ResKD: Residual-Guided Knowledge Distillation
Xuewei Li*, Songyuan Li*, Bourahla Omar, Fei Wu, and Xi Li

Abstract—Knowledge distillation, aimed at transferring the knowledge from a heavy teacher network to a lightweight student network, has emerged as a promising technique for compressing neural networks. However, due to the capacity gap between the heavy teacher and the lightweight student, there still exists a significant performance gap between them. In this paper, we see knowledge distillation in a fresh light, using the knowledge gap, or the residual, between a teacher and a student as guidance to train a much more lightweight student, called a res-student. We combine the student and the res-student into a new student, where the res-student rectifies the errors of the former student. Such a residual-guided process can be repeated until the user strikes the balance between accuracy and cost. At inference time, we propose a sample-adaptive strategy to decide which res-students are not necessary for each sample, which can save computational cost. Experimental results show that we achieve competitive performance with 18.04%, 23.14%, 53.59%, and 56.86% of the teachers’ computational costs on the CIFAR-10, CIFAR-100, Tiny-ImageNet, and ImageNet datasets. Finally, we do thorough theoretical and empirical analysis for our method.

Index Terms—Knowledge Distillation, Residual, Sample-Adaptive.

I. INTRODUCTION

As deep learning goes deeper, state-of-the-art neural networks [11, 12, 13, 14] have obtained better and better performance, and yet demand more and more computational resources. While models with large capacity can achieve high accuracy, they are impractical for resource-limited devices such as embedded systems. To this end, researchers have studied cost-effective networks [5, 6, 7, 8] and efficient training strategies [9, 10, 11, 12, 13, 14, 15, 16, 17]. Knowledge distillation (KD) [18] has emerged as a compression technique where an analogy of the teacher-student relationship is drawn to explain the idea that the knowledge of a powerful yet heavy teacher network can be distilled into a lightweight student network by adding a loss term that encourages the student to mimic the teacher.

Due to the capacity gap between a heavy teacher and a lightweight student, there is still a significant performance gap between them. Existing KD methods have made efforts to modify the loss term to improve the student’s performance [9, 19, 20, 21, 22]. However, the discrepancy between a teacher and a student can be considered as knowledge, and it remains relatively unexplored in knowledge distillation.

In this paper, we see knowledge distillation in a fresh light, using the knowledge gap between a teacher and a student as guidance. First, we train a student network \( S_0 \) from a teacher \( T \) as usual. Then, we train a much more lightweight network, named a res-student, to learn the knowledge gap, or the residual, between the teacher \( T \) and the student \( S_0 \). The combination of the student \( S_0 \) and the res-student \( R_1 \) becomes a new student \( S_1 \), where \( R_1 \) corrects the errors of \( S_0 \). Similarly, an even more lightweight res-student \( R_2 \) can be used to learn the knowledge gap between \( T \) and \( S_1 \) to build a new student \( S_2 \). Such a residual-guided process can be repeated until a final student \( S_n \) is obtained. We call our framework residual-guided knowledge distillation (ResKD). The idea is akin to approximating functions by a polynomial. In a series expansion of a function, a higher-order polynomial would be a better approximation but would demand more computations. Similarly, a higher-order ResKD-style student would be closer to the teacher but would be more expensive. Users can control the total capacity of the final student network \( S_n \) by setting hyper-parameters for termination.

In addition, the knowledge gap mentioned above is different from sample to sample. For instance, given an \( S_2 \) student, we observed that, for some images, \( S_1 \) (i.e., \( S_0 + R_1 \)) or even \( S_0 \) itself has highly confident scores, while \( R_2 \) or \( R_1 + R_2 \) has little contributions. It is unnecessary to use all the res-students for each sample at inference time. Thus, we introduce a sample-adaptive strategy for the inference phase. For each sample, if the confidence of \( S_i \) is high enough, we truncate the unnecessary res-students to save computational cost.

We do experiments on several standard benchmarks. The experimental results show that we achieve competitive performance with 18.04%, 23.14%, 53.59%, and 56.86% of the teacher’s FLOPs for CIFAR-10, CIFAR-100, Tiny-ImageNet, and ImageNet respectively. Also, we apply our ResKD framework to different KD methods and show that the framework is generic to knowledge distillation methods. Finally, we analyze the effectiveness of this idea and use informativeness [23] to visualize the bridging gap process.

Our contributions in this paper are summarized as follows:
- We design a residual-guided learning method that uses a series of res-students to bridge the gap between a student network and a teacher network.
- We introduce a sample-adaptive strategy at inference time to make our framework adaptive to different samples to save the cost of additional res-student networks.
- We evaluate our method on different datasets and perform detailed experiments that showcase the importance of each part of the framework.

II. RELATED WORK

A. Knowledge Distillation

We categorize knowledge distillation methods in terms of the number of stages. Traditionally, knowledge distillation is a two-stage method, in which a teacher network is trained first,
and then a student network is trained under the guidance of the teacher network. Bucilă et al. [24] pioneered the idea of transferring the knowledge from a cumbersome model to a small model. Hinton et al. [18] popularized this idea by the concept of knowledge distillation (KD), in which a student neural network is trained with the benefit of the soft targets provided by teacher networks. Compared to traditional one-hot labels, the output from a teacher network contains more information about the fine-grained distribution of data, which helps the student achieve better performance. Recently, many works have focused on improving the information propagation way or putting strictness to the distillation process via optimization [25], [26], [27], [19], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] to teach the student better. For example, Peng et al. [19] proposed that a student network should not only focus on mimicking from a teacher at an instance level, but also imitating the embedding space of a teacher so that the student can possess intra-class compactness and inter-class separability. In addition, the effect of different teachers is also researched [38], [39], [40]. For example, Sau et al. [39] proposed an approach to simulate the effect of multiple teachers by injecting noise to the training data and perturbing the logit outputs of a teacher. In such a way, the perturbed outputs not only simulate the setting of multiple teachers but also result in noise in the softmax layer, thus regularizing the distillation loss. With the help of many teachers, the student is improved a lot. Kang et al. [40] used Neural Architecture Search (NAS) to acquire knowledge for both the architecture and the parameters of the student network from different teachers. Besides the classic image classification task, KD can also be used in many other different fields, such as face recognition [41], visual question answering [42], video tasks [43], [44] etc.

