *T* as possible.

To achieve more effective knowledge transfer from *T* to *S*, many attempts have been made [15–18]. However, this problem is far from being solved. As shown in Fig. 1, even though KD (orange line) helps improve the performance of student (blue line), there still exists a significant gap compared to the teacher (dashed grey line). This is mainly caused by two reasons. First, *S* has a much weaker representation ability than *T*. As can be observed in Fig. 1, although the performance of *S* can gradually approximate that of *T* by increasing its capacity, the inherent discrepancy between the model size of *S* and *T* still prevents the student to fully acquire knowledge from a well-trained teacher. Second, there lacks of an effective strategy to distill the knowledge inside *T*. Previous KD methods typically use one teacher to supervise one student, resulting in a one-to-one learning. That is to say, the distillation process happens only once when *S* is optimized to mimic *T*, as shown in Fig. 2(a). Recall that there is a big gap between the capacity of *S* and *T*. Therefore, such one-time transfer scheme will lead to information lost to some extent.

To ease the process of knowledge transfer, we present a novel method called Residual error based Knowledge Distillation (RKD), which introduces an assistant (*A*) into the KD process to help *S* in getting better knowledge from *T*. The concept diagram of RKD is depicted in Fig. 2(b). More concretely, feature maps are treated as knowledge in this work, and *S* is trained with the goal of producing identical feature maps as in *T*. Instead of hoping the student to achieve the ideal mimicking, we employ an additional model, *A*, to assist *S* by learning the residual error between the feature maps of *S* and *T*. Then, the output feature map of *S*, summed up with the residual error learned by *A*, will be used in practice. With such transferring scheme, the final feature map is more indistinguishable from *T*, hence narrowing down the performance gap between *S* and *T*, as shown in Fig. 1. The contributions of this work are summarized as follows:

* Corresponding author.

https://doi.org/10.1016/j.neucom.2020.10.113
0925-2312/

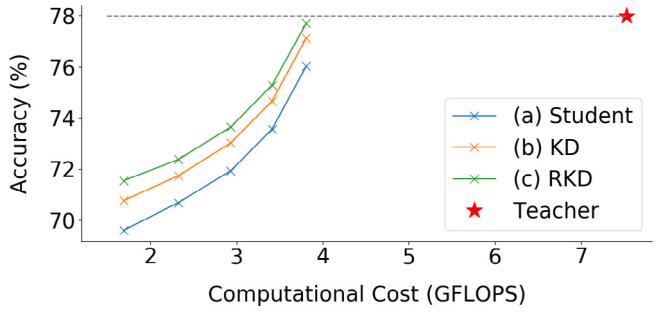


Fig. 1. Relationships between model capacity and performance on ImageNet across different methods, including (a) training a student model without supervision from teacher, (b) KD[10], and (c) our proposed RKD. Here, model capacity is measured by the computational cost in unit of Giga floating-point operations (GFLOPS) and we use different model with student to achieve these. Comparison of capacity with other knowledge distillation method can be seen in 4 Red star on the top-right corners indicates the teacher model. Better viewed in color.

- We propose a novel model compression method, RKD, by extending the conventional KD with an assistant model. Different from previous methods that perform a single round of knowledge distillation, RKD distills the knowledge for a second time by training *A* to learn the residual error between *S* and *T*. In this way, *A* is able to capture the information which *S* fails to learn from *T*, leading to a more complete knowledge transfer. Furthermore, the idea of learning residual error can also help improve the performance of some existing KD methods.
- We present a way to separate an existing model into two sub-models to serve as *S* and *A* respectively. In doing so, the size of the entire model, *S* in addition with *A*, will not increase, resulting in a fair comparison with alternative methods.
- We experimentally demonstrate that it is sufficient for *A* to mimic the residual error perfectly with a lightweight model (around 1/10 model size of *S*). In contrast with the negligible incremental computational cost, *S* appreciates significant performance gain benefiting from *A*.

2. Related work

2.1. Knowledge distillation

Knowledge distillation (KD) is firstly introduced in [9] and then brought back to popularity by [10]. The rational behind is to use a student model (*S*) to learn from a teacher model (*T*) without sacrificing much accuracy. Challenges lie in two aspects, which are (1) how to extract the knowledge from *T*, and (2) how to transfer the knowledge to *S*.

2.1.1. Knowledge type

Existing methods have designed various types of knowledge to improve KD. Ba and Caruana [19] treated the hard label predicted by *T* as the underlying knowledge, with the assumption that the well-trained *T* has already eliminated some label errors contained in the ground-truth data. Hinton et al. [10] argued that the soft label produced by *T*, like the classification probabilities, can provide richer information. Some work extracted the knowledge from *T* by processing the hidden feature map. Zagoruyko and Komodakis [18] averaged the feature map across channel dimension to obtain spatial attention map, Yim et al. [17] defined inter-layer flow by computing the inner product of two feature maps, and Lee et al. [20] improved this idea with singular value decomposition (SVD). *A* very recent work [21] demonstrated the effectiveness of mimicking feature map directly in KD task. Similarly, this work also applies feature map as the knowledge, since the residual error between feature maps is well defined. Differently, however, our proposed RKD can also work together with other types of knowledge, such as attention map, as long as there is a way to compute the residual error.

