Self-Knowledge Distillation with Progressive Refinement of Targets

Kyungyul Kim[1]   ByeongMoon Ji[1]   Doyoung Yoon[1]   Sangheum Hwang[2*]

[1]LG CNS AI Research, Seoul, South Korea
[2]Seoul National University of Science and Technology, Seoul, South Korea
{kyungyul.kim, jibm, dy0916}@lgcns.com, shwang@seoultech.ac.kr

Abstract

The generalization capability of deep neural networks has been substantially improved by applying a wide spectrum of regularization methods, e.g., restricting function space, injecting randomness during training, augmenting data, etc. In this work, we propose a simple yet effective regularization method named progressive self-knowledge distillation (PS-KD), which progressively distills a model's own knowledge to soften hard targets (i.e., one-hot vectors) during training. Hence, it can be interpreted within a framework of knowledge distillation as a student becomes a teacher itself. Specifically, targets are adjusted adaptively by combining the ground-truth and past predictions from the model itself. We show that PS-KD provides an effect of hard example mining by rescaling gradients according to difficulty in classifying examples. The proposed method is applicable to any supervised learning tasks with hard targets and can be easily combined with existing regularization methods to further enhance the generalization performance. Furthermore, it is confirmed that PS-KD achieves not only better accuracy, but also provides high quality of confidence estimates in terms of calibration as well as ordinal ranking. Extensive experimental results on three different tasks, image classification, object detection, and machine translation, demonstrate that our method consistently improves the performance of the state-of-the-art baselines. The code is available at https://github.com/lgcnsai/PS-KD-Pytorch.

1. Introduction

The recent progress made in deep neural networks (DNNs) has significantly improved performance in various tasks related to computer vision as well as natural language processing, e.g., image classification [14, 18, 22, 37], object detection / segmentation [13, 32], machine translation [41] and language modeling [20]. Scaling up of DNN is widely

adopted as a promising strategy to achieve higher performances [12, 14, 39]. However, deeper networks require a large number of model parameters that need to be learned, which could make models more prone to overfitting. Thus, DNNs typically produce overconfident predictions even for incorrect predictions, and this is because the predictions are highly miscalibrated [10, 26].

To improve generalization performance and training efficiency of DNNs, a number of regularization methods have been proposed. The widely employed methods in practice include: L_1- and L_2-weight decay [23, 27] to restrict the function space, dropout [36] to inject randomness during training, batch normalization [19, 33] to accelerate training speed by normalizing internal activations in every layer. There also have been several methods that are specifically designed for a particular task. For example, advanced data augmentation techniques that are specific to computer vision tasks such as Cutout [5], Mixup [51], AugMix [16] and CutMix [49] have shown to boost classification accuracy and also improve robustness and uncertainty of a model. Another effective regularization method is to adjust the targets when they are given in the form of one-hot coded vectors (i.e., hard targets), including label smoothing (LS) [38], label perturbation [43], etc.

Among those methods about adjusting targets, LS [38] has been widely applied to many applications [31, 41, 53] and has shown to improve generalization performance as well as the quality of confidence estimates (in terms of calibration) on image classification and machine translation tasks [25]. LS softens a hard target as a smoothed distribution by assigning a small amount of probability mass to non-target classes. However, it is also empirically confirmed that it is not complementary to current advanced regularization techniques. For example, if we utilize LS and CutMix simultaneously for image classification, the performance on both classification and confidence estimation is substantially degraded [3].