Recently, some KD methods have been proposed to have less or more than two stages. For one thing, KD can be a one-stage strategy [45], [9]. Zhang et al. [9] proposed that a pool of untrained student networks with the same network structure can be used to simultaneously learn the target task together instead of the traditional two-stage knowledge distillation strategy. For another, a line of research [46], [47] focuses on KD methods with more than two stages. In [46], several Teacher Assistant networks are used to transfer the knowledge from a teacher more softly and effectively. The teacher propagates its knowledge to the assistant networks first and then the assistant networks propagate its knowledge to the student network. In [47], the student network has the same architecture as the teacher network at the beginning. The last block of the student network is replaced by a simple block and trained at the first stage. Next, the penultimate block is replaced similarly and the last two blocks are trained. In this style, all blocks are trained after several stages and a lightweight student network is achieved in the end.

In this paper, we mainly focus on how to use the gap between the teacher and the student as knowledge. We use a series of lightweight networks, named res-students, to learn the gap in a multi-stage manner.

B. Ensemble Methods

Ensemble methods, which have been studied extensively for improving model performance [48], [49], [50], are strategies to combine models by averaging, majority voting or something else, which means several models having the same status are used to improve final performance. Ensembles of models perform at least as well as each of its ensemble members [51]. There are several lines of research of ensemble methods; introducing different regularization, reducing training time [52] and saving test time [24].

For combining knowledge distillation and the ensemble idea, Lan et al. [53] proposed to aggregate the logits of several homogeneous student models to become an ensemble teacher and then to distill the knowledge from the teacher. On the contrary, we first carry on knowledge distillation to build a student and res-students and then combine them. Also, the roles that the student and res-students play are not the same. The student acquires the knowledge from the teacher, while the res-students acquire the knowledge from knowledge gaps. Furthermore, we introduce a sample-adaptive strategy to decide which res-students to use at inference time.

C. Residual Learning

Residual learning has been studied in face alignment for improving the final performance by learning from the gap between the suboptimal results and the target. Specifically, regressors for face alignment refine the current suboptimal results (landmark [54], [55], [56], [57], shape estimation [58] or heatmaps [59]) to approach the target. In [56], local binary features are extracted first and then a learning global linear regression is applied to generate the final landmark. In [58], a cascade shape regression are employed to generate the optimal pose from the initial pose according to the shape-indexed features of the input image. In [59], heatmaps and suboptimal landmarks are extracted by a multi-order cross geometry-aware model from the original images and a multi-order high-precision hourglass network is applied to achieve the heatmap subpixel face alignment and generate the optimal landmarks.

Knowledge distillation aims to deliver a lightweight student network to approximate a given teacher network. At training time, our ResKD learns not only from the ground truth but also from the residual between the teacher network and the former student network in a coarse-to-fine manner. When learning from the teacher network, our residual-guided supervision can be applied at different levels: the final output (just as the ground truth), intermediate feature maps, structural knowledge, etc.

III. RESIDUAL-GUIDED KNOWLEDGE DISTILLATION

In this section, we present our residual-guided knowledge distillation framework (ResKD). First, we briefly introduce the background. Second, we describe our main idea of residual-guided learning. Next, we propose a sample-adaptive strategy at inference time. Finally, we put all the things together to build a whole framework. For convenience, Table 1 summarizes the notations we use in this section.
A. Background and Notations

We first describe a general formulation of knowledge distillation, and then introduce the residual-guided knowledge distillation in this formulation.

We define a teacher network $T$ as a function

$$T = f(x, w_T, \alpha_T),$$

where $x$ is an input image, $w_T$ is the weights of $T$, and $\alpha_T$ is its architecture parameters. Similarly, $S = f(x, w_S, \alpha_S)$ denotes a student network. For convenience, a network and its logits are used interchangeably. The goal of knowledge distillation is to learn $w_S$ so that the results of the student $S$ are as close as possible to the teacher $T$.

Classic knowledge distillation from a teacher is done in two steps. First, the student architecture $\alpha_S$ is determined. Then, $w_S$ is optimized by

$$\arg\min_{w_S} \sum_{j=1}^{N} L_{KD} \left( f(x^{(j)}, w_S, \alpha_S), f(x^{(j)}, w_T, \alpha_T) \right),$$

where $x^{(j)}$ is an input image in a dataset with $N$ training samples and $L_{KD}$ is defined as

$$L_{KD}(S, y^{(j)}, T, \tau, t) = \tau \cdot t^2 \cdot L_{T-S}(\sigma(S), \sigma(T)) + (1 - \tau) \cdot L_{CE}(\sigma(S), y^{(j)}),$$

where $y^{(j)}$ is the label of the input image $x^{(j)}$, $\sigma(\cdot)$ is the softmax function, $L_{CE}$ is the conventional cross-entropy loss function, $\tau$ and $t$ are scalar hyper-parameters, and $L_{T-S}$ is the loss between the student’s logits and the teacher’s logits, e.g., the Kullback–Leibler divergence.

The optimized $w_S$ from Eq. (2) makes the predictions obtained from $S = f(x, w_S, \alpha_S)$ as close to those obtained by...
\[ T = f(\mathbf{x}, w_T, \alpha_T) \] as possible, but we still observe a difference \( \Delta \) between the teacher’s and student’s logits:

\[ \Delta = T - S, \quad (4) \]

which is what we propose to reduce as follows.

B. Residual-Guided Learning

We propose a framework distilling knowledge in a residual-guided fashion: we use the difference, or the residual, between a teacher and a student as guidance. Given a teacher network, we first train a student network by classic knowledge distillation. Then, we introduce another student, called a res-student, to learn their residual. After that, we combine the original student and the res-student to form a new student. Such a process can be repeated to add more res-students to further bridge the performance gap, as shown in Fig. 1.

Formally, let the logits of the teacher be \( T \), the logits of the first student be \( S_0 \), and the logits of the res-student networks be \( \{R_i\}_{i=1}^n \). We train \( S_0 \) and \( \{R_i\}_{i=1}^n \) in stages. In stage 0, \( S_0 \) is trained in a classic knowledge distillation manner with \( T \) as the teacher network. The gap can be observed by \( \Delta_0 = T - S_0 \). In stage 1, we use \( S_0 \) as a new teacher, called a res-student, to train a new student \( R_1 \). Once we obtain \( R_1 \), we can define a new student \( S_1 = S_0 + R_1 \). Similarly, in each following stage, we use the gap from the current stage \( \Delta_i = T_i - R_i \) as the next stage teacher network \( T_{i+1} \). We distill a res-student \( R_{i+1} \) from \( T_{i+1} \) and obtain the next student \( S_{i+1} = S_0 + R_1 + \cdots + R_{i+1} \). Finally, we use the student network and res-student networks trained in all \( n+1 \) stages. We feed an input image to all networks independently and sum their logits to obtain the final logits:

\[ S_n = S_0 + \sum_{i=1}^{n} R_i, \quad (5) \]

1) NAS-Assisted Architecture: When we use a res-student to bridge the gap to the teacher, it is better to use NAS to get the architecture of the res-student network instead of a handcrafted one. We also use the knowledge distillation loss function in the search stage when using NAS, which is reflected by adding \( \alpha_S \) as an optimizable parameter in Eq. (2):

\[ \arg \min \limits_{w_S, \alpha_S} \sum_{j=1}^{N} \mathcal{L}_{KD} \left( f(\mathbf{x}^{(j)}, w_S, \alpha_S), f(\mathbf{x}^{(j)}, w_T, \alpha_T) \right). \quad (6) \]

We use the same search space as that of STACNAS [60]. A STACNAS-style neural network consists of a series of building blocks called cells. We define a set of candidate operations \( O \) to be used inside these cells, and from which we will select the best operations for the architecture.