2.1.2. Transferring strategy

Besides knowledge type, transferring strategy is another widely studied direction in KD. Romero et al. [15] presented FitNets which selects a hidden layer from *T* and *S* respectively to be hint layer and guided layer. Through pre-training the guided layer with the hint

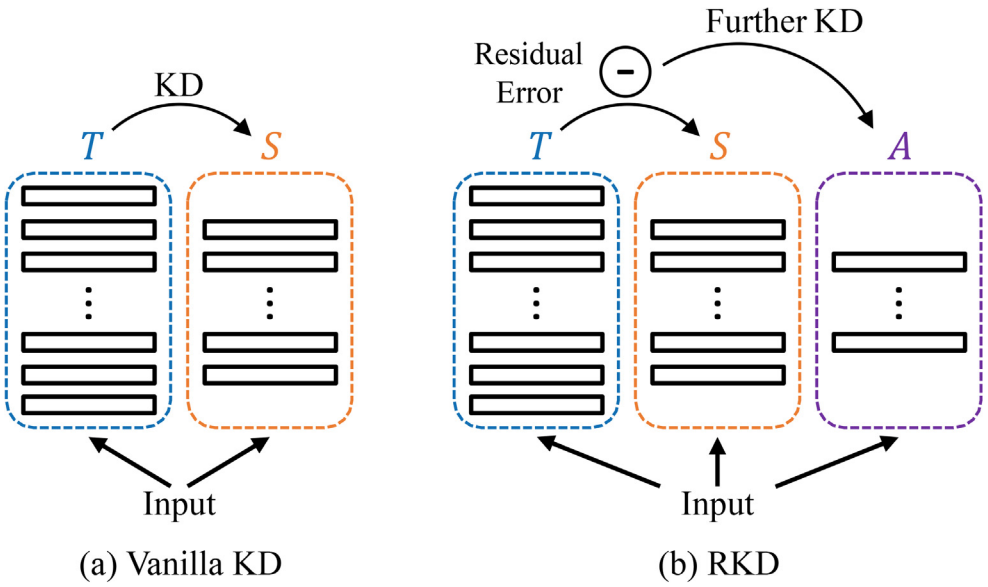


Fig. 2. Compared to the the vanilla knowledge distillation (KD) method in (a) [10], where the student (*S*) is trained to gain information from teacher (*T*) all at once, (b) illustrates the concept diagram of the proposed RKD, which employs an assistant (*A*) to further distill the knowledge inside *T*. *A* helps to achieve better knowledge transfer with the purpose of learning the residual error between the feature maps of *S* and *T*.

layer as supervision, S is able to get a better initialization than trained from scratch. Net2net [22] also explored the way to initialize the parameters of S by proposing function-preserving transformation, which makes it possible to directly reuse an already trained model. In addition to initialization, there are some methods combining KD with other techniques to transfer knowledge from T to S more efficiently. Belagiannis et al. [23] involved adversarial learning into KD by employing a discriminator to tell whether the outputs of S and T are close enough, Ashok et al. [24] exploited reinforcement learning to find out the best network structure of S under the guidance of T, and Wang et al. [25], Gao et al. [21] referred to the idea of progressive learning to make knowledge transferred step by step. Nevertheless, all of the above methods use a single model, S, to learn from T, and the knowledge is distilled only once. Considering the difference between the learning capacities of S and T, there is no guarantee that S can obtain enough information to reproduce the performance of T via such one-time transfer. On the contrary, we propose to make further knowledge distillation with an additional model, called assistant (A). A is able to distill the knowledge for a second time and hence help transfer the information from T to S more sufficiently.

In this literature, there are also some approaches that realize KD without using the T-S pair. Furlanello et al. [26], Lan et al. [27] blurred the boundaries between T and S by introducing the concept of self-learning, in which case the knowledge is transferred within the same network. Some online distillation methods [28–30] trained a set of S models simultaneously and made them learn from each other in peer-teaching manner. However, S and A in this work complement with each other in a different way. A recent work, called TAKD[31], proposed to improve KD method by introducing intermediate teacher assistant model (TA). Specially, TAKD transfers knowledege from T to S with two steps (i.e., first from T to TA, then from T A to S), which are independent from each other. Differently, however, RKD employs the assistant model A to learn the reidual error between S and T, such that A can acquire information from both of them.

2.2. Residual learning

Residual learning is an effective learning scheme, which is firstly adopted to CNN by [6]. After that, this strategy has been applied to various tasks, such as visual question–answering [32], stereo matching [33], image denoising [34], and image super resolution [35,36]. In general, the core thought is based on the hypothesis that learning the residual is easier than optimizing the target function directly without any reference. This is consistent with the coarse-to-fine idea, where a problem is solved with a coarser (identity branch) and a finer (residual branch). This work introduces this principle into KD task. Unlike prior work that only optimized S (coarser) to learn from T, our proposed RKD employs A (finer) to learn the residual error between S and T. In this way, A is able to refine the feature map of S, such that the knowledge of T is distilled more completely. To the best of our knowledge, this is the first work that utilizes residual learning on model compression.