One natural question raised on LS could be: is there a more effective strategy to soften hard targets so as to obtain
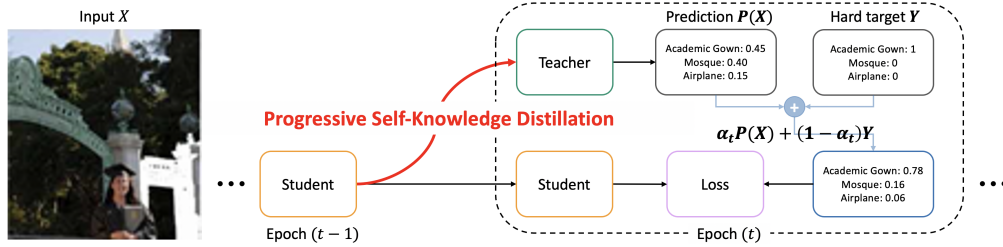
*Corresponding author

Figure 1. A schematic of PS-KD. At epoch t, a student at epoch $(t-1)$ becomes a teacher and a model at epoch t is trained with the soft targets computed as a linear combination of hard targets and the predictions from the teacher.

more informative labels? To answer this question, we propose a simple regularization technique named *progressive self-knowledge distillation* (PS-KD) that distills the knowledge in a model itself and uses it for training the model. It means that a student model becomes a teacher model itself, which gradually utilizes its own knowledge for softening hard targets to be more informative during training. Specifically, the model is trained with the soft targets which are computed as a linear combination of the hard targets and the past predictions at a certain epoch, which are adjusted adaptively as training proceeds.

To justify our proposed method, we have shown that *PS-KD gives more weights to hard-to-learn examples by a gradient rescaling scheme during training*, which clearly reveals that a student model can be enhanced even if trained with a teacher worse than the student (e.g., past predictions). The proposed method is easy to implement, can be applied to any supervised learning tasks where the hard targets are given as the ground-truth labels. Moreover, it can be easily combined with current advanced regularization techniques, thereby enhancing further their generalization performance. With this simple method, the generalization ability of DNNs can be greatly improved regarding the target metrics (e.g., accuracy) as well as the quality of confidence estimates in terms of both calibration and ordinal ranking (i.e., misclassification detection) [9, 24].

To rigorously evaluate the advantages of PS-KD, we conduct extensive experiments on diverse tasks with popular benchmark datasets: image classification on CIFAR-100 and ImageNet, object detection on PASCAL VOC, and machine translation on IWSLT15 and Multi30k. The experimental results demonstrate that training with PS-KD further enhances the generalization performance of the state-of-the-art baselines. For image classification on CIFAR-100 and ImageNet, our results show that the model trained with PS-KD provides not only better predictive performance, but also high quality of confidence estimates. We further confirm that the advanced image augmentation techniques such as Cutout and CutMix also benefit from PS-KD. From the evaluation on object detection with PASCAL VOC, it is observed that PS-KD makes a model learn better representa-

tions compared to the existing approaches. To show the wide applicability of PS-KD, we also conduct the experiments on machine translation, which improve BLEU scores of baselines significantly.

2. Related Work

Conventional Knowledge Distillation (KD) [17] methods use knowledge from a larger and better performing teacher model to generate soft targets for a student network. Recently, several works have tried to use the student network itself as a teacher, named *self-knowledge distillation* (self-KD) [11, 45, 50].

One approach to self-KD is to reduce the feature distance between similar inputs from a single network. Xu and Liu [45] propose a mechanism based on image distortion. Given an image, it generates two separate distorted images by using random mirroring and cropping. Then, model is trained to decrease the distance between features extracted from those two images. Yun *et al.* [50] propose a method called class-wise self-KD (CS-KD) which focuses on distilling knowledge between samples in the same class. For an input $\mathbf{x}$, another data $\mathbf{x}'$ with the same label is randomly sampled, and the KL divergence between predictive distributions from those is minimized during training. However, this approach may cause overfitting since it forces all samples in a specific class to have a high density in the learned representation space.

Another approach is to directly use outputs from a teacher whose architecture is exactly the same as a student. This approach is closer to the original notion of KD than the aforementioned approach. Born-Again Networks (BANs) [8] first train a network and use this pretrained network as a teacher for next generation. Then, it repeats this process, and thereby, performs multiple generations of KD where the k-th generation model is trained with knowledge distilled from the $(k-1)$-th model. Similar to BANs, Yuan *et al.* [48] empirically examine the common belief on KD and suggest the teacher-free KD (TF-KD) method, which uses a pretrained student as a teacher for a single generation. Zhang *et al.* [52] propose a method which di-

vides a network into several blocks and attaches auxiliary classifiers to them independently. These auxiliary classifiers are trained using outputs of the final classifier as a teacher's knowledge. Yang *et al.* [47] introduce utilizing a snapshot model at a certain epoch as a teacher: the whole training process is split into multiple mini-generations and the last snapshot model of each mini-generation is used as a teacher for the next mini-generation. These methods require augmenting a network's architecture [52] or several hyperparameters to be carefully tuned, e.g., a learning rate policy and the number of mini-generation [47].

