Role-Wise Data Augmentation for Knowledge Distillation

Jie Fu[1]*, Xue Geng[2]*, Zhijian Duan[3], Bohan Zhuang[4], Xingdi Yuan[5], Adam Trischler[5], Jie Lin[2], Chris Pal[1], and Hao Dong[3]

[1] Mila, Polytechnique Montréal, jie.fu@polymtl.ca
[2] I^{2}R, A*STAR
[3] Peking University
[4] Monash University
[5] Microsoft Research Montréal

Abstract. Knowledge Distillation (KD) is a common method for transferring the "knowledge" learned by one machine learning model (the *teacher*) into another model (the *student*), where typically, the teacher has a greater capacity (e.g., more parameters or higher bit-widths). To our knowledge, existing methods overlook the fact that although the student absorbs extra knowledge from the teacher, both models share the same input data – and this data is the only medium by which the teacher's knowledge can be demonstrated. Due to the difference in model capacities, the student may not benefit fully from the same data points on which the teacher is trained. On the other hand, a human teacher may demonstrate a piece of knowledge with individualized examples adapted to a particular student, for instance, in terms of her cultural background and interests. Inspired by this behavior, we design data augmentation agents with distinct roles to facilitate knowledge distillation. Our data augmentation agents generate distinct training data for the teacher and student, respectively. We find empirically that specially tailored data points enable the teacher's knowledge to be demonstrated more effectively to the student. We compare our approach with existing KD methods on training popular neural architectures and demonstrate that role-wise data augmentation improves the effectiveness of KD over strong prior approaches. The code for reproducing our results can be found at https://github.com/bigaidream-projects/role-kd

Keywords: Data Augmentation, Knowledge Distillation, Quantization

1 Introduction

In the educational psychology literature, it is generally considered beneficial if teachers can adapt curricula based upon students' prior experiences [2,4,12,32]. These vary widely depending on students' cultural backgrounds, previous educational experiences, interests, and motivations.

* Equal contribution.

Knowledge distillation (KD) [5,17] is a common framework for training machine learning models. It works by transferring knowledge from a higher-capacity teacher model to a lower-capacity student model. Most KD methods can be categorized by how they define the knowledge stored in the teacher (i.e., the "soft targets" of training as defined in existing literature). For instance, [17] originally proposed KD for neural networks, and they define the output class probabilities (i.e., soft labels) generated by the teacher as the targets for assisting the training of students. In a follow up work, [30] defined the soft targets via the feature maps in the teacher model's hidden layers.

To train a student network with KD effectively, it is important to distill as much knowledge from the teacher as possible. However, previous methods overlook the importance of the *medium* by which the teacher's knowledge is demonstrated: the training data points. We conjecture that there exist examples, not necessarily seen and ingested by the teacher, that might make it easier for the student to absorb the teacher's knowledge. Blindly adding more training examples may not be beneficial because it may slow down training and introduce unnecessary biases [18]. The analogy with how human teachers adjust their teaching to their students' particular situations (e.g., with the feedback gathered from the students during teaching) suggests that a reasonable yet uninvestigated approach might be to augment the training data for both the teacher and student according to *distinct* policies.

In this paper, we study whether and how adaptive data augmentation and knowledge distillation can be leveraged synergistically to better train student networks. We propose a two-stage, role-wise data augmentation process for KD. This process consists of: (1) training a teacher network till convergence while learning a schedule of policies to augment the training data specifically for the teacher; (2) distilling the knowledge from the teacher into a student network while learning another schedule of policies to augment the training data specifically for the student. It is worth noting that this two-stage framework is orthogonal to existing methods for KD, which focus on how the knowledge to be distilled is defined; thus, our approach can be combined with previous methods straightforwardly.

Although our proposed method can in principle be applied to any models trained via KD, we focus specifically on how to use it to transfer the knowledge from a full-precision teacher network into a student network with lower bit-width. Network quantization is crucial when deploying trained models on embedded devices, or in data centers to reduce energy consumption [33]. KD-based quantization [41,28] jointly trains a full-precision model, which acts as the teacher, alongside a low-precision model, which acts as the student. Previous work has shown that distilling a full-precision teacher's knowledge into a low-precision student, followed by fine-tuning, incurs noticeable performance degradation, especially when the bit-widths are below four [41,28]. We show that it is advantageous to use adaptive data augmentation to generate more training data for the low-precision network based on its specific weaknesses. For example, low-precision networks may have difficulties learning rotation-related patterns,[6] and the data augmentation agent should be aware of this and generate more such data points.

2 Related Work

Knowledge distillation. KD is initially proposed for model compression, where a powerful wide/deep teacher distills knowledge to a narrow/shallow student to improve her performance [17,30]. Most KD methods mainly differ in how they define the knowledge learned by the teacher. For example, [17] define the class probabilities (i.e., soft labels) generated by the teacher as the knowledge, and [30] treat the teacher's feature maps as the knowledge to be transferred. Moreover, some literature [39,27,31,19] designs advanced distillation strategies in order to let the student learn better from the teacher's rich knowledge. Due to the effectiveness of KD, it has also been widely used in many computer vision tasks. For example, Zhang *et al.* [37] propose to transfer the knowledge learned with optical flow CNN to improve the action recognition performance. And several works propose to learn efficient object detection [7,35] and semantic segmentation [16] with distillation. In terms of the definition of knowledge to be distilled from the teacher, existing models typically use teacher's class probabilities [17] and/or intermediate features [30,27,31,19]. Among those KD methods that utilize intermediate feature maps, Relational KD (RKD) considers [27] the intra-relationship in the same feature map, while Multi-Head KD (MHKD)[31] and KD using SVD (KD-SVD) [19] utilize the inter-relationship across feature maps. Compared to these relationship-based KD methods, we incorporate both the intra- and inter-relationships within and across feature maps, which is an additional engineering trick used in our paper. More importantly, all the previous KD methods ignore the dual-role of the training data for the teacher and the student. In this paper, we propose to use different training data for the teacher and the student, where the training data is augmented by distinct learned policies.