3. Residual error based knowledge distillation

We formulate RKD by employing two models, *i.e.* S and A, to get knowledge from T in a complementary manner, as shown in Fig. 3 (M.1). To transfer knowledge more accurately and make RKD easier to train, we also present two variants of this framework, including a progressive learning scheme (M.2), and an end-to-end training scheme that integrates A into S (M.3). In addition, we explore an efficient way to derive the network structures of S and A from a pre-designed model, such that the total computational cost will not increase after introducing A. More details will be discussed in the following sections.

3.1. Knowledge distillation

Knowledge distillation (KD) method typically employs a student $S(\cdot)$ to learn from a well-trained teacher model $T(\cdot)$, aiming at reproducing the predictive capability of T. In other words, given an image-label pair $(\mathbf{x}, y)$, T will make a prediction $\hat{y}^T = T(\mathbf{x})$, and S is trained with the purpose of outputting similar result as $\hat{y}^T$. Here, the prediction made by S is denoted as $\hat{y}^S = S(\mathbf{x})$.

To achieve this goal, KD targets at exploring a way to extract the information contained in a CNN model and then push the information of S to be as close to that of T as possible. Accordingly, KD can be formulated as

$$\min_{\Theta_S} \mathscr{L}_S = d(\psi(T(\cdot), \Theta_T), \psi(S(\cdot), \Theta_S)), \tag{1}$$

where Θ_T and Θ_S are the trainable parameters of T and S respectively. $\psi(\cdot,\cdot)$ is the function that helps define the knowledge of a particular model, and $d(\cdot,\cdot)$ is the metric to measure the distance between the knowledge of two models.

Note that, only Θ_S in Eq. (1) is updated, since T is assumed to have already been optimized with ground-truth data. For classification tasks, cross-entropy loss is used as the objective function

$$\min_{\Theta_T} \mathscr{L}_T = -\sum_{i=1}^{N} y_i \log \hat{y}_i^T, \tag{2}$$

where N is the number of categories, y is an N-dimensional one-hot vector indicating the ground-truth label, while $\hat{y}^T$ is the soft probabilities predicted by T.

3.2. Residual learning with A (M.1)

In this work, we treat feature maps of a CNN model as the underlying knowledge. Generally, a model can be divided into a set of blocks, and the output of each block is considered as a hidden feature map. Taking teacher in Fig. 3 as an example, T consists of K blocks, $\{T_i(\cdot)\}_{i=1}^K$, and possesses K hidden feature maps, $\{\mathbf{f}_i^T\}_{i=1}^K$, correspondingly. Besides the K blocks shown in Fig. 3, T also employs a classifier, *i.e.* a fully-connected layer activated by softmax function, to convert the final feature map $\mathbf{f}_K^T$ to soft label prediction $\hat{y}^T$. However, inspired by [21], classifiers of T and S share the same structure as well as equal learning capacity, and hence they are excluded from the knowledge distillation process. We therefore only focus on the knowledge contained in the feature extraction part. Similar as T, S is also divided into K blocks. Accordingly, to transfer knowledge from T to S, Eq. (1) can be simplified as

$$\min_{\Theta_S} \mathscr{L}_S = ||\mathbf{f}_K^T - \mathbf{f}_K^S||_2^2, \tag{3}$$

where $||\cdot||_2$ denotes the l_2 distance between two tensors. In other words, S attempts to produce identical feature map as T such that they can achieve comparable performance. However, considering the substantial gap between the representation capacities of S and T, it is not trivial to fit sufficiently well the underlying knowledge captured by the features maps of T with just S alone. To solve this problem, we employ another model, called assistant (A), to aid S in this mimicking process, which is shown in Fig. 3 (M.1).

Specifically, A, also with K blocks, $\{A_i(\cdot)\}_{i=1}^K$, takes the image $\mathbf{x}$ as input and is optimized to learn the residual error between $\mathbf{f}_K^S$ and $\mathbf{f}_K^T$ with loss function

$$\min_{\Theta_A} \mathscr{L}_A = ||(\mathbf{f}_K^T - \mathbf{f}_K^S) - \mathbf{f}_K^A||_2^2. \tag{4}$$