Along with previous studies, our proposed PS-KD method uses a model's own predictions as a teacher's knowledge to enhance the generalization performance of DNNs. However, our method provides distinct advantages over them from a practical viewpoint: it does not require a pretraining phase unlike BANs [8] and TF-KD [48], and can be easily applied to any supervised learning tasks due to its generality and simplicity compared to [45, 11, 52, 47]. More importantly, it is shown that we can enjoy the advantages of KD even if a student is taught by a poor teacher (i.e., lower predictive performance than a student), which is distinguishable from previous works relying on a well-performing teacher.

3. Self-Knowledge Distillation

3.1. Knowledge Distillation as Softening Targets

KD is a technique to transfer knowledge from one model (i.e., a teacher) to another (i.e., a student), usually from a larger model to a smaller one. The student learns from more informative sources, the predictive probabilities from the teacher, besides one-hot labels. Hence, it can attain a similar performance compared to the teacher although it is usually a much smaller model, and show comparable or even better performance when the student has the same capacity as the teacher [8, 48].

For an input $\mathbf{x}$ and a K-dimensional one-hot target $\mathbf{y}$, a model produces the logit vector $\mathbf{z}(\mathbf{x}) = [z_1(\mathbf{x}), \cdots, z_K(\mathbf{x})]$, and then outputs the predicted probabilities $P(\mathbf{x}) = [p_1(\mathbf{x}), \cdots, p_K(\mathbf{x})]$ by a softmax function. Hinton et al. [17] suggest to utilize temperature scaling to soften these probabilities for better distillation:

$$\widetilde{p}_i(\mathbf{x}; \tau) = \frac{\exp(z_i(\mathbf{x})/\tau)}{\sum_j \exp(z_j(\mathbf{x})/\tau)} \qquad (1)$$

where τ denotes a temperature parameter. By scaling the softmax output $P^T(\mathbf{x})$ of the teacher as well as $P^S(\mathbf{x})$ of the student, the student is trained with the loss function $\mathcal{L}_{KD}$, given by:

$$\mathcal{L}_{KD}(\mathbf{x}, \mathbf{y}) = (1 - \alpha)H\left(\mathbf{y}, P^S(\mathbf{x})\right) + \\ \alpha\tau^2 H\left(\widetilde{P^T}(\mathbf{x}; \tau), \widetilde{P^S}(\mathbf{x}; \tau)\right) \qquad (2)$$

where H is a cross-entropy loss and α is a hyperparameter. Note that when the temperature τ is set to 1, Eq. (2) is equivalent to the cross-entropy of $P^S(\mathbf{x})$ to the soft target, a linear combination of $\mathbf{y}$ and $P^T(\mathbf{x})$:

$$\mathcal{L}_{KD}(\mathbf{x}, \mathbf{y}) = H\left((1 - \alpha)\mathbf{y} + \alpha P^T(\mathbf{x}), P^S(\mathbf{x})\right). \qquad (3)$$

Therefore, the existing methods that use the soft targets for regularization can be interpreted within the framework of knowledge distillation. For example, LS [25] is equivalent to distilling the knowledge from the teacher which produces uniformly distributed probabilities on any inputs.