Automated data augmentation. Manually applying data augmentation rules such as random rotating, flipping, and scaling are common practices for training neural models on image classification tasks [22,14]. Several recent works attempt to automate the data augmentation process. Generative adversarial networks [29] and Bayesian optimization [34] have been used for this process. [10] augment training data in the learned feature space by injecting noise and interpolation. [23] learn how to combine pairs of images for data augmentation. AutoAugment [9] searches for the optimal data augmentation policies (e.g., how to rotate) based on reinforcement learning. However, the search process is computationally expensive. Population-based augmentation (PBA) [18] uses an evolution-based algorithm to automatically augment data in an efficient way. In contrast to previous approaches, we study the effect of the training data for different roles

[6] We will visualize the learned schedules of policies in Section 5.5.

in KD (i.e., the teacher and the student) and propose to use automatic data augmentation to train the student better from her teacher.

Network quantization. Quantization is an effective approach for compressing models by using low bitwidth weights and activations. It can be categorized into fixed-point quantization and binary neural networks. Uniform approaches [40,41] perform fixed-point quantization with a constant quantization step. To reduce the quantization error, non-uniform strategies [38,20,6] propose to jointly learn the quantizer and model parameters for better accuracy. Moreover, to relax the non-differentiable quantizer, which is core issue of quantization, some works propose to make the gradient-based optimization feasible by using gumble softmax [24] or learning with regularization [1]. KD methods have shown to be effective at improving the performance of lower-capacity networks using the knowledge from higher-capacity networks [26,41]. However, the KD methods used in [26,41] are generic and not tailored to the quantization problems. In this paper, our proposed KD method takes into account the fact that the student may prefer different training data from the teacher.

3 Preliminaries

3.1 Population-Based Augmentation (PBA)

Population Based Augmentation (PBA) [18] is an algorithm that quickly and efficiently learns data augmentation functions for neural network training. Instead of generating a fixed augmentation policy, as an evolutionary search algorithm, PBA learns a dynamic *per-epoch* schedule of augmentation policies, denoted as $\mathcal{A}$. Since this schedule is epoch-based, it will *re-create* the augmented dataset every epoch. PBA begins with a population of models that are trained in parallel on a small subset of the original training data. The weights of the worse performing models in the population are replaced by those from better performing models (i.e., exploitation), and the augmentation policies $\mathcal{A}$ are changed to new ones from the pre-defined policy search space (i.e., exploration). More concretely, $\mathcal{A}$ consists of a series of vectors of $(operator, probability, magnitude)$, and PBA learns a schedule to get a new datapoint $\tilde{x_i} = operator(x_i, magnitude)$ with an adaptive *probability*. Following [18], we use 15 common operators such as random cropping, flipping, scaling, rotating, and translating. Note that when $probability = 0$, it is equivalent to a null operator.

After training, PBA usually keeps the learned augmentation schedule of policies but *discards* the elementary parameters of the models. The discovered schedules of the small subset can be used directly on the original training data. Even more, a different model (e.g., larger one) can also use the learned schedule to improve their training on the same task.

3.2 Knowledge Distillation (KD)

Following the notations in [27], a KD method aims to minimize the objective function:

$$\mathcal{L}_{\text{general}} = \mathcal{L}_{\text{task}} + \lambda \cdot \mathcal{L}_{\text{KD}}, \tag{1}$$

where λ is a hyper-parameter to balance the impact of the KD loss term.

In this paper for classification tasks,

$$\mathcal{L}_{\text{task}} = \sum_{x_i \in \mathcal{X}} \mathcal{H}(\text{softmax}(\mathcal{F}_S^{\text{final}}(x_i)), y_{\text{truth}}), \tag{2}$$

where $\mathcal{X}$ refers to training sample space, $y_{\text{truth}} \in \mathcal{Y}$ are the ground-truth labels, $\mathcal{F}_S(\cdot)$ is the student network, and $\mathcal{H}(\cdot)$ denotes the cross-entropy.

The KD term can be defined as:

$$\mathcal{L}_{\text{KD}} = \sum_{x_i \in \mathcal{X}} l(\mathcal{F}_T(x_i), \mathcal{F}_S(x_i)), \tag{3}$$

where $\mathcal{F}(\cdot)$ is the function of the network. And $l(\cdot)$ is a loss function to compute the difference between the teacher network and the student network.

For KD methods [17] that use soft labels, the objective can be defined as:

$$\mathcal{L}_{\text{KD}}^{\text{soft}} = \sum_{x_i \in \mathcal{X}} \mathcal{H}(\text{softmax}(\mathcal{F}_T^{\text{final}}(x_i)), \text{softmax}(\mathcal{F}_S^{\text{final}}(x_i))), \tag{4}$$

where $\mathcal{F}^{\text{final}}(x_i)$ is the feature map of the final layer.