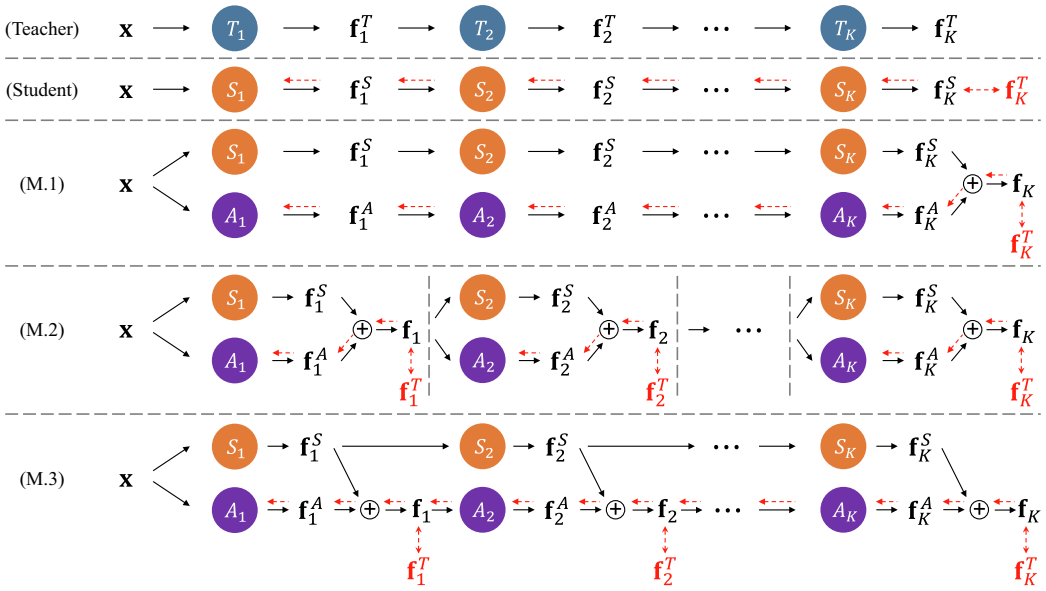


Fig. 3. The first two rows illustrate how the knowledge is transferred from teacher (T) to student (S), both of which are divided into K blocks. After S is well trained, an assistant (A) is employed to learn the residual error between the feature maps of S and T in (M.1). Similar as T and S, A is also divided into K blocks. (M.2) applies the methodology of (M.1) onto each block and then trains S and A progressively. For each step of training, residual error inside one block will be calculated. After one block is trained well, the next block will be trained and so on. (M.3) improves (M.2) by integrating A into S such that A can be trained from end to end. Different from (M.2), the residual error in all the block will be considered in each training step.

Then, after A and S reaching their optima respectively, the feature map summed up with the residual error, $\mathbf{f}_K = \mathbf{f}_K^S + \mathbf{f}_K^A$, will be finally used for inference.

By introducing A, the knowledge is distilled via two phases, where S is firstly optimized with Eq. (3) to mimic the hidden feature map of T, and then the parameters of A are updated with Eq. (4) at the residual learning stage to refine the feature map learned by S. Note that only one model, either S or A, is trained at each phase. Although trained separately, A share the same goal with S, which is to approximate $\mathbf{f}_K^T$ with $\mathbf{f}_K$. Consequently, A complements with S to improve the performance by picking up the information which is missed in the first phase.

3.3. Progressive learning (M.2)

As mentioned above, there is a total of K hidden feature maps in each model, *i.e.* T, S and A, however, only the final one is used as reference in (M.1). In this way, even with the help of A, some low-level information may still get lost. To learn knowledge from T more completely, one straightforward way is to make knowledge transferred in every block and train S and A progressively, as shown in Fig. 3(M.2).

The training process of a particular block is similar to the end-to-end training in (M.1) by keeping all other blocks fixed. The only difference is that the input is replaced from image $\mathbf{x}$ to the residual-error-involved feature map from previous block. With this strategy, S, in addition with A, is able to get both high-level and low-level information from T, resulting in better mimicking.

3.4. Integrating A into S (M.3)

Although the block-by-block learning scheme mentioned in Section 3.3 manages to transfer knowledge precisely, training (M.2) is very complex, in that S and A are required to learn individually for every block. For example, assuming that there are 4 blocks in each model, 8 experiments should be conducted in sequence and user should monitor the entire process so that a new experiment can be started after the previous one finished. To this end, we propose (M.3) in Fig. 3 by integrating A into S, forming a unified framework where A can be trained from end to end.

There are mainly two improvements compared to (M.2). Firstly, the input feature map of each block in S is no longer refined by the residual error produced by A such that all blocks of S can be pre-trained with Eq. (3) as a whole. But the inputs of the blocks in A are still summed up with the residual error. The reason in doing so is that S learns the feature map of T coarsely and hence does not need precise input. In contrast, A requires the input to be precise so as to learn the residual error accurately. Secondly, A learns the residual errors of feature maps from different levels simultaneously with

$$\min_{\Theta_A} \mathscr{L}_A = \sum_{i=1}^{K} ||(\mathbf{f}_i^T - \mathbf{f}_i^S) - \mathbf{f}_i^A||_2^2. \tag{5}$$

3.5. Separating A from S

RKD is a novel model compression method that is able to further distill the knowledge from T with the help of A (M.1). Besides improving performance (M.2) and simplifying the training process (M.3), another crux of RKD is how to choose the model structure of A. One solution is to employ an existing model to serve as an assistant model A, such as using ResNet-18, but when using this strategy, the model size may increase due to the introduction of additional computation.