3.2. Distilling Knowledge from the Past Predictions

We propose a new way of KD, called *progressive self-knowledge distillation* (PS-KD), which distills the knowledge of itself to enhance the generalization capability. In other words, the student becomes the teacher itself, and utilizes its past predictions to have more informative supervisions during training as can be seen in Fig. 1. Let $P_t^S(\mathbf{x})$ be the prediction about $\mathbf{x}$ from the student at t-th epoch. Then, our objective at t-th epoch can be written as:

$$\mathcal{L}_{KD,t}(\mathbf{x}, \mathbf{y}) = H\left((1 - \alpha)\mathbf{y} + \alpha P_{i<t}^S(\mathbf{x}), P_t^S(\mathbf{x})\right). \qquad (4)$$

Note that using the predictions from t-th epoch as the teacher's knowledge is trivial since it will not incur any loss.

The main difference from the conventional KD is that the teacher is not a static model, but dynamically evolves as training proceeds. Among all past models that are candidates for the teacher, we use the model at $(t-1)$-th epoch as the teacher since it can provide most valuable information among the candidates. Concretely, in t-th epoch of training, the target for the input $\mathbf{x}$ is softened as $(1-\alpha)\mathbf{y}+\alpha P_{t-1}^S(\mathbf{x})$. It is empirically observed that this approach utilizing the past model as a teacher regularizes the model effectively.

One more thing we have to consider is how to determine α in Eq. (4). The α controls how much we are going to trust the knowledge from the teacher. In the conventional KD, the teacher remains unchanged so the α is usually set to a fixed value during training. However, in PS-KD, the reliability of the teacher should be considered since the model generally does not have enough knowledge about data at the early stage of training. To this end, we increase the value of α gradually. Like the learning rate scheduling, there are several strategies to increase the α as a function of the epoch, e.g., step-wise, exponential, linear growth, etc. To minimize the number of hyperparameters involved in the scheduling, we apply the linear growth approach. Therefore, the α at t-th epoch is computed as follows:

$$\alpha_t = \alpha_T \times \frac{t}{T}, \qquad (5)$$

where T is the total epoch for training and α_T is the α at the last epoch, which is a single hyperparameter to be determined via validation process. Surprisingly, this simple

strategy combined with past predictions improves the generalization performance significantly across a wide range of tasks. To summarize, our objective function at t-th epoch can be written as:

$$\mathcal{L}_{KD,t}(\mathbf{x}, \mathbf{y}) = H\left((1 - \alpha_t)\mathbf{y} + \alpha_t P_{t-1}^S(\mathbf{x}), P_t^S(\mathbf{x})\right). \quad (6)$$

Theoretical support. We show that *PS-KD pays more attention to hard examples during training* when a hyperparameter α_t is properly set, and *the α_t value should be gradually increased to preserve such an effect of hard example mining*. This effect is realized by example re-weighting, motivated by Proposition 2 in Tang *et al.* [40]. The gradient of $\mathcal{L}_{KD,t}$ in Eq. 6 with respect to a logit value z_i, $i = 1, ..., K$ for a fixed α[1] is given by

$$\frac{\partial \mathcal{L}_{KD,t}}{\partial z_i} = \partial_i^{KD,t} = (1 - \alpha)(p_{t,i} - y_i) + \alpha(p_{t,i} - p_{t-1,i}). \quad (7)$$

Therefore, for z_{GT} where GT denotes the target class,

$$\begin{aligned} \partial_{GT}^{KD,t} &= (1 - \alpha)(p_{t,GT} - 1) + \alpha(p_{t,GT} - p_{t-1,GT}) \\ &= (p_{t,GT} - 1) - \alpha(p_{t-1,GT} - 1), \end{aligned} \quad (8)$$

and for z_i, $i \neq GT$,

$$\begin{aligned} \partial_i^{KD,t} &= (1 - \alpha)p_{t,i} + \alpha(p_{t,i} - p_{t-1,i}) \\ &= p_{t,i} - \alpha p_{t-1,i} \end{aligned} \quad (9)$$