The construction of cells and the whole network can be described as a directed acyclic graph (DAG) as shown in Fig. 2. A cell contains a sequence of \( N \) nodes \( f_1, f_2, \ldots, f_N \) each of which is a stack of feature maps. These nodes are connected by directed edges. An edge \((i, j)\) represents some operation \( o_{k}^{(i,j)}(\cdot) \in O \) that transforms \( f_i \) to \( f_j \). In a certain cell, each node is computed as a weighted sum of all its predecessors:

\[ f_j = \sum_{i<j} \sum_{k \in O} o_{k}^{(i,j)}(\cdot) \cdot (7) \]

where \( o_{k}^{(i,j)}(\cdot) \) is the weight for the operation \( o_{k}^{(i,j)} \) applied to \( f_i \) when calculating the node \( f_j \). We use the same training strategy as that of STACNAS [60] to obtain the final suitable network.

2) Termination Condition: When a ResKD student \( S_i \) is close enough to the teacher \( T \), we can stop the residual-guided process. Here, we define a concept “Energy” to measure the difference between a student and a teacher.

The Energy of network \( S_i \) for a certain sample \( \mathbf{x}^{(j)} \) is

\[ \text{Energy}(S_i, \mathbf{x}^{(j)}) = \| \sigma(S_i(\mathbf{x}^{(j)})) \|^2_2, \quad (8) \]

where \( \sigma(\cdot) \) is the softmax function, and thus the Energy of network \( S_i \) on dataset \( D \) is

\[ \text{Energy}(S_i, D) = \frac{1}{N} \sum_{j=1}^{N} \| \sigma(S_i(\mathbf{x}^{(j)})) \|^2_2, \quad (9) \]

where \( N \) is the total number of samples in dataset \( D \). In this way, the Energy of a network represents its overall confidence on a dataset. When the Energy of \( S_i \) has reached a comparable level of the Teacher’s Energy (e.g., 90%), we can set \( n = i \) and finish the residual-guided learning process.

In practice, we calculate Energy on a validation set \( D_v \) whose data are uniformly sampled from the training set \( D \). The overall training process is shown in Algorithm I.

C. Sample-Adaptive Inference

When we finish the residual-guided learning process, we have \( S_n = S_0 + \sum_{i=1}^{n} R_i \), where \( R_i \) is supposed to bridge the gap \( \Delta_{i-1} \) between \( S_{i-1} \) and \( T \). However, for each sample, \( \Delta_{i-1} \) is also different. For instance, if the sample is easy to recognize, \( \Delta_0 \) can be subtle, and if the sample is difficult, even \( \Delta_2 \) can be considerable. In other words, for an easy sample, \( S_0 \) is enough, but for a difficult sample, we should use more res-students.

Based on the above observation, we propose an adaptive strategy for each sample at inference time. Similar to the Energy idea in Section III-B, we uniformly sample a validation
set $D_0$ from the training set $D$, and calculate an Energy threshold for the final ResKD student $S_n$: \[
\text{TH}_\text{energy} = E(S_n, D_v). \tag{10}
\]

When the Energy of $S_i$ with a sample $x^{(j)}$, i.e., $E(S_i, \{x^{(j)}\})$, is higher than $\text{TH}_\text{energy}$, we set the rest res-students $R_{i+1}, \ldots, R_n$ aside. As a result, given a sample $x^{(j)}$, we define $S_n^{(i)}$ as the student network it uses at inference time:
\[
S_n^{(i)}(x^{(j)}) = S_L(x^{(j)}), \quad 0 \leq L \leq n, \tag{11}
\]
where we use $S_L$ instead of $S_n$ for the sample $x^{(j)}$. For example, let $S_n = S_1$. Given a sample $x^{(j)}$, if $E(S_1, x^{(j)}) > \text{TH}_\text{energy}$, then $L = 1$, which means that we only use $S_0$ and $R_1$, and set $R_2$ and $R_3$ aside. The whole process is shown in Algorithm 2.

D. ResKD: The Whole Framework

When faced with a knowledge distillation problem, our first step is to train a student network $S_0$, which can be handcrafted or searched, under the guidance of the teacher $T$. Next, we start to use our residual-guided learning strategy to find res-students. We train the res-student $R_i$ in a KD manner under the guidance of $T_i$. Such a residual-guided process can be repeated until $S_i$ has achieved the Energy comparable to $T$. At inference time, we apply our sample-adaptive strategy.

IV. EXPERIMENTS

In this section, we first introduce the datasets and protocols we use in Section IV-A. Next, we do ablation studies in Section IV-B1 and Section IV-B2 to evaluate the effectiveness of our strategies. Finally, we evaluate our framework on several datasets in Section IV-B3.

A. Datasets and Protocols

1) CIFAR-10/100: The CIFAR-10/100 dataset consists of 50k training images and 10k testing images in 10/100 classes. We use a weight decay of 0.0001 and a momentum of 0.9. Our models are trained with a mini-batch size of 128. We start with a learning rate of 0.1, divide it by 10 at 150 and 200 epochs, and terminate training at 250 epochs. We follow the simple data augmentation in [1] for training: 4 pixels are padded on each side, and a 32 × 32 crop is randomly sampled from the padded image or its horizontal flip for training. For testing, we only evaluate the single view of the original 32 × 32 image.

2) ImageNet: ImageNet consists of 1000 classes. Our models are trained on the 1.28 million training images, and evaluated on the 50k validation images. Images are randomly resized and a 224 × 224 crop is randomly sampled from the resized image or its horizontal flip for training. For testing, we scale the short side of images to 224, and a 224 × 224 center crop is sampled from the scaled image. When training ResNet series networks, We start with a learning rate of 0.1, divide it by 10 at 30, 60 and 80 epochs, and terminate training at 90 epochs. We use a weight decay of 0.0001 and a momentum of 0.9. When dealing with the network given by neural architecture search, we use a weight decay of 0.0003 and a momentum of 0.9. An exponential learning rate strategy is used to control the learning rate from 0.1 and multiplying the learning rate by 0.92 after each epoch. We use 100 epochs in total. Our models are trained with a mini-batch size of 128.