We notice that there exists some KD methods utilizing the intermediate feature maps in complementary ways. For example, Relational KD [27] considers the intra-relationships. That is, given the feature map of layer j, the KD loss can be formulated as:

$$\mathcal{L}_{\text{KD}}^{\text{intra}} = \sum_{x_i \in \mathcal{X}} l(\Phi(\mathcal{F}_T^j(x_i)), \Phi(\mathcal{F}_S^j(x_i))), \tag{5}$$

where $\Phi(\cdot)$ refers to the potential function measuring the pairwise relationship inside a feature map from student network or teacher network and $\mathcal{F}^j(x_i)$ is the feature map of layer j, including the final logits layer. Therefore, this feature-based KD method includes the benefits of using soft labels.

On the other hand, some works [31,19] consider the inter-relationships, where the KD term can be formulated as:

$$\mathcal{L}_{\text{KD}}^{\text{inter}} = \sum_{x_i \in \mathcal{X}} l(\varphi(\mathcal{F}_T^j(x_i), \mathcal{F}_T^k(x_i)), \varphi(\mathcal{F}_S^j(x_i), \mathcal{F}_S^k(x_i))), \tag{6}$$

where $\varphi(\cdot)$ measures the inter-relationship between feature maps of different layers, i.e. $k \neq j$.

3.3 Quantization

In this work, we use DoReFa[7] [40] to quantize both weights and activations. The quantization function $Q(\cdot)$ is defined as:

$$r_q = Q(r) = \frac{1}{2^{n_{\text{bits}}} - 1} \cdot \text{round}((2^{n_{\text{bits}}} - 1) \cdot r), \tag{7}$$

where r is the full-precision value, r_q indicates the quantized value, n_{bits} refers to the number of bits to represent this value. With this quantization function, the quantization on weights w is defined as:

$$w_q = 2 \cdot Q(\frac{\tanh(r)}{2 \cdot \max(|\tanh(w)|)} + \frac{1}{2}) - 1. \tag{8}$$

The back-propagation is approximated by the straight-through estimator [3], the partial gradient $\frac{\partial l}{\partial r}$ w.r.t. the loss l is computed as:

$$\frac{\partial l}{\partial r} = \frac{\partial l}{\partial r_q} \cdot \frac{\partial r_q}{\partial r} \approx \frac{\partial l}{\partial r_q}. \tag{9}$$

4 The Proposed Method

Our proposed method has two stages which will be described in the following subsections. In the first stage (Stage-α), we train a teacher network, denoted as $\mathcal{N}_T$, with the help of PBA-based augmentation. In the second stage (Stage-β), we further distill the knowledge from $\mathcal{N}_T$ (pre-trained in the first stage) to the student network, denoted as $\mathcal{N}_S$, while learning another augmentation schedule to augment the training data for $\mathcal{N}_S$.

4.1 Stage-α

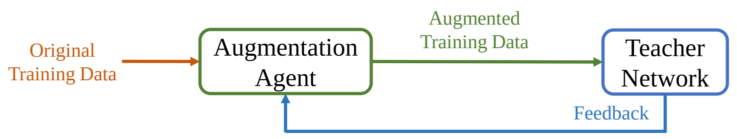


Fig. 1: Diagram (stage-α) of training a data augmentation agent for the $\mathcal{N}_T$.

In general, a teacher can provide better training signals for the student if the teacher's performance increases [25]. As shown in Fig. 1, we apply PBA to learn an dynamic per-epoch schedule of augmentation policies, $\mathcal{A}_T$, for $\mathcal{N}_T$ on a

[7] It should be noted that our proposed method is orthogonal to any particular quantization method.

small *subset* of training data. That is, the augmentation agent's training signal is defined as the feedback of $\mathcal{N}_T$'s accuracy on a subset of the dataset. After this, we use the discovered schedule $\mathcal{A}_T$ to augment the whole training dataset and re-train $\mathcal{N}_T$ on it till convergence.

Stage-α can be seen as a preparation stage which is used to improve the teacher's performance and does not interfere with the augmentation policy learning for the student. This can been also shown in Fig. 1, where the feedback is only from $\mathcal{N}_T$.

4.2 Stage-β

In this stage, we distill the knowledge from $\mathcal{N}_T$ (pre-trained in Stage-α) to the student network, denoted as $\mathcal{N}_S$, while learning another augmentation schedule to augment the training data for $\mathcal{N}_S$. In order to take advantage of this functionality, we apply the KD methods together with data augmentation in stage-β as shown in Fig. 2.

More concretely, we first use PBA to learn an epoch-based augmentation schedule $\mathcal{A}_S$ for $\mathcal{N}_S$ on a subset of the dataset. Different from the schedule $\mathcal{A}_T$ learned in stage-α, $\mathcal{A}_S$ is learned based on the feedback (i.e., accuracy) from $\mathcal{N}_S$, who is trained with KD. In other words, $\mathcal{N}_S$ receives additional training signals from $\mathcal{N}_T$ that is pre-trained in stage-α. The augmentation policy $\mathcal{A}_S$ is learned to facilitate this KD process, and thus would be different from $\mathcal{A}_T$ that is used by the teacher $\mathcal{N}_T$.