If α is set to be $p_{t,i} - \alpha p_{t-1,i} \geq 0$ for all $i \neq GT$, i.e., $\alpha \leq \min_i (p_{t,i}/p_{t-1,i})$, then $(p_{t,GT} - 1) - \alpha(p_{t-1,GT} - 1) < 0$ and $\sum_{i \neq GT} |p_{t,i} - \alpha p_{t-1,i}| = (1 - p_{t,GT}) - \alpha(1 - p_{t-1,GT})$ holds. Therefore, L_1 norm $\sum_i |\partial_i^{KD,t}|$ of the gradient can be written as:

$$\sum_i |\partial_i^{KD,t}| = 2(1 - p_{t,GT}) - 2\alpha(1 - p_{t-1,GT}). \quad (10)$$

Let us consider the ratio of L_1 norms $\sum_i |\partial_i^{KD,t}|/\sum_i |\partial_i|$, which represents the gradient rescaling factor induced by PS-KD. By combining Eq. 10 and the fact that $\sum_i |\partial_i| = 2(1 - p_{t,GT})$, the rescaling factor is given by:

$$\frac{\sum_i |\partial_i^{KD,t}|}{\sum_i |\partial_i|} = 1 - \alpha\left(\frac{1 - p_{t-1,GT}}{1 - p_{t,GT}}\right) \equiv 1 - \alpha\left(\frac{\gamma_{t-1}}{\gamma_t}\right). \quad (11)$$

Note that γ represents the probability of being incorrect. Without loss of generality, it can be assumed that $p_{t,GT} \geq p_{t-1,GT}$ and $p_{t,i} \leq p_{t-1,i}$ for all $i \neq GT$ since P_t^S provides better prediction than P_{t-1}^S on training data. Therefore, $\gamma_{t-1} \geq \gamma_t$ always holds. A large $\frac{\gamma_{t-1}}{\gamma_t}$ means that

---

[1]For notational simplicity, we ignore t in α_t.
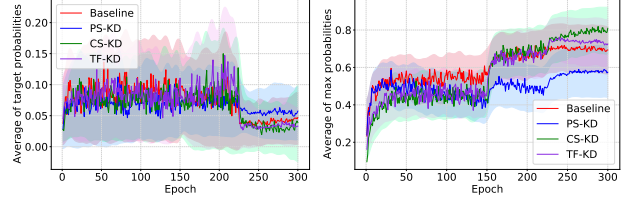
(a) Target probability (for GT)　　**(b)** Maximum probability

Figure 2. (a) Target and (b) maximum probabilities for 100 hard-to-learn samples (correctly classified less than 50 times during 300 epochs) on CIFAR-100. PS-KD keeps learning from hard examples by giving more weights to them.

predictions on that example are greatly improved during iterations (i.e., easy-to-learn). Conversely, hard-to-learn examples will have a small value. Consequently, the gradient rescaling factors for hard-to-learn examples are greater than those for easy-to-learn examples, which implies that PS-KD gives more weights on hard-to-learn examples during training, and it is empirically confirmed as shown in Fig. 2. Fig. 2 shows that other methods are more overconfident on incorrect predictions than PS-KD, and clearly demonstrates that PS-KD focuses on hard examples implicitly by the gradient rescaling scheme.

To expect such hard example mining effects, α should satisfy the condition described above for all examples. From this, α should be set to a sufficiently small value during an early training phase. As training proceeds, the differences of $\frac{\gamma_{t-1}}{\gamma_t}$'s for all examples become smaller. Therefore, α should be gradually increased to preserve the effect.

Implementation. For PS-KD, the predictions from the model at $(t - 1)$-th epoch are necessary for training at t-th epoch. There are two ways to obtain these past predictions. One is to load the model at $(t - 1)$-th epoch on memory when t-th epoch is started so that the past predictions for softening targets are also computed in forward passes. The other is to save the past predictions on disk in advance during $(t - 1)$-th epoch, and read this information to compute the soft targets at t-th epoch. These two approaches have pros and cons. The former way may need more GPU memory. On the other hand, the latter way does not need additional GPU memory but requires more space to store past predictions.