3) Tiny-ImageNet: Tiny-ImageNet consists of 200 classes. Our models are trained on the 100k training images, and evaluated on the 10k validation images. Images are randomly resized and a 32 × 32 crop is randomly sampled from the resized image or its horizontal flip for training. For testing, we resize the original images to 224 × 224 and evaluate them. When training ResNet series networks, we start with a learning rate of 0.1, divide it by 10 at 40, 80, and 120 epochs, and terminate training at 150 epochs. We use a weight decay of 0.0001 and momentum of 0.9. When dealing with the network given by neural architecture search, we use the same setting as we do on ImageNet. Our models are trained with a mini-batch size of 128.

When it comes to the termination condition of residual-guided learning, we set comparable Energy as 90% of $T$’s Energy. In the following sections, when it is not mentioned, we use $L_2$ loss function for the $L_{T,S}$ in Eq. (3). “KD” in some tables (e.g., Table VI) means using this loss function, too. We do all experiments on four NVIDIA GTX 1080 Ti cards.
TABLE II: Comparison with other ensemble methods on CIFAR-10/100. Res110 is the ResNet-110 teacher network. Res20 (SGD) is a ResNet-20 trained with SGD, and Res20 (KD) is a ResNet-20 trained by KD to learn from the teacher. ‘Ensemble’, ‘Trained together’ and ‘Ours’ share the same architecture where $S_0$ is ResNet-20 and $R_1$ is ResNet-14. ‘Ensemble’ means that ResNet-20 and ResNet-14 are trained independently and the sum of their logits is used for evaluation. ‘Trained together’ means that the combination of ResNet-20 and ResNet-14 is trained together.

| Architecture   | Acc. (%)  | #Params (M) | MFLOPs |
|----------------|-----------|-------------|--------|
| Teacher        |           |             |        |
| Res110         | 94.19 / 72.44 | 1.728 / 1.734 | 255 / 255 |
| Students       |           |             |        |
| Res20 (SGD)    | 91.80 / 68.82 | 0.270 / 0.276 | 41 / 41 |
| Res20 (KD)     | 93.06 / 70.09 | 0.270 / 0.276 | 41 / 41 |
| Res20-14       |     |             |        |
| ensemble       | 93.17 / 70.41 | 0.442 / 0.454 | 68 / 68 |
| trained together | 93.12 / 70.85 | 0.442 / 0.454 | 68 / 68 |
| Ours           | 93.27 / 71.96 | 0.442 / 0.454 | 68 / 68 |

TABLE III: Multi-stage residual-guided learning on CIFAR-10/100. ResX-Y-Z means we use ResNet-X as $S_0$, ResNet-Y as $R_1$, and ResNet-Z as $R_2$. Multi-stage residual-guided learning can further bridge the gap between the teacher and the student. **Bold:** The best results out of students. **Underline:** The second best results out of students.

| Architecture   | Acc. (%)  | #Params (M) | MFLOPs |
|----------------|-----------|-------------|--------|
| Teacher        |           |             |        |
| Res110         | 94.19 / 72.44 | 1.728 / 1.734 | 255 / 255 |
| Students       |           |             |        |
| Res20 (SGD)    | 91.80 / 68.82 | 0.270 / 0.276 | 41 / 41 |
| Res20 (KD)     | 93.06 / 70.09 | 0.270 / 0.276 | 41 / 41 |
| Res20-14       |           |             |        |
| ResKD $S_1$    |           |             |        |
| Res20-14       | 93.27 / 71.96 | 0.442 / 0.454 | 68 / 68 |
| Res20-20       | 93.35 / 72.22 | 0.539 / 0.551 | 82 / 82 |
| ResKD $S_2$    |           |             |        |
| Res20-14-8     | 93.30 / 72.18 | 0.518 / 0.535 | 80 / 80 |
| Res20-14-14    | 93.68 / 72.42 | 0.615 / 0.632 | 94 / 94 |

B. Ablation Study

1) Residual-Guided Learning: First, we validate the effect of a res-student. We use ResNet-110 as the teacher $T$, ResNet-20 as the classic KD student $S_0$, and ResNet-14 as the res-student $R_1$. As illustrated in Table II, our ResKD student $S_1 = S_0 + R_1$ outperforms $S_0$, which means a res-student can correct the errors of a classic KD student.

However, since we obtain a ResKD student by combining two networks, it is arguable whether the effect of the res-student comes from our residual-guided learning. It could be attributed to additional capacity. Thus, it is necessary to compare a ResKD student with other types of combinations, such as an ensemble network, to show the effect of our method. As illustrated in Table II, our method also outperforms the “ensemble” and the “trained together” approaches, which means that using res-students is more effective than using an ensemble.

We can repeat residual-guided learning to further bridge the gap between the student and the teacher. As illustrated in Table III, we train another res-student $R_2$ to build $S_2$, and the performance of $S_2$ is better than $S_1$.

2) Sample-Adaptive Inference: This strategy focuses on striking the balance between performance and cost. Our threshold $T_{\text{energy}}$ based on the validation set is a key hyper-parameter in it. We justify our choice of $T_{\text{energy}}$ in two fields: our $T_{\text{energy}}$ can be generalized to the test set, and the $T_{\text{energy}}$ is effective.

Our sample-adaptive strategy focuses on striking the balance between performance and cost by setting $T_{\text{energy}}$. According to the relationship between performance and cost $w.r.t.$ $T_{\text{energy}}$ as shown in Fig. 3, we can set an appropriate $T_{\text{energy}}$. We set $T_{\text{energy}}$ as 90% of the final student network’s energy (here is $S_1$’s energy) on the validation set to have a good tradeoff. As a result, $T_{\text{energy}}$ is 0.88 for CIFAR-100 and 0.70 for ImageNet. In this case, our sample-adaptive strategy can avoid remarkable FLOPs of $R_1$ and preserve the accuracy at the same time.

We also verify the transfer from our validation set to the test set. We carry out experiments to show the relationship between $T_{\text{energy}}$ and cost on the validation and test set in Fig. 3. We can learn that as the $T_{\text{energy}}$ reduces, the decreasing styles of $R_1$’s cost on both settings are similar, which means the $T_{\text{energy}}$ based on the validation set is reasonable to use on the test set.