It should be noted that the above process is only used to learn the augmentation policy $\mathcal{A}_S$ for the student on a subset of the whole training data. After this, we use the learned $\mathcal{A}_S$ to augment the whole training dataset, and *re-train* $\mathcal{N}_S$ on it with the distilled knowledge from $\mathcal{N}_T$. Note that, because the learned schedule is epoch-based, we do not use the discovered schedule $\mathcal{A}_T$ from stage-α to augment the training data as initialization.

When $\mathcal{N}_S$ is a low-precision network, following [11], we share the same network architecture[8] between $\mathcal{N}_T$ and $\mathcal{N}_S$. When $\mathcal{N}_S$ is a full-precision network, she will have fewer layers compared to $\mathcal{N}_T$.

5 Experiments

5.1 Settings

We evaluate our approach on two benchmark datasets: CIFAR-10 [21] and CIFAR-100. We search over a "reduced" CIFAR-10/CIFAR-100 with 4,000 training images and 36,000 validation images, which is the same as in [18]. All the data augmentation models are run with 16 total trials to generate augmentation schedules. Following PBA, in stage-α, we run PBA to create schedules over separate models and then transfer the CIFAR-10 policy to CIFAR-100. However, for student network training in stage-β, we empirically use the respective "reduced"

[8] Note this is not a hard constraint, we choose such strategy to reduce the number of factors that might influence the final performance.

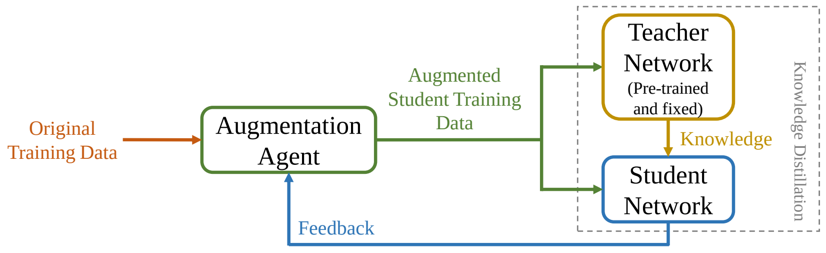


Fig. 2: Diagram (stage-β) to augment training datapoints for both $\mathcal{N}_T$ and $\mathcal{N}_S$. The $\mathcal{N}_T$ has been pre-trained using the method shown in Section 4.1, and is fixed during training. The augmentation agent in stage-β is designed to learn schedules of polices that are different from those learned in stage-α, and thus the agent only receives the feedback from $\mathcal{N}_S$.

dataset. The data augmentation approaches for the baselines include random crop and horizontal flipping operations. Following [18], our policy search space has a total of 15 operations, each having two magnitude and discrete probability values. We use discrete probability values from 0% to 100%, in increments of 10%. Magnitudes range from 0 to 9.

For the verification on quantization, the models we evaluate on include AlexNet [22] and ResNet18 [14]. For the verification on full-precision networks in Section 5.6, the networks we evaluate are Wide Residual Network [36], PyramidNet [13] and PreResNet [15].

The number of epochs is 200 and the batch size is 128. For full-precision network $\mathcal{N}_T$, the learning rate starts from 0.1 and is decayed by 0.1 after every 30% of the total epochs. we use SGD with a Nesterov momentum optimizer. The weight decay is set to 5×10^{-4}. For quantization, the learning rate is set to 10^{-3} and is divided by 10 every 30% of the total epochs. We use the pre-trained teacher network model as the initial point of student network. We use a smaller weight decay 10^{-5} assuming that less regularization is needed by the lower-precision networks. Following DoReFa [40], the first layer and last layer are not quantized.

Following [8], during training, we gradually transition the student from learning based on the teacher to training based on the ground-truth labels. This heuristic provides the student with more rich training signals in the early stage but does not force the student to strictly mimic the teacher's behaviors. As for the implementation, we decay the balancing hyper-parameter λ in the KD loss by 0.5 every 60 epochs.

5.2 Comparing Different KD Methods

As mentioned in Section 3.2, there exist complementary KD methods considering both intra- and inter-relationships within and across feature maps. A natural

Table 1: Accuracy comparison among various KD methods for CIFAR-100 using ResNet18 with 2-bit/4-bit settings. We compare ours with the following methods: Soft labels [17], DML [39], RKD [27], MHGD [31] and KD-SVD [19].

Bit-Width (Weight / Activation)	Soft labels	DML	RKD	MHGD	KD-SVD	II-KD
4/4	70.48	72.47	71.84	73.52	73.92	74.21
2/2	70.09	69.72	70.71	71.80	72.97	73.35

question is if it would be beneficial to combine them to further boost the performance together with data augmentation. Therefore, we propose a simple extension to these complementary KD methods, dubbed as II-KD, by incorporating intra-relationships inside the feature map and inter-relationships across different feature maps. We incorporate the two relationships into the final objective function as follows:

$$\mathcal{L}_{\text{KD}}^{\text{II}} = \mathcal{L}_{\text{task}} + \lambda \cdot (\mathcal{L}_{\text{KD}}^{\text{intra}} + \mathcal{L}_{\text{KD}}^{\text{inter}}), \tag{10}$$

where we only use a single balancing hyper-parameter λ between the original loss and the distillation loss, which does not introduce extra hyper-parameters.