The choice of how to obtain the past predictions depends on the task we are dealing with. For example, on machine translation with a large-scale corpus, it is nearly impossible to store the predicted probabilities over all tokens. For this, we can choose the former strategy. Note that softening targets via a moving average [1] or distilling knowledge with task-specific operations [45, 50] is not applicable to this task. In our experiments, we employ an efficient way according to the task, e.g., the past predictions from the model on GPU memory is utilized for the tasks on ImageNet classification and IWSLT15 machine translation.

4. Experimental Results

In this section, we show the effectiveness of PS-KD across a variety of tasks including image classification, object detection, and machine translation. More details on datasets, evaluation metrics and experimental settings are available in the supplementary material. All experiments were performed on NVIDIA DGX-1 system with Py-Torch [30].

4.1. CIFAR-100 Classification

On CIFAR-100 classification, we consider four CNN models: ResNet [14], ResNeXt [44], DenseNet [18], and PyramidNet [12]. First, we compare PS-KD with LS and recent self-KD methods, CS-KD [50] and TF-KD [48], on the architectures we consider. Then, we demonstrate that PS-KD is complementary to the existing regularization methods including data augmentation (e.g. Cutout [5], Cut-Mix [49], etc.), and ensembles.

Experimental settings. The detailed architectures we consider are PreAct ResNet-18 [15], ResNet-101 [14], ResNeXt-29 (cardinality=8, width=64) [44], DenseNet-121 (growth rate=32) [18] and PyramidNet-200 (widening factor=240) [12]. We follow standard data augmentation schemes: 32×32 random crop after padding with 4 pixels and random horizontal flip. All CNNs are trained using SGD with a momentum of 0.9 for 300 epochs, and the learning rate is decayed by a factor of 10 at 150 and 225 epochs. For ResNet, ResNeXt, and DenseNet, we set the mini-batch size, a weight decay, and an initial learning rate to 128, 0.0005, and 0.1, respectively. For PyramidNet, the mini-batch size, a weight decay, and an initial learning rate are set to 64, 0.0001, and 0.25, respectively, following [12, 49].

To compare the performance of PS-KD with LS, CS-KD[2] and TF-KD[3], the hyperparameters are set according to those reported in the corresponding studies. Our PS-KD has only a single hyperparameter α_T. To determine the optimal α_T, we use randomly sampled 10% of training data as a validation dataset. In this experiment, we set the α_T to 0.8 which shows the optimal validation performance in terms of accuracy and calibration on ResNet-18.[4] With this parameter value, we then train a model on the entire training dataset for a fair comparison. Note that the value tuned for ResNet-18 is also used for other all architectures since we expect that PS-KD is fairly robust to the hyperparameter α_T, and it is confirmed from the experimental results.

For existing regularization methods to be combined with PS-KD, we also follow the hyperparameter values reported

---

[2]https://github.com/alinlab/cs-kd

[3]https://github.com/yuanli2333/Teacher-free-Knowledge-Distillation

[4]The hyperparameters of each method and the ablation study on α_T are provided in the supplementary material.

Model + Method	Top-1 Err (%)	Top-5 Err (%)	NLL	ECE (%)	AURC ($\times 10^3$)
ResNet-18	24.18	6.90	1.10	11.84	67.65
+ LS	20.94	6.02	0.98	10.79	57.74
+ CS-KD	21.30	5.70	0.88	6.24	56.56
+ TF-KD	22.88	6.01	1.05	11.96	61.77
+ PS-KD	**20.82**	**5.10**	**0.76**	**1.77**	**52.10**
ResNet-101	20.75	5.28	0.89	10.02	55.45
+ LS	19.84	5.07	0.93	**3.43**	95.76
+ CS-KD	20.76	5.62	1.02	12.18	64.44
+ TF-KD	20.10	5.10	0.84	6.14	58.8
+ PS-KD	**19.43**	**4.30**	**0.74**	6.92	**49.01**
DenseNet-121	20.05	4.99	0.82	7.34	52.21
+ LS	19.80	5.46	0.92	3.76	91.06
+ CS-KD	20.47	6.21	1.07	13.80	73.37
+ TF-KD	19.88	5.10	0.85	7.33	69.23
+ PS-KD	**18.73**	**3.90**	**0.69**	**3.71**	**45.55**
ResNeXt-29	18.65	4.47	0.74	**4.17**	44.27
+ LS	17.60	4.23	1.05	22.14	41.92
+ CS-KD	18.26	4.37	0.80	5.95	42.11
+ TF-KD	17.33	3.87	0.74	6.73	40.34
+ PS-KD	**17.28**	**3.60**	**0.71**	9.15	**39.78**
PyramidNet	16.80	3.69	0.73	8.04	36.95
+ LS	17.82	4.72	0.89	3.46	105.02
+ CS-KD	18.31	5.70	1.17	14.70	70.05
+ TF-KD	16.48	3.37	0.79	10.48	37.04
+ PS-KD	**15.49**	**3.08**	**0.56**	**1.83**	**32.14**