3) The Whole Framework, ResKD: We use our whole framework on CIFAR-10, CIFAR-100, Tiny-ImageNet and ImageNet. As illustrated in Table IV and Table V, ResKD achieves consistent results on these datasets. Our ResKD student $S_1 = S_0 + R_1$ where $R_1$’s architecture is searched by NAS outperforms $S_0$ in both accuracy and energy, which means a res-student can also correct the errors of a classic KD student on all four datasets.

C. Discussion

To verify the generalization of our ResKD, we carry out the following four case studies.
TABLE IV: Performance on CIFAR-10 / CIFAR-100. \( T \) is ResNet-110, \( S_0 \) is a ResNet-20, \( R_1 \) is a res-student whose architecture is searched by NAS. \( S_1 \) means using both \( S_0 \) and \( R_1 \). ‘SA’ means using our sample-adaptive strategy at inference time. Energy is the metric we mention in Section III-B. **Bold:** The best results out of students. **Underline:** The second best results out of students.

| Architecture | Optimizer | SA | Acc. (%) | #Params (M) | MFLOPs | MFLOPs proportion to \( T \) (%) | Energy |
|--------------|-----------|----|----------|------------|--------|---------------------------------|--------|
| **Teacher**  |           |    |          |            |        |                                 |        |
| Res110       | SGD       |    | 94.19 / 72.44 | 1.728 / 1.734 | 255 / 255 | 100 / 100 | 0.9993 / 0.9894 |
| **Students** |           |    |          |            |        |                                 |        |
| \( S_0 \)    | KD        |    | 93.06 / 70.09 | 0.270 / 0.276 | 41 / 41  | **16.08 / 16.08** | 0.9923 / 0.8380 |
| \( S_1 \)    | KD        |    | **93.92 / 74.06** | 0.453 / 0.485 | 79 / 81  | 30.98 / 31.76 | 0.9958 / 0.9204 |

ResKD students

| Architecture | Optimizer | SA | Acc. (%) | #Params (M) | MFLOPs | MFLOPs proportion to \( T \) (%) | Energy |
|--------------|-----------|----|----------|------------|--------|---------------------------------|--------|
| **Teacher**  |           |    |          |            |        |                                 |        |
| Res50        | SGD       |    | 70.42 / 75.50 | 23.92 / 25.56 | 4117 / 4119 | 100 / 100 | 0.9963 / 0.7782 |
| **Students** |           |    |          |            |        |                                 |        |
| \( S_0 \)    | KD        |    | 67.90 / 70.02 | 11.28 / 11.69 | 1821 / 1821 | **44.23 / 44.22** | 0.9805 / 0.6575 |
| \( S_1 \)    | KD        |    | **68.99 / 73.73** | 16.38 / 19.71 | 2206 / 2342 | 53.59 / 56.86 | - |

TABLE V: Performance on Tiny-ImageNet / ImageNet. \( T \) is ResNet-50, \( S_0 \) is a ResNet-18, \( R_1 \) is a res-student whose architecture is searched by NAS. \( S_1 \) means using both \( S_0 \) and \( R_1 \). ‘SA’ means using our sample-adaptive strategy at inference time. Energy is the metric we mention in Section III-B. **Bold:** The best results out of students. **Underline:** The second best results out of students.

| Architecture | Optimizer | \( \mathcal{L}_{KD} \) | Acc. (%) | #Params (M) | MFLOPs |
|--------------|-----------|-----------------|----------|------------|--------|
| **Teacher**  |           |                  |          |            |        |
| Res110       | SGD       | -               | 94.19    | 1.728      | 255    |
| **Students** |           |                  |          |            |        |
| \( S_0 \)    | KD        | \( L_2 \)      | 91.80    | 0.270      | 41     |
| \( S_1 \)    | KD        | \( L_2 \)      | 93.06    | 0.270      | 41     |
| \( S_1 \)    | KD        | \( KL \)       | 92.76    | 0.270      | 41     |
| \( S_1 \)    | KD        | \( DML \)      | 92.53    | 0.270      | 41     |
| \( S_1 \)    | RD        | \( L_2 \)      | 92.45    | 0.270      | 41     |

ResKD students

| Architecture | Optimizer | \( \mathcal{L}_{KD} \) | Acc. (%) | #Params (M) | MFLOPs |
|--------------|-----------|-----------------|----------|------------|--------|
| **Students** |           |                  |          |            |        |
| \( S_0 \)    | KD        | \( L_2 \)      | 93.27    | 0.442      | 41     |
| \( S_0 \)    | KD        | \( KL \)       | 93.50    | 0.442      | 68     |
| \( S_0 \)    | KD        | \( L_2 \)      | 93.35    | 0.539      | 87     |
| \( S_0 \)    | KD        | \( KL \)       | 93.57    | 0.539      | 82     |
| \( S_0 \)    | RD        | \( L_2 \)      | 93.95    | 0.809      | 123    |
| \( S_0 \)    | RD        | \( KL \)       | 93.93    | 0.809      | 123    |
| \( S_0 \)    | DML       | \( L_2 \)      | 92.88    | 0.442      | 68     |
| \( S_0 \)    | RD        | \( L_2 \)      | 93.17    | 0.442      | 78     |

Fig. 4: This figure shows the results about \( \mathcal{L}_{energy} \). (a) indicates the relationship between the performance and FLOPs of \( S_1 \) with different \( \mathcal{L}_{energy} \) on CIFAR-100. (b) shows the relationship on ImageNet. For CIFAR-100, \( T \) is ResNet-110, \( S_0 \) is ResNet-20, and \( R_1 \) is ResNet-14. For ImageNet, \( T \) is ResNet-50, \( S_0 \) is ResNet-18, and \( R_1 \) is a NAS-based architecture. The results indicate that our trade-off is done by our sample-adaptive strategy, and most of \( R_1 \)'s FLOPs can be reduced with little performance decreasing.

1) Different Loss Functions: To show that our residual-guided learning is generic to different \( \mathcal{L}_{T-S} \), we carry on experiments using Kullback–Leibler divergence and \( L_2 \) distance loss function on CIFAR-10. As illustrated in Table VI, both Kullback–Leibler divergence and \( L_2 \) distance loss function work well with our ResKD framework.

2) Settings of \( \tau \) and \( t \): We have tried different \( \tau \) s (from 0.1 to 0.9) and \( t \) s (from 5 to 25) and choose the best one. It turns out that the performance of ResKD is not very sensitive to these hyper-parameter configurations as shown in Fig. 5. We set \( \tau = 0.9/0.5/0.5/0.1 \) for \( S_0 \) on CIFAR-10 / CIFAR-100 / Tiny-ImageNet / ImageNet, \( \tau = 0.1 \) for \( R_1 \) on all datasets and...
(a) $t$’s influence on $S_0$ and $S_1$. (b) $t$’s influence on $S_0$ and $S_1$.