To keep the total computational cost almost the same as student model, we figure out another strategy to separate A from S model directly. The basic idea is to divide a wide model into two thinner ones by assigning the number of channels, while both derived models share the same model structure, *e.g.* the number of layers and kernel size of each layer. More concretely, we firstly divide the original student model S_{init} into K blocks and for each block we split S_{init} into the final student network S and assistant network A. Here, we set up a splitting proportion. Taking four-to-one proportion as an instance, a block with 100 channels will be

separated into two parts with 80 channels for student model and 20 channels for assistant model, and each block will be divided according to this proportion. In practice, the computational cost is not linear with the number of channels in a model. We use floating-point operations (FLOPS) as the measurement to evaluate the computational cost. After splitting, we will obtain two sub-models: S and A.

3.6. Implementation details

In this work, we choose networks from ResNet family [6] to serve as T, S, and A. To compute residual error, it requires the feature maps of these models to have the same shape, especially along the channel dimension. Therefore, an additional convolutional layer with 1×1 kernel size is applied to deal with the shape-mismatch case. According to the structure of ResNet, four sets of residual blocks are employed. We directly use the outputs of these blocks as hidden feature maps, whose spatial resolutions are 56×56, 28×28, 14×14, 7×7 respectively. T is directly trained with Eq. (2) using stochastic gradient descent (SGD) optimizer with momentum equal to 0.9. The learning rate is initialized as 0.1 and drops to 10% when the model converges until reaching 0.0001. When training S and A, the same optimizer and learning rate decay policy are applied. Differently, after the final feature map reaching optima, the classifier of S is trained from scratch to fit labelled data with Eq. (2).

4. Experiments

We perform extensive experiments to verify the effectiveness of our proposed RKD. Section 4.1 studies how the size of the assistant model will affect the learning of residual error. Section 4.2 evaluates the performances of different variants of RKD mentioned in Section 3.2, Section 3.3, and Section 3.4. Section 4.3 explores the model separation method presented in Section 3.5. Section 4.4 compares RKD with state-of-the-art KD methods on different image classification datasets, including CIFAR-100 and ImageNet. We will firstly introduce the datasets used in this work.

Datasets. CIFAR-100 [37] is a commonly used object recognition dataset, which has 100 classes containing 600 images each. Among them, 500 images are considered as training samples while 100 as testing. Following prior work [19,18,17,20], we use CIFAR-100 to evaluate the performance of knowledge transfer. ImageNet [38] is a 1000-categories dataset consisting of 1.2 M training samples and 50 K validation samples. It is widely applied in image classification task. This work uses ImageNet to verify whether RKD works well in large-scale experiments.

4.1. Evaluation on model size of A

Recall that RKD employs an additional model, A, to aid the knowledge distillation process by learning the residual error between S and T. However, beyond transferring knowledge more completely, the introduction of A will also increase the model size, which is opposite to the primary goal of model compression. Accordingly, we expect A to improve the performance but barely affect the computational cost. To verify whether RKD has such property, we carry out several experiments on ImageNet dataset with different models serving as A, as shown in Table 1(b). Here, we use ResNet-18 to learn from ResNet-34, whose performances are listed in Table 1(a). To validate the effect of residual learning in RKD, we also prepare a baseline by training the student model to mimic the feature map of teacher, yet without the help of A. ResNet-18–1/2 represents a model with ResNet-18 structure but each layer only has half number of channels, hence the computational cost is around 25% of ResNet-18. Similarly, ResNet-18–1/4 only keeps a quarter of channels per layer.

From the results in Table 1(b), we have three observations. (i) All four experiments perform better than mimicking feature map *w/o A*, suggesting that RKD is able to transfer knowledge from T to S more accurately. Besides, we also compute the l_2 distance between the feature maps of T and S (summed up with the output of A). The last column in Table 1 tells that A manages to learn the residual error and then pushes the feature map of S close to that of T. (ii) RKD is able to achieve better accuracy with negligible computational cost increments. For example, we get 0.87% higher accuracy by employing ResNet-18–1/4 as A, which only consumes 6.25% computing resources compared to S. This is because learning residual error is more easily as described in [6]. (iii) Increasing the model size of A will not always result in performance gain. Interestingly, even adopting ResNet-34 (same as teacher) as A, the accuracy is still far behind that of T. Therefore, A just requires a lightweight model and major computing resources should be allocated to S to get satisfactory results.

4.2. Evaluation on different training schemes

In this section, we evaluate the capabilities of three different training schemes of our proposed RKD, including (M.1) learning residual error of the final feature map, (M.2) learning residual error of multi-level feature maps progressively, and (M.3) integrating A into S to get multi-level knowledge from end to end. We train three independent models on ImageNet dataset with above transferring strategies respectively. Comparative results are shown in Table 1 (c).

We can easily tell that both M.2 and M.3 beat M.1 with more than 0.4% accuracy, suggesting that RKD benefits a lot from the low-level supervision from T. l_2 distances of M.2 and M.3 are also closer than that of M.1, meaning that learning multi-level information helps better knowledge transfer. Furthermore, M.2 significantly improves the performance of the student model (from 69.572% to 71.787%), demonstrating the effectiveness of RKD in distilling the knowledge of T for a second time with the help of A. Although M.2 shows the best performance, training M.2 is tedious, as explained in Section 3.4. On the contrary, M.3 is able to achieve similar result but with simpler training procedure. Accordingly, we recommend to use M.3 in practice.