Table 1. Evaluation results on CIFAR-100 compared to other methods with popular architectures, averaged over three runs. The best result is shown in boldface.

in the literature, for example, the hole size in Cutout is set to 8 and the parameter α of Beta distribution in CutMix is set to 1. Cutout and CutMix produce randomly synthesized images from two inputs at every iteration. In this case, applying PS-KD with them at the same time is not straightforward. Therefore, for the experiments where PS-KD is combined with Cutout or CutMix, each data selects the regularization method with a probability of 0.5. Simply, PS-KD is applied to the half of the data in a randomly shuffled mini-batch, and Cutout or CutMix is performed on another half of the data.

Evaluation metrics. We use top-1 and top-5 error as standard performance measures for multi-class classification. We also employ the negative log-likelihood (NLL), expected calibration error (ECE) [29] and the area under the risk-coverage curve (AURC) [9] to evaluate the quality of predictive probabilities in terms of confidence estimation. ECE is a widely used metric to determine whether a model's predictions are well-calibrated, approximating the difference in expectation between classification accuracy and confidence estimates. AURC measures the area under the curve from plotting the risk (i.e., error rate) according to coverage. A low AURC implies that correct and incorrect predictions can be well-separable based on confidence esti-

mates. In these experiments, the maximum class probability is used as a confidence estimator.

Result. The comparison results are summarized in Table 1. Note that we report average values over three runs. First, we observe that training with PS-KD performs better than baseline and LS in terms of classification accuracy across all architectures, e.g., an improvement of 1.37% and 0.32% from baseline and LS on ResNeXt, respectively. Compared with CS-KD and TF-KD, PS-KD also shows better accuracy while significantly improving the performance on confidence estimation, for example, it improves accuracy by 1.74% and 1.15% from CS-KD and TF-KD on DenseNet-121, respectively, and it reduces ECE by 12.87% and 8.65% from CS-KD and TF-KD on PyramidNet, respectively. The performance improvement on confidence estimation is consistently observed across all metrics (i.e., NLL, ECE, and AURC) except for the two cases, ECE on ResNet-101 and ResNeXt.[5] Nevertheless, PS-KD provides robust calibration performance on all architectures, e.g., less than 10% in ECE, as can be seen from the calibration plots in Fig. 3. Interestingly, ResNeXt provides well-calibrated predictions in terms of ECE, and it would be worth investigating which architectural factors of CNNs affect calibration performance.

From the results, it is confirmed that PS-KD is the only method which shows consistent and robust performance improvement across all metrics. Importantly, CS-KD performs worse than LS or even baseline in some cases from our experiments. We suspect that these results are caused by the implicit property of CS-KD, which pulls all samples in the same class each other, and it may accelerate overfitting when a model has high capacity or hyperparameters are not properly tuned.