More precisely, our KD method incorporates components of three conventional KD methods: RKD [27], MHGD [31] and KD-SVD [19]. As shown in Eq. (10), we add the three KD terms together with equal coefficients. We use the loss function $l(\cdot)$ following their approaches. For the back-propagation, we clip the gradient for KD loss as in KD-SVD, because this will smoothly post-processes the gradient to limit the impact of KD loss in training. For AlexNet we select the feature maps of ReLU layers after the convolution/max pooling layer. For ResNet18, we select the feature maps of the last ReLU layer of each residual block.

We evaluate our proposed KD extension on CIFAR-100 with ResNet18 for different bit-width settings by comparing with various KD methods. For the baseline methods, we use their default settings with a fixed and pre-trained teacher network in the training stage and $\lambda = 1$ for the knowledge distillation loss. We set $\lambda = 0.4$ for II-KD in Eq. (10), as we have two KD terms. Tab. 1 reports the results on various augmented KD methods. We observe that our proposed methods clearly outperform the other KD methods on all the settings, though the improvements over MHGD and KD-SVD are not huge. The results also reveal that only relying soft labels is not as effective as utilizing multiple supervising signals from the teacher. It should be noted that we do not thoroughly tune the coefficients within II-KD, as we want to minimize the number of hyperparameters and we find that it is effective with the most simple setting.

5.3 Is Role-Wise Augmentation with KD Effective for Quantization?

In this subsection, we aim to answer this question: is our two-stage role-wise augmentation with KD effective for network quantization? We conduct experiments on CIFAR-10 and CIFAR-100 datasets under full-precision, 4-bit, and 2-bit settings.

Table 2: Accuracy on CIFAR-10 and CIFAR-100 datasets with different bit-widths. **Vanilla** for 4-bit and 2-bit refers to training a network based on DoReFa [40] from scratch without learned data augmentation. **Stage-α** refers to using learned schedules discovered by PBA to *re-train* $\mathcal{N}_T$ as described in Section 4.1. **only II-KD** refers to training $\mathcal{N}_S$ using II-KD but without the learned data augmentation. **Stage-β** refers to training $\mathcal{N}_S$ using II-KD and the learned data augmentation. For **Vanilla** and **Stage-α**, we report the accuracy of $\mathcal{N}_T$, and for the rest we report the accuracy of $\mathcal{N}_S$.

Methods		AlexNet CIFAR-10	AlexNet CIFAR-100	ResNet18 CIFAR-10	ResNet18 CIFAR-100
Vanilla	32-bit	90.58	65.80	93.57	74.85
	4-bit	89.72	60.25	90.97	69.81
	2-bit	88.77	58.96	90.00	67.06
Stage-α	32-bit	91.62	66.40	94.49	75.19
	4-bit	90.06	60.65	91.47	70.24
	2-bit	89.28	58.59	89.99	67.32
Only II-KD	4-bit	90.55	65.55	91.42	73.85
	2-bit	89.18	63.49	90.60	72.44
Stage-β	4-bit	92.00	65.69	94.44	74.21
	2-bit	90.63	64.06	93.20	73.35

From Tab. 2, we can observe that training with learned data augmentation schedules does not improve the performance of low-precision networks too much. Similar to the results obtained in [41], transferring knowledge from the full-precision to the low-precision student usually helps the training of students, which is especially obvious on the CIFAR-100 dataset. Tab. 2 also clearly shows that our proposed pipeline consistently improves the performance of the low-precision student networks. For example, the 4-bit $\mathcal{N}_S$ is comparable with full-precision reference without loss of accuracy for CIFAR-10 and with loss of accuracy within 1.0% on CIFAR-100. When decreasing the numerical precision to 2-bit, the results are still promising as compared with other baselines, even though there is a performance gap between the 2-bit and the full-precision models. For instance, our approach usually outperforms the strong baseline, only using II-KD, by more than 1.0%. In particular, we observe that when the numerical precision is lower, the augmentation probably help more in improving the performance of KD methods as comparing the performance of II_KD and Stage-β. For instance, our proposed methods has 0.91% gain in 2-bit but only has 0.36% gain in 4-bit with ResNet18 network for CIFAR-100.

5.4 Comparing Schedules

Here we aim to answer this question: how effective is it if we use $\mathcal{A}_T$, learned based on the feedback from $\mathcal{N}_T$ in stage-α, to dynamically augment the training dataset and train $\mathcal{N}_S$ on it. Tab. 3 reports the accuracy comparison with different KD methods and augmentation schedules. We can clearly see that augmenting the training dataset for $\mathcal{N}_S$ with $\mathcal{A}_S$ consistently outperforms those using the

Table 3: Accuracy comparison on learned schedules from teacher and student separately for CIFAR-100 with 4-bit networks using different KD methods.