4.3. Evaluation on model separation

As mentioned before, the success of RKD benefits from the introduction of the assistant model, A, which will increase the entire model size. To verify whether the performance gain comes from the additional computation or the idea of further distillation, we explore a way to separate a model apart to serve as S and A respectively, which is described in Section 3.5. Table 2 shows the results of experiments on ImageNet dataset with different dissection ratios between S and A. To simplify the training procedure, M.3 is used in this section.

From Table 2, we obtain four conclusions as following. (i) After separation, the two derived models, *i.e.* S and A, have nearly the same computational cost as the original model. For each experiment, the proportion between the computational costs of S and A is also as desired, demonstrating the effectiveness of our proposed model separation method. (ii) RKD works well when S dominates the distillation process, *e.g.* S consumes 90% computing resources. The experiment with 50%-50% proportion between S and A even performs worse than training S *w/o* guidance from T (68.513% *v.s.* 69.572%). This is consistent with the observation in Section 4.1. In other words, S should learn the majority knowledge from T,

Table 1
Evaluation on model size of assistant model on ImageNet. (a) shows the baseline results, whilst (b) and (c) are explained in Section 4.1 and Section 4.2 respectively.

	Method	Model	Top-1 (%)	Top-5 (%)	l_2 Distance
(a)	Teacher	ResNet-34	73.554	91.456	-
	Student	ResNet-18	69.572	89.244	2514
	KD	ResNet-18	70.294	89.349	1517
(b)	M.1	ResNet-18 (*S*) & ResNet-18–1/4 (*A*)	71.161	89.926	1034
	M.1	ResNet-18 (*S*) & ResNet-18–1/2 (*A*)	71.230	89.933	881
	M.1	ResNet-18 (*S*) & ResNet-18 (*A*)	71.354	90.000	839
	M.1	ResNet-18 (*S*) & ResNet-34 (*A*)	71.376	90.012	721
(c)	M.1	ResNet-18 (*S*) & ResNet-18–1/2 (*A*)	71.230	89.933	881
	M.2	ResNet-18 (*S*) & ResNet-18–1/2 (*A*)	**71.787**	**90.254**	**693**
	M.3	ResNet-18 (*S*) & ResNet-18–1/2 (*A*)	71.631	90.226	701

Table 2
Experiments on ImageNet by separating *A* from *S* with different proportions. Details are explained in Section 4.3.

Method	Model	GFLOPS	Top-1 (%)	Top-5 (%)
Teacher	ResNet-34	3.4121	73.554	91.456
Student	ResNet-18	1.6895	69.572	89.244
M.3	ResNet-18–50% (*S*) & ResNet-18–50% (*A*)	1.6558	68.513	88.528
M.3	ResNet-18–70% (*S*) & ResNet-18–30% (*A*)	1.6850	70.481	89.675
M.3	ResNet-18–80% (*S*) & ResNet-18–20% (*A*)	1.6875	71.075	89.849
M.3	ResNet-18–90% (*S*) & ResNet-18–10% (*A*)	1.6888	**71.461**	**90.195**
Teacher	ResNet-101	7.5230	77.985	93.872
Student	ResNet-50	3.8084	76.013	92.980
M.3	ResNet-50–90% (*S*) & ResNet-50–10% (*A*)	3.8085	**77.692**	**93.023**

while *A* only complements *S* by capturing missed information. (iii) With same model size, RKD is also capable of improving the performance of student model remarkably (from 69.572% to 71.461%). Consequently, it is mainly the effect of learning residual error instead of the incremental computational cost that aids *S* in acquiring more knowledge from *T*. (iv) The proposed separation method is generally applicable. Besides using ResNet-18 to learn from ResNet-34, we also train ResNet-50 to mimic ResNet-101 to verify that our method can separate not only shallow model, but deep model as well. Results in Table 2 suggest that RKD improves ResNet-50 with 1.7% higher accuracy. It is also worth mentioning that after distilling knowledge from *T*, *S* achieves competitive results as *T* (77.692% *v.s.* 77.985%), significantly narrowing down the performance gap.

4.4. Evaluation on different datasets

This section conducts experiments on different datasets, *i.e.* CIFAR-100 and ImageNet, to further validate the efficiency of RKD. We also compare RKD with state-of-the-art knowledge transfer methods, including KD [10], FitNets [15], and AT [18]. Here, to make fair comparisons, we set $T = 4$ and $\lambda = 16$ for KD method following [10]. For details, FitNets uses the last layer of the second block in ResNet as hint layer, and AT [18], FT [39], AB [40], SP [41] use four hidden feature maps produced by four block respectively to compute distillation loss. Our RKD also uses four output feature maps after each block in ResNet. Note that, M.3 in addition with model separation method are used in this section, hence the total computational cost is nearly the same as alternative approaches.