Fig. 5: The relationship between hyper-parameters ($\tau$ and $t$) and the performance of $S_0$ and $S_1$ on CIFAR-10. $T$ is ResNet-110, $S_0$ is ResNet-20, and $R_1$ is ResNet-14. The result indicates that these hyper-parameters can partly affect the final performance but not large.

TABLE VII: Performance on CIFAR-10 / CIFAR-100. $R_1$ is a res-student whose architecture is searched by NAS. The results of $S_1$ are combined with sample-adaptive inference. $S_1$ (VGG) / $S_1$ (Xception) / $S_1$ (Mobile) / $S_1$ (Shuffle) means $S_0$ is VGG11BN / Xception / MobileNetv2 / ShuffleNetv2 and $R_1$ is a NAS-based network. **Bold:** The best results out of students. **Underline:** The second best results out of students.

| Architecture       | Teacher | Acc. (%) | MFLOPs |
|--------------------|---------|----------|--------|
| Inceptionv3        | -       | 95.28 / 79.40 | 3399 / 3400 |
| Xception           | -       | 94.58 / 78.13 | 1134 / 11134 |
| Xception           | Inceptionv3 | 94.99 / 78.58 | 1134 / 11134 |
| $S_1$ (Xception)   | Inceptionv3 | 95.15 / 83.13 | 1163 / 11934 |
| VGG11BN            | -       | 93.70 / 71.50 | 418 / 419 |
| VGG11BN            | -       | 91.99 / 68.96 | 172 / 173 |
| VGG11BN            | VGG11BN | 92.17 / 71.46 | 172 / 173 |
| $S_1$ (VGG)        | VGG11BN | 94.01 / 76.58 | 206 / 203 |
| ResNet50           | -       | 95.33 / 78.21 | 1305 / 1306 |
| MobileNetv2        | -       | 92.75 / 69.30 | 67 / 68 |
| MobileNetv2        | ResNet50 | 93.21 / 75.31 | 67 / 68 |
| $S_1$ (Mobile)     | ResNet50 | 93.83 / 78.30 | 70 / 101 |
| ShuffleNetv2       | -       | 93.40 / 71.52 | 45 / 46 |
| ShuffleNetv2       | ResNet50 | 93.98 / 75.92 | 45 / 46 |
| $S_1$ (Shuffle)    | ResNet50 | 94.56 / 78.57 | 49 / 65 |

$t = 20$ in all situations.

3) **Different Backbones:** To show that our framework is generic to different backbones, we also apply our residual-guided learning to other networks like VGG-11, MobileNetv2, ShuffleNetv2, and Xception. As shown in Table VII, our framework achieves consistent results with different backbones.

4) **Accuracy-Cost Trade-off:** We show the comparison of the accuracy-cost trade-off between ResKD and the baseline as illustrated in Fig. 6. With the same FLOPs, ResKD models achieve better accuracy, and when achieving the same accuracy, ResKD models demand less FLOPs.

5) **Different KD Methods:** To show that our ResKD framework is generic to other distillation methods (feature distillation and both prediction and feature distillation) and for a fair comparison, we use the new implementation [37]. Experimental results show that our ResKD models achieve consistent accuracy improvement for all the twelve KD methods as shown in Table VIII.

V. ANALYSIS: WHY RESIDUAL-GUIDED LEARNING WORKS

In this section, we try to shed some light on why and how our residual-guided learning helps the training process.

A. **Theoretical Analysis**

Our residual-guided learning is based on the gap between a teacher $T$ and a student $S$. We should use a suitable metric to measure the gap between $T$ and $S$. Many metrics are already proposed from simple $L_1$, $L_2$ to complex metrics in [61], [23], [62], [63]. Inspired by [23], we measure the informativeness of training examples by analyzing their resulting gradients since the training data contribute to optimization via gradients. The gap of informativeness ($\text{gap}_{\text{info}, GI})$ between $S$ and $T$ for a training example $x^{(j)}$ at an iteration $t$ is defined as:

$$\text{GI}(x^{(j)}, S, T, t) = \| \nabla_{\theta_t} \mathcal{L}(S(x^{(j)}), T(x^{(j)})) \|_2,$$

(12)

where $\nabla_{\theta_t}$ denotes the gradients of $S$’s parameters $\theta$ at the iteration $t$. However, computing this $L_2$-norm directly is expensive. Instead, we could estimate the upper bound $\hat{\text{GI}}$ for $\text{GI}$. 

![Accuracy-cost trade-off on CIFAR-100](image-url)
TABLE VIII: The results of different distillation methods on CIFAR-100. T is ResNet-110 with an accuracy of 74.31% and a cost of 255.28 MFLOPs. S₀ is ResNet-20. S₁ is a NAS-based network R₁ combined with S₀. We report the accuracies as “69.65 / 68.99 [37]”, which means that the re-implemented accuracy is 69.65, and the reported accuracy in [37] is 68.99. For a fair comparison, we control the cost of all S₁ at a similar level. Our ResKD can work well with different distillation methods.

| KD Method | Model | Accuracy (%) | MFLOPs |
|-----------|-------|--------------|--------|
| Feature Distillation | | | |
| FitNet [25] | S₀ | 69.65 / 68.99 [37] | 41.22 |
| | S₁ | 75.12 (+5.47 / +6.13) | 64.49 (+23.27) |
| AT [24] | S₀ | 70.50 / 71.22 [37] | 41.22 |
| | S₁ | 75.65 (+5.15 / +5.43) | 63.74 (+22.52) |
| SP [23] | S₀ | 70.72 / 70.04 [37] | 41.22 |
| | S₁ | 75.44 (+4.72 / +5.40) | 63.63 (+22.41) |
| CC [19] | S₀ | 69.44 / 69.48 [37] | 41.22 |
| | S₁ | 75.34 (+5.90 / +5.86) | 65.72 (+24.50) |
| VID [32] | S₀ | 70.22 / 70.16 [37] | 41.22 |
| | S₁ | 76.10 (+5.88 / +5.94) | 63.46 (+22.24) |
| RKD [30] | S₀ | 69.40 / 69.25 [37] | 41.22 |
| | S₁ | 75.52 (+6.12 / +6.27) | 64.97 (+23.75) |
| PKT [34] | S₀ | 70.38 / 70.25 [37] | 41.22 |
| | S₁ | 75.82 (+5.44 / +5.57) | 64.02 (+22.80) |
| AB [36] | S₀ | 69.94 / 69.53 [37] | 41.22 |
| | S₁ | 77.11 (+7.17 / +7.58) | 67.75 (+23.53) |
| FT [35] | S₀ | 70.06 / 70.22 [37] | 41.22 |
| | S₁ | 75.78 (+5.72 / +5.56) | 64.45 (+23.23) |
| NST [33] | S₀ | 70.24 / 69.53 [37] | 41.22 |
| | S₁ | 75.93 (+5.69 / +6.40) | 64.46 (+23.24) |
| CRD [37] | S₀ | 71.33 / 71.46 [37] | 41.22 |
| | S₁ | 76.07 (+4.74 / +4.61) | 62.80 (+21.58) |
| Both Prediction and Feature Distillation | | | |
| CRD+KD [37] | S₀ | 71.61 / 71.56 [37] | 41.22 |
| | S₁ | 75.75 (+4.14 / +4.19) | 57.37 (+16.15) |