	AlexNet			ResNet18		
Methods	DML	MHGD	II-KD	DML	MHGD	II-KD
Schedules based on teacher	61.61	61.62	64.97	71.78	69.76	73.46
Schedules based on student	63.73	63.47	65.69	72.47	73.52	74.21

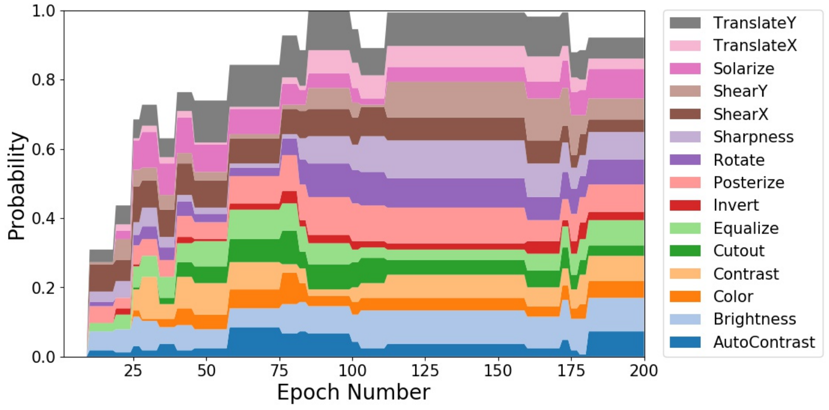
transferred schedules $\mathcal{A}_T$ among different KD methods. This observation is consistent with our assumption that $\mathcal{N}_S$ has her own optimal augmentation schedule, $\mathcal{A}_S$, that is different from $\mathcal{A}_T$ for $\mathcal{N}_T$. In particular, blindly applying the teacher augmentation schedule $\mathcal{A}_T$ may negatively influence the training of $\mathcal{N}_S$ as compared to only using KD. For example, the learned schedule based on the teacher $\mathcal{A}_T$ degrades the performance of $\mathcal{N}_S$ by 0.58% for AlexNet on CIFAR-100 as compared to applying KD methods, as shown in Tab. 2. Furthermore, we observe that II-KD outperforms other KD methods significantly on CIFAR-100. This observation probably proves that the combination of intra- and inter- KD methods helps to boost the classification performance.

5.5 Analyzing the Learned Schedules

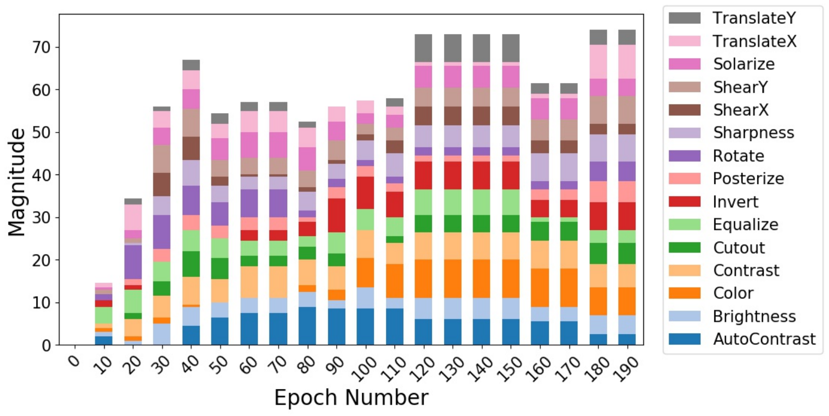
To analyze the difference on the discovered schedules between $\mathcal{N}_T$ (i.e., full-precision ResNet18) and $\mathcal{N}_S$ (i.e., 4-bit ResNet18), we report their augmented schedules quantitatively in terms of normalized probability and magnitude on CIFAR-100. Fig. 3 shows the augmented schedules for teacher while Fig. 4 shows the student. We normalize the probability of each epoch by dividing the maximal summation of probabilities for all operations across all epochs.

It can be seen that the discovered schedules $\mathcal{A}_S$ for $\mathcal{N}_S$ is quite different from $\mathcal{A}_T$ for $\mathcal{N}_T$. In particular, for $\mathcal{A}_T$, there is an emphasis on Brightness, Posterize, Rotate, Sharpness and TranslateY, while $\mathcal{A}_S$ cares more about Contrast, ShearX and TranslateY. Furthermore, we observe that the probability and magnitude increase as the epoch evolves. For $\mathcal{A}_S$, in the beginning, KD plays a more important role, and there is no augmentation operation before about epoch 50. As the training continues, the augmentation policies become more important. One possible reason is that, for low-precision networks, KD methods can provide rich training signals such that data augmentation does not help in the early training phases.

Furthermore, we observe that, compared to $\mathcal{A}_T$, the schedules for student $\mathcal{A}_S$ evolves more smoothly in the sense that the policy updating frequency is lower. For example, the probability and magnitude values change about every 40 epochs for student, while the policies for teacher update about every 15 epochs. One possible reason might be that for the low-precision $\mathcal{N}_S$, KD methods make the training process more smooth and it is not necessary to change the augmentation policies too frequently. This is consistent with the observations shown in Tab. 2 that KD can already provide useful training signals. It should be also noted that the teacher policies $\mathcal{A}_T$ become more smooth in the later stages than those in



(a) Normalized plot of operation probability parameters over time for the teacher network $\mathcal{N}_T$.



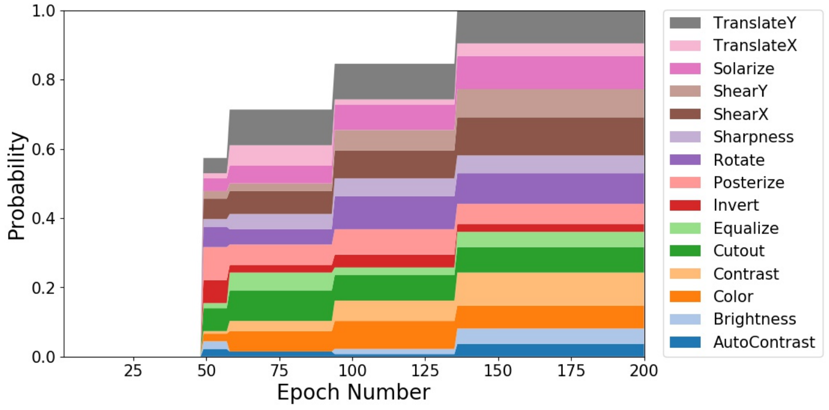
(b) Operation magnitude parameters over time for the teacher network $\mathcal{N}_T$.