4.4.1. Evaluation on CIFAR-100

Like existing work [15,18] did, we firstly carry out small-scale experiments on CIFAR-100 dataset. Table 3 shows the comparison results, where RKD outperforms the baseline model with 2.75% higher accuracy and nearly achieves same performance as teacher model. We also beat the second competitor by 0.42% accuracy.

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

Method	Model	Top-1 (%)	Top-5 (%)
Teacher	ResNet-34	73.045	91.545
Student	ResNet-18	68.062	89.598
KD [10]	ResNet-18	72.393	91.062
FitNets [15]	ResNet-18	71.662	90.277
AT [18]	ResNet-18	70.741	90.036
FT [39]	ResNet-18	71.773	90.829
AB [40]	ResNet-18	72.056	90.993
SP [41]	ResNet-18	72.139	91.041
RKD	ResNet-18	**72.816**	**91.408**

Table 4
Comparison results of image classification on ImageNet dataset.

Method	Model	Top-1 (%)	Top-5 (%)
Teacher	ResNet-34	73.554	91.456
Student	ResNet-18	69.572	89.244
D [10]	ResNet-18	70.759	89.806
FitNets [15]	ResNet-18	70.662	89.232
AT [18]	ResNet-18	70.726	90.038
FT [39]	ResNet-18	70.981	90.056
AB [40]	ResNet-18	71.262	90.128
SP [41]	ResNet-18	71.047	90.109
RKD	ResNet-18	71.461	90.195
RKD + AT	ResNet-18	71.536	90.258
RKD + AB	ResNet-18	**71.582**	**90.317**

4.4.2. Evaluation on ImageNet

We also conduct larger-scale experiments on ImageNet dataset, whose results are shown in Table 4. We can easily conclude that RKD improves the baseline model with 1.9% accuracy, significantly surpassing the alternative KD methods. Furthermore, by cross-comparing the results in Table 3 and Table 4, we find out that other methods may perform inconsistently on different datasets. For example, AT works well on ImageNet but fails on CIFAR-100. By contrast, RKD shows stronger stability and robustness.

We also perform experiments by combining RKD with other knowledge transfer methods, *e.g.* AT and AB. Last row in Table 4

Table 5
Success rate of applying adversarial attacks on proposed method and other state-of-the-art knowledge distillation method.

Attack	ResNet-18	ResNet-34	KD	FitNets	AT	FT	AB	SP	RKD
L_0 Attack	0.9	12.9	9.7	9.1	8.2	5.9	7.6	8.1	**9.9**
L_2 Attack	1.2	15.6	**10.8**	9.6	8.3	6.6	7.2	8.4	10.3
L_∞ Attack	0.6	10.3	10.5	9.1	7.8	6.9	6.4	8.2	**10.7**

suggests that the idea of RKD can be also generalized to other KD methods as long as the residual error is well defined. To this end, we believe RKD is a promising solution to model compression problem that is worth exploring.

4.5. Evaluation of robustness

In this section, we demonstrate the robustness of our approach with adversarial samples. As decribed in [42,43], a small amount of transformation in an image may cause changes in the classification results. A robust network should not change its performance when facing such attacks. As a result, we apply experiments on CIFAR-100 with three attack algorithms which are optimized to produce adversarial examples minimizing L_0,L_2,L_∞ distance metrics respectively, as introduced in [42], and compare its robustness of our approach with other distillation methods.

Table 5 shows the results. In the face of adversarial attacks, the baseline model of ResNet-18 only achieves less than 2% in accuracy. In all of the attacks, RKD achieves about 10% in performance which is 5 times of the baseline model. Comparing with other distillation methods and teacher model, the proposed method shares the most similar robustness with teacher and are more stable than others. The experiment results shows that with the proposed method, we can achieve an more robust model than the baseline model and other distilled models.

5. Conclusion

This paper presents a novel model compression approach, called RKD, which further distills the knowledge from teacher model with an assistant (A). By mimicking the residual error between the feature maps of S and T, A is able to complement S with the missing information and thus improves the performance significantly. In addition, we also find a way to split a model apart to S and A, such that the computational cost will not increase after introducing A. Numerous experimental results demonstrate the effectiveness and efficiency of RKD.

CRediT authorship contribution statement

Mengya Gao: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft. **Yujun Wang:** Writing - review & editing, Validation, Visualization. **Liang Wan:** Supervision, Writing - review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