Following the work of [63] and without loss of generality, we use a multi-layer perceptron (MLP) as the model in our analysis. Let \( \theta^{(l)} \in \mathbb{R}^{M_l \times M_{l+1}} \) be the weight matrix for layer \( l \) and \( \sigma^{(l)}(\cdot) \) be a Lipschitz continuous activation function, and then we have:

\[
\begin{align*}
a^{(0)} &= x^{(j)}, \\
h^{(l)} &= \theta^{(l)}a^{(l-1)}, \\
a^{(l)} &= \sigma^{(l)}(h^{(l)}), \\
f(x^{(j)}, \theta) &= a^{(L)},
\end{align*}
\]

where \( a^{(l)} \) denotes the feature maps after layer \( l \) and \( x^{(j)} \) is a certain sample. We define:

\[
\Sigma_l(h^{(l)}) = \text{diag}(\sigma^{(l)}(h_1^{(l)}), \sigma^{(l)}(h_2^{(l)}) \ldots, \sigma^{(l)}(h_{M_l}^{(l)})).
\]

\[
\Pi^{(l)} = (\prod_{i=1}^{l-1} \Sigma_i(h^{(i)}) \theta_i^t)(\sigma_{L}(h^{(l)})).
\]

\( L \) is the loss function \( \mathcal{L}(S(x^{(i)}), T(x^{(i)})) \) in Eq. (12). The GI\((l)\) is the informativeness of the parameters in layer \( l \) and it can be expressed as:

\[
\begin{align*}
\text{GI}^{(l)} &= ||(\Pi^{(l)} \nabla a^{(l)} \mathcal{L})(a^{(l-1)})^T||_2 \\
& \leq ||\Pi^{(l)}||_2 ||(a^{(l-1)})^T||_2 ||\nabla a^{(l)} \mathcal{L}||_2.
\end{align*}
\]

Various weight initialization [65] and activation normalization techniques [66, 67] uniformize the activations across samples. As a result, the variation of the gradient norm is mostly captured by the gradient of the loss function with respect to the pre-activation outputs of the last layer of our neural network. Consequently, we can derive the following upper bound to the gradient norm of all the parameters. Suppose that \( C_{max} \) is a constant:

\[
\text{GI} \leq C_{max} \|\nabla a^{(l)} \mathcal{L}\|_2.
\]

Based on Eq. (17), we set:

\[
\text{GI} = C_{max} \|\nabla a^{(l)} \mathcal{L}\|_2.
\]

For our method, we use \( L_2 \) distance loss function to measure the difference between current student network and the guidance of current teacher:

\[
\mathcal{L} = \|S - T\|_2^2.
\]

We calculate the first derivative of our loss function:

\[
\nabla a^{(l)} \mathcal{L} = 2 : \|S - T\|_2.
\]

According to Eq. (18) and Eq. (20), we set \( C_1 = 2C_{max} \), and we define \( S_0^t / R_1^t \) is the result that the \( S_0 / R_1 \) has been optimized well:

\[
\text{GI}_{S_0} = C_1 \|S_0 - T\|_2,
\]

\[
\text{GI}_{S_1} = C_1 \|R_1 - (T - S_0^t)\|_2,
\]

\[
\text{GI}_{S_1} = C_1 \|R_2 - (T - S_0^t - R_1^t)\|_2,
\]

\[
\ldots
\]

With proper optimization for \( R_1 \), we have the best \( R_1^t \). The best \( R_1^t \) is better than other value of \( R_1 \) include that \( R_1 \) always

TABLE IX: GI of different student networks to the teacher network on CIFAR-10. T is ResNet-110, and S₀ is ResNet-20. ResXY-Z means that we use ResNet-X as S₀, ResNet-Y as R₁, and ResNet-Z as R₂. Bold: The best results out of students.

| Architecture | Acc. (%) | L₂ to T | GI/C₁ to T |
|--------------|----------|---------|------------|
| Teacher      |          |         |            |
| Res110       | 94.19    | -       | -          |
| Student      |          |         |            |
| Res20        | 93.06    | 0.90    | 1.73       |
| ResKD networks |        |         |            |
| Res20-8      | 93.03    | 0.82    | 1.64       |
| Res20-14     | 93.27    | 0.74    | 1.53       |
| Res20-20     | 93.35    | 0.61    | 1.37       |
| Res20-20-8   | 93.55    | 0.49    | 1.15       |
| Res20-20-14  | 93.84    | 0.43    | 1.03       |
| Res20-20-20  | 93.95    | 0.37    | 0.89       |
equals 0, so we have the upper bound of $\tilde{\text{GI}}_{S_1}$ in residual-guided knowledge distillation.

$$\tilde{\text{GI}}_{S_0} \leq C_1 \| 0 - (T - S_0^r) \|_2 \leq \tilde{\text{GI}}_{S_0}.$$  \quad (22)

We rewrite Eq. (22) and get similar conclusion of other equations, and set $\Delta_i^{(\text{GI})} \geq 0$:

$$\tilde{\text{GI}}_{S_0} = \tilde{\text{GI}}_{S_1} + \Delta_0^{(\text{GI})},$$

$$\tilde{\text{GI}}_{S_1} = \tilde{\text{GI}}_{S_2} + \Delta_1^{(\text{GI})},$$

$$\cdots,$n

$$\tilde{\text{GI}}_{S_{n-1}} = \tilde{\text{GI}}_{S_n} + \Delta_{n-1}^{(\text{GI})}.$$  \quad (23)

We add all the equations in Eq. (23):

$$\tilde{\text{GI}}_{S_0} = \tilde{\text{GI}}_{S_n} + \sum_{i=0}^{n-1} \Delta_i^{(\text{GI})}.$$  \quad (24)