Fig. 3: Evolution of magnitude and probability parameters in the learned schedules of teacher. Each operation appears in the parameter list twice, and we take the mean values of the parameter.

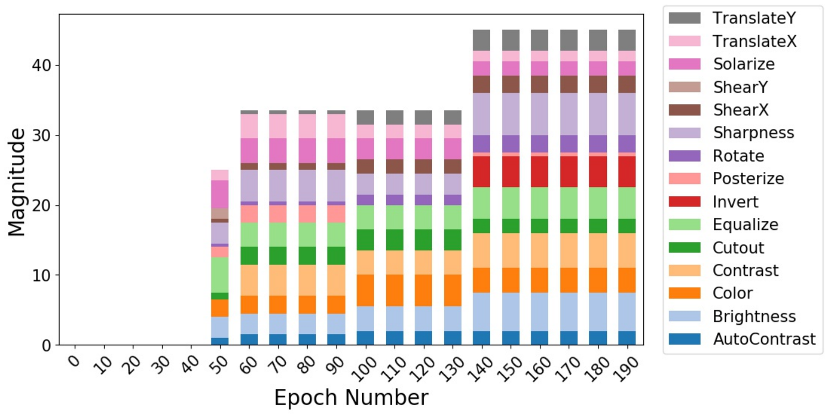
the early stages, and the learning of the student policies $\mathcal{A}_S$ can be seen as an extension and modification to $\mathcal{A}_T$, though in an indirect way by the KD signals. More importantly, this validates our assumption that $\mathcal{N}_S$ has her own optimal augmentation schedule $\mathcal{A}_S$ that is different from $\mathcal{A}_T$.

5.6 Comparisons on Full-Precision Networks

This subsection aims to verify the effectiveness of our proposed methods on more conventional KD settings where both $\mathcal{N}_T$ and $\mathcal{N}_S$ are full-precision networks. In particular, $\mathcal{N}_S$ is a shallow or a narrow network. We conduct experiments on various network achitectures, including Wide Residual Network (WRN) [36],



(a) Normalized plot of operation probability parameters over time for the student network $\mathcal{N}_S$.



(b) Operation magnitude parameters over time for the student network $\mathcal{N}_S$.

Fig. 4: Evolution of magnitude and probability parameters in the learned schedules of student.

PyramidNet [13] and PreResNet. In particular, we use WRN-28-10 (we use the standard notation WRN-d-k to refer to a wide residual network with depth d and width multiplier k), PreResNet164 and PyramidNet-200-240 (PyramidNet-d-k refers to a PyramidNet network with depth d and widening factor k) as the teacher networks while WRN-16-2, PreResNet56, PreResNet44 and PreResNet32 as the student networks.

Tab. 4 reports the accuracy on CIFAR-100 under different network settings of teachers and students. We observe that the improvement of augmentation with KD is significant increase of about 3% as compared to the vanilla baseline training. It shows that the discovered augmentation schedules further boosts the performance of the shallow $\mathcal{N}_S$ based on II-KD. In other words, our proposed method also works well on full-precision training tasks. Specifically, compared

Table 4: Accuracy on CIFAR-100 with full-precision under different settings for student network $\mathcal{N}_S$ and teacher network $\mathcal{N}_T$. **Vanilla** refers to training a full-precision student network from scratch. **After Stage-α** refers to using learned schedules discovered by PBA to re-train $\mathcal{N}_S$ as described in Section 4.1. **only II-KD** refers to training $\mathcal{N}_S$ using II-KD but without the learned data augmentation. **After Stage-β** refers to training $\mathcal{N}_S$ using II-KD and the learned data augmentation. The accuracies of teachers on CIFAR-100: WRN-28-10 has 80.73%, ResNet164 has 72.24% and PyramidNet-200-240 has 84.43%.

Teacher	Student	Vanilla	After Stage-α	Only II-KD	After Stage-β
WRN-28-10	WRN-16-2	72.68	73.79	74.41	76.19
WRN-28-10	WRN-16-4	77.28	78.01	79.30	80.59
WRN-28-10	PreResNet56	71.98	73.45	73.04	74.91
PreResNet164	PreResNet32	70.28	71.68	70.53	72.14
PyramidNet-200-240	PreResNet44	71.55	73.31	73.00	73.71
PyramidNet-200-240	WRN-16-2	72.68	73.79	74.64	75.03

with only using KD, the learned schedules help to improve the performance by about 1.5%.

Furthermore, it shows that when the accuracy gap between the teacher and student is large, the teacher can guide the training of the student better, which is consistent with previous KD literature [26,25]. For example, considering distilling knowledge into the same student WRN-16-2, PyramidNet-200-240 (with accuracy 84.43%) does a better job than WRN-28-10 (with accuracy 80.73%). However, when combining KD and data augmentation, the PyramidNet-200-240 behaves worse than the WRN-28-10 as a teacher. It seems that the improvement brought by the augmentation operations is more obvious when the teacher and student have similar network architectures.