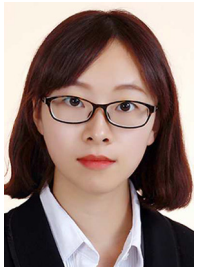
Acknowledgements

This work was supported by the National Natural Science Foundation of China (Project No. 61572354).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, NeurIPS (2012).
[2] Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, CVPR (2014).
[3] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: towards real-time object detection with region proposal networks, NeurIPS (2015).
[4] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing 187 (2016) 27–48.
[5] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, ICLR (2015).
[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CVPR (2016).
[7] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, CVPR (2017).
[8] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[9] C. Bucilua, R. Caruana, A. Niculescu-Mizil, Model compression, SIGKDD (2006).
[10] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, NeurIPS Workshop (2014).
[11] Y. LeCun, J.S. Denker, S.A. Solla, Optimal brain damage, NeurIPS (1990).
[12] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in, NeurIPS (2015).
[13] K. Hwang, W. Sung, Fixed-point feedforward deep neural network design using weights +1, 0, and -1, in: SiPS Workshop (2014).
[14] X. Zhang, J. Zou, X. Ming, K. He, J. Sun, Efficient and accurate approximations of nonlinear convolutional networks, CVPR (2015).
[15] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: Hints for thin deep nets, ICLR (2015).
[16] Z. Huang, N. Wang, Like what you like: Knowledge distill via neuron selectivity transfer, arXiv preprint arXiv:1707.01219 (2017).
[17] J. Yim, D. Joo, J. Bae, J. Kim, A gift from knowledge distillation: fast optimization, network minimization and transfer learning, CVPR (2017).
[18] S. Zagoruyko, N. Komodakis, Paying more attention to attention: iimproving the performance of convolutional neural networks via attention transfer, ICLR (2017).
[19] J. Ba, R. Caruana, Do deep nets really need to be deep?, NeurIPS (2014).
[20] S.H. Lee, D.H. Kim, B.C. Song, Self-supervised knowledge distillation using singular value decomposition, ECCV (2018).
[21] M. Gao, Y. Shen, Q. Li, C.C. Loy, X. Tang, Feature matters: a stage-by-stage approach for knowledge transfer, arXiv preprint arXiv:1812.01819 (2018).
[22] T. Chen, I. Goodfellow, J. Shlens, Net2net: accelerating learning via knowledge transfer, ICLR (2016).
[23] V. Belagiannis, A. Farshad, F. Galasso, Adversarial network compression, arXiv preprint arXiv:1803.10750 (2018).
[24] A. Ashok, N. Rhinehart, F. Beainy, K.M. Kitani, N2n learning: network to network compression via policy gradient reinforcement learning, ICLR (2018).
[25] H. Wang, H. Zhao, X. Li, X. Tan, Progressive blockwise knowledge distillation for neural network acceleration, IJCAI (2018).
[26] T. Furlanello, Z.C. Lipton, M. Tschannen, L. Itti, A. Anandkumar, Born again neural networks, in: ICML, 2018.
[27] X. Lan, X. Zhu, S. Gong, Self-referenced deep learning, ACCV (2018).
[28] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, CVPR (2018).
[29] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G.E. Dahl, G.E. Hinton, Large scale distributed neural network training through online distillation, ICLR (2018).
[30] X. Zhu, S. Gong, et al., Knowledge distillation by on-the-fly native ensemble, NeurIPS (2018).
[31] S.-I. Mirzadeh, M. Farajtabar, A. Li, H. Ghasemzadeh, Improved knowledge distillation via teacher assistant, AAAI (2020).
[32] J.-H. Kim, S.-W. Lee, D. Kwak, M.-O. Heo, J. Kim, J.-W. Ha, B.-T. Zhang, Multimodal residual learning for visual qa, NeurIPS (2016).
[33] J. Pang, W. Sun, J.S. Ren, C. Yang, Q. Yan, Cascade residual learning: a two-stage convolutional neural network for stereo matching, ICCV Workshops (2017).
[34] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: residual learning of deep cnn for image denoising, TIP 26 (7) (2017) 3142–3155.
[35] J. Kim, J. Kwon Lee, K. Mu Lee, Accurate image super-resolution using very deep convolutional networks, CVPR (2016).
[36] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C.C. Loy, Y. Qiao, X. Tang, Esrgan: enhanced super-resolution generative adversarial networks, ECCV Workshops (2018).
[37] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Technical Report, Citeseer, 2009.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, CVPR (2009).
[39] S.P.J. Kim, N. Kwak, Paraphrasing complex network: network compression via factor transfer, in: Advances in Neural Information Processing Systems (NIPS), 2018.
[40] B. Heo, M. Lee, S. Yun, J.Y. Choi, Knowledge transfer via distillation of activation boundaries formed by hidden neurons, AAAI (2019).
[41] F. Tung, G. Mori, Similarity-preserving knowledge distillation, in: International Conference on Computer Vision (ICCV), 2019.
[42] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (sp), IEEE, 2017, pp. 39–57.
[43] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy (SP), IEEE, 2016, pp. 582–597.



Mengya Gao received her B.S and is persuing her M.S in Computer Software Engineering from Tianjin University. Her research is focused on model acceleration and knowledge distillation.

Yujun Wang received his B.S and M.S in Computer Science from The University of Hong Kong in 2015 and 2017 respectively. He is pursuing his PhD in Computer Science from The University of Hong Kong since 2017. His research interests are in knowledge transfer and face recognition.

Liang Wan received the B.Eng and M.Eng degrees in computer science and engineering from Northwestern Polytechnical University, P.R. China, in 2000 and 2003, respectively. She obtained a Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong in 2007. She is currently a full Professor in the college of Intelligence Computing, Tianjin University, P. R. China. Her research interest is mainly on image processing and computer vision, including panoramic image processing, image tracking, and medical image analysis.