$R^*_i$ is the best result of optimizing $R_i$ to approach $T - S_0^r - \sum_{j=1}^{i-1} R_j$, so we suppose that when we have optimized $R^*_i$, $\tilde{\text{GI}}_{S_i}$ has changed to $k_i(T - S_0^r - \sum_{j=1}^{i-1} R_j)$ and $k_i \in (0, 1)$. In this constraint, $\tilde{\text{GI}}_{S_i}$ can be rewritten as:

$$\Delta_i^{(\text{GI})} = k_i \cdot \tilde{\text{GI}}_{S_{i-1}}, k_i \in (0, 1).$$  \quad (25)

When the $R_i$ becomes stronger, the $k_i$ becomes larger and the network $R_i$ bridge the gap better. Also the $\tilde{\text{GI}}_{S_i}$ can be rewritten as:

$$\tilde{\text{GI}}_{S_i} = \frac{1}{\prod_{j=1}^{i} (1 - k_j)}(T - S_0^r),$$

$$k_j \in (0, 1).$$  \quad (26)

We can learn that the final performance of $S_i$ is depended on the expression ability of each network in $S_0, R_1, \cdots, R_n$. In Section IV-B1 we will show some results that how the choice of a certain res-student network and the number of res-student networks affect the final performance.

Fig. 7: The visualization of 2D PCA representation of ResKD student’s logits using different number of res-students on CIFAR-10. The red dots in (a), (b) and (c) are the distribution of $T$, ResNet-110, and they are the same one. (a) the distribution of $S_0$, (b) the distribution of $S_1$, (c) the distribution of $S_2$.

Fig. 8: The visualization of 2D PCA representation of $S_1$’s / $S_2$’s logits using different $R_1$ / $R_2$ on CIFAR-10. The red dots are the distribution of $T$, ResNet-110, and they are the same one in (a), (b), (c), (d), (e), and (f). The blue dots in (a), (b), and (c) are the distributions of different $S_1$ using different $R_1$. $S_0$ is ResNet-20. The blue dots in (d), (e), and (f) are the distributions of $S_2$ using different $R_2$ and using ResNet-20 as both $S_0$ and $R_1$.

B. Empirical Analysis

In this part, we show the GI of different students to validate our theoretical analysis on CIFAR-10 first. Next, we show the distribution of logits of different students directly to validate our residual-guided knowledge distillation.

1) Gap info (GI) Observation: We empirically verify whether the expression ability of the student network and res-student networks will influence the GI. The average GI and the average logsit distribution samples that anyone of our residual-guided model predicts correctly and $S_0$ gives the wrong prediction are showed. We use ResNet-110 as the teacher network and ResNet-20 as the $S_0$ on CIFAR-10. When we focus on $R_2$, we set ResNet-20 as $R_1$ and observe how different res-student network $R_2$ affects the final performance. The expression of ResX-Y-Z means that we use ResNet-X as $S_0$, use ResNet-Y as $R_1$ and use ResNet-Z as $R_2$.

Firstly, we discuss the influence when the number of res-student networks used increases. The results are shown in Table IX. When we use the first res-student network $R_1$ to bridge the gap between $T$ and $S_0$, the output of $S_0 + R_1$ is more similar to the output of $T$ (the average second norm to $T$ is less) and the GI is smaller than the one of $S_0$. Similarly, when we use $R_2$, the output of $S_0 + R_1 + R_2$ is more likely to $T$ and the GI continues decreasing. When we use ResNet-20 for $S_0$, $R_1$ and $R_2$, the GI is about the half of $S_0$.

Next, we discuss the difference among different latest res-student networks. In Table IX the different number of res-student networks can cause different results. When we use ResNet-8 / 14 / 20 to for $R_1$, the “GI/C_1 to T” and “L_2 to T” decrease, and when we use ResNet-20, the value is the
smallest. The phenomenon is similar when we use ResNet-8 / 14 / 20 to for R2 and fix S0 and R1. We can learn that the stronger R1/R2 is, the more similar S1/S2 is with the T.

g) Logits Visualization: We show the 2D PCA representation of different S’s logits on CIFAR-10. In each figure, the red points are the same one and indicate the distribution of T’s logits.

In Fig.7 the blue points indicate the distributions of S0’s, S1’s and S2’s logits. We use ResNet-20 for S0, R1 and R2 in this figure. We can learn that with the number of res-students increasing, the distribution of S1’s logits is more and more similar to the distribution of T’s.

In Fig.8 the blue points indicate the distribution of S1 / S2. We use ResNet-20 / 14 / 8 for R1 / R2, S0 is ResNet-20, and R1 is ResNet-20 when we focus R2. We can learn that when res-students R1 / R2 is stronger, the distribution of S1 / S2 is more similar with the distribution of T.

VI. CONCLUSION

We have studied an under-explored yet important field in knowledge distillation of neural networks. We have shown that using res-students to bridge the gap between student and teacher is a key to improve the quality of knowledge distillation. We propose our residual-guided learning and sample-adaptive inference to realize this idea. We also validate the effectiveness of our approach in various datasets and studied its properties both empirically and theoretically.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. CVPR, 2016, pp. 770–778.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. CVPR, 2017, pp. 4700–4708.

[3] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” arXiv preprint arXiv:1905.11946, 2019.

[4] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, “Fixing the train-test resolution discrepancy: FixEfficientNet,” arXiv preprint arXiv:2003.08237, 2020.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in Proc. CVPR, 2018, pp. 4510–4520.

[7] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in Proc. CVPR, 2018, pp. 6848–6856.

[8] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in Proc. ECCV, 2018, pp. 116–131.

[9] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in Proc. CVPR, 2018, pp. 4320–4328.

[10] T. S. Nowak and J. J. Corso, “Deep net triage: Analyzing the importance of network layers via structural compression,” arXiv preprint arXiv:1801.04651, 2018.

[11] E. J. Crowley, G. Gray, and A. J. Storkey, “Moonshine: Distilling with cheap convolutions,” in Proc. NeurIPS, 2018, pp. 2888–2898.

[12] H. Wang, H. Zhao, X. Li, and X. Tan, “Progressive blockwise knowledge distillation for neural network acceleration,” in Proc. of IJCAI, 2018, pp. 2769–2775.

[13] J.-L. Liu, J. Peng, and A. G. Schwing, “Knowledge flow: Improve upon your teachers,” arXiv preprint arXiv:1904.05878, 2019.

[14] J. Gu and V. Tresp, “Search for better students to learn distilled knowledge,” arXiv preprint arXiv:2001.11612, 2020.