6 Conclusion

Previous literature on KD focuses on exploring the knowledge representation and the strategies for distillation. However, both the teacher and student learn from the same training data without adapting the different learning capabilities. To address this issue, we propose customizing distinct agents to automatically augment the training data for the teacher and student, respectively. We extensively study the effectiveness of combining data augmentation and knowledge distillation. We also propose a simple feature-based KD variant that incorporates both intra- and inter-relationships within and across feature maps. We observe that the student can learn better from the teacher with the proposed approach, both in the challenging low-precision scenarios and with conventional full-precision networks. Furthermore, the teacher and student have their own optimal epoch-based augmentation schedules.

References

1. Bai, Y., Wang, Y.X., Liberty, E.: Proxquant: Quantized neural networks via proximal operators. In: Proc. Int. Conf. Learn. Repren. (2019)
2. Bandura, A.: Social cognitive theory in cultural context. Applied Psychology (2002)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
4. Brumfiel, G.: Who has designs on your students' minds? Nature Publishing Group (2005)
5. Bucilua, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM (2006)
6. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. pp. 5918–5926 (2017)
7. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: Proc. Adv. Neural Inf. Process. Syst. (2017)
8. Clark, K., Luong, M.T., Khandelwal, U., Manning, C.D., Le, Q.V.: Bam! born-again multi-task networks for natural language understanding. arXiv preprint arXiv:1907.04829 (2019)
9. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
10. DeVries, T., Taylor, G.W.: Dataset augmentation in feature space. arXiv preprint arXiv:1702.05538 (2017)
11. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks. arXiv preprint arXiv:1805.04770 (2018)
12. Gurlitt, J., Renkl, A., Motes, M.A., Hauser, S.: How can we use concept maps for prior knowledge activation: different mapping-tasks lead to different cognitive processes. In: Proceedings of the 7th international conference on Learning sciences. International Society of the Learning Sciences (2006)
13. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Proc. Eur. Conf. Comp. Vis. (2016)
16. He, T., Shen, C., Tian, Z., Gong, D., Sun, C., Yan, Y.: Knowledge adaptation for efficient semantic segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2019)
17. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Proc. Adv. Neural Inf. Process. Syst. Workshops (2014)
18. Ho, D., Liang, E., Stoica, I., Abbeel, P., Chen, X.: Population based augmentation: Efficient learning of augmentation policy schedules. In: Proc. Int. Conf. Mach. Learn. (2019)
19. Hyun Lee, S., Ha Kim, D., Cheol Song, B.: Self-supervised knowledge distillation using singular value decomposition. In: Proc. Eur. Conf. Comp. Vis. (2018)
20. Jung, S., Son, C., Lee, S., Son, J., Kwak, Y., Han, J.J., Choi, C.: Joint training of low-precision neural network with quantization interval parameters. arXiv preprint arXiv:1808.05779 (2018)

21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. Adv. Neural Inf. Process. Syst. (2012)
23. Lemley, J., Bazrafkan, S., Corcoran, P.: Smart augmentation learning an optimal data augmentation strategy. Ieee Access (2017)
24. Louizos, C., Reisser, M., Blankevoort, T., Gavves, E., Welling, M.: Relaxed quantization for discretized neural networks. In: Proc. Int. Conf. Learn. Repren. (2019)
25. Mirzadeh, S.I., Farajtabar, M., Li, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. arXiv preprint arXiv:1902.03393 (2019)
26. Mishra, A., Marr, D.: Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In: Proc. Int. Conf. Learn. Repren. (2018)
27. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2019)
28. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. In: Proc. Int. Conf. Learn. Repren. (2018)
29. Ratner, A.J., Ehrenberg, H., Hussain, Z., Dunnmon, J., Ré, C.: Learning to compose domain-specific transformations for data augmentation. In: Proc. Adv. Neural Inf. Process. Syst. (2017)
30. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: Proc. Int. Conf. Learn. Repren. (2015)
31. Seunghyun Lee, B.C.S.: Graph-based knowledge distillation by multi-head attention network. In: Proc. Brit. Mach. Vis. Conf. (2019)
32. Slotta, J.D., Chi, M.T.: Helping students understand challenging topics in science through ontology training. Cognition and instruction (2006)
33. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
34. Tran, T., Pham, T., Carneiro, G., Palmer, L., Reid, I.: A bayesian data augmentation approach for learning deep models. In: Proc. Adv. Neural Inf. Process. Syst. (2017)
35. Wei, Y., Pan, X., Qin, H., Ouyang, W., Yan, J.: Quantization mimic: Towards very tiny cnn for object detection. In: Proc. Eur. Conf. Comp. Vis. (2018)
36. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
37. Zhang, B., Wang, L., Wang, Z., Qiao, Y., Wang, H.: Real-time action recognition with enhanced motion vector cnns. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. pp. 2718–2726 (2016)
38. Zhang, D., Yang, J., Ye, D., Hua, G.: Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In: Proc. Eur. Conf. Comp. Vis. pp. 365–382 (2018)
39. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2018)
40. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
41. Zhuang, B., Shen, C., Tan, M., Liu, L., Reid, I.: Towards effective low-bitwidth convolutional neural networks. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2018)