To show that PS-KD can be used in conjunction with other advanced regularization methods, we present the detailed experimental results on PyramidNet in Table 2. Compared with Cutout and CutMix on PyramidNet, PS-KD shows slightly higher accuracy while significantly improving performances on confidence estimation. We observe the top-1 error of 14.82% when Cutout is combined with PS-KD, which is 1.23% improvement of Cutout. When PS-KD, CutMix, and SD are utilized simultaneously, the top-1 error from the combination of CutMix and SD is reduced by 0.48%. In this setting, it is confirmed again that PS-KD provides a positive effect on confidence estimation: all metrics, NLL, ECE, and AURC, are improved by PS-KD. More experimental results on other self-KD methods are presented in the supplementary material. These results demonstrate that current state-of-the-art regularization methods benefit

---

[5]When α_T is tuned on ResNeXt-29, we observed that most metrics are further improved: for $\alpha_T = 0.7$, Top-1=17.06%, Top-5=3.68%, NLL=0.69, ECE=6.3%, and AURC=38.64.

Model + Method	Top-1 Err (%)	Top-5 Err (%)	NLL	ECE (%)	AURC (×10³)
PyramidNet	16.80	3.69	0.73	8.04	36.95
+ PS-KD	**15.49**	**3.08**	**0.56**	**1.83**	**32.14**
+ Cutout [5]	16.05	3.42	0.67	7.15	33.20
+ Cutout + PS-KD	**14.82**	**2.86**	**0.54**	**3.69**	**29.77**
+ CutMix [49]	15.62	3.38	0.68	8.16	34.60
+ CutMix + PS-KD	**15.03**	**2.91**	**0.58**	**5.81**	**30.22**
+ CutMix + SD [46]	14.07	2.38	0.51	3.96	28.65
+ CutMix + SD + PS-KD	**13.59**	**2.18**	**0.49**	**3.46**	**25.98**

Table 2. Performance enhancement of data augmentation methods, Cutout and CutMix, by PS-KD, averaged over three runs. The best result is shown in boldface.

Model + Method	Top-1 Err (%)	Top-5 Err (%)	NLL	ECE (%)	AURC (×10³)
ResNet-18	21.36	5.05	0.89	4.29	54.61
+ CS-KD	**18.39**	4.21	0.74	**3.07**	43.33
+ TF-KD	21.07	4.80	0.90	6.48	52.70
+ PS-KD	18.79	**4.17**	**0.68**	5.12	44.29
ResNet-101	18.27	3.99	0.74	4.01	44.46
+ CS-KD	17.97	4.00	0.78	4.61	43.83
+ TF-KD	18.16	3.97	0.71	**1.52**	43.66
+ PS-KD	**17.26**	**3.42**	**0.63**	2.15	**37.85**
DenseNet-121	17.09	3.51	0.65	1.96	39.01
+ CS-KD	17.04	3.85	0.73	4.08	42.40
+ TF-KD	16.96	3.32	0.71	5.36	39.31
+ PS-KD	**16.25**	**2.85**	**0.57**	**1.92**	**34.64**
ResNeXt-29	16.72	3.44	**0.65**	**3.78**	36.98
+ CS-KD	16.86	3.51	0.75	8.58	37.06
+ TF-KD	16.03	3.33	0.70	8.89	35.14
+ PS-KD	**15.99**	**3.10**	0.68	11.1	**21.79**
PyramidNet	14.58	2.85	0.60	**2.63**	30.04
+ CS-KD	15.29	3.93	0.76	4.72	36.53
+ TF-KD	14.77	2.77	0.65	5.60	30.55
+ PS-KD	**14.11**	**2.58**	**0.50**	2.79	**27.29**

Table 3. Performance improvement of ensembles by PS-KD. For ensembling, three trained models in Table 1 are used. The best result is shown in boldface.

from PS-KD in terms of not only classification accuracy, but also confidence estimation. From the previous study [3], it is known that LS might be harmful to generalization performance when applied concurrently with the advanced methods. Our empirical findings reveal that how to soften the hard targets is important and the distilled knowledge from a model itself can be a good source to create more informative targets.

We also examine the ensemble effects of PS-KD. For ensembling, we utilize the three trained models from the previous experiment. Table 3 shows the performances of ensembles from baseline, CS-KD, TF-KD, and PS-KD models. From the results, it is shown that ensembles can also benefit from PS-KD, which implies that PS-KD does not degrade the diversity of independently trained models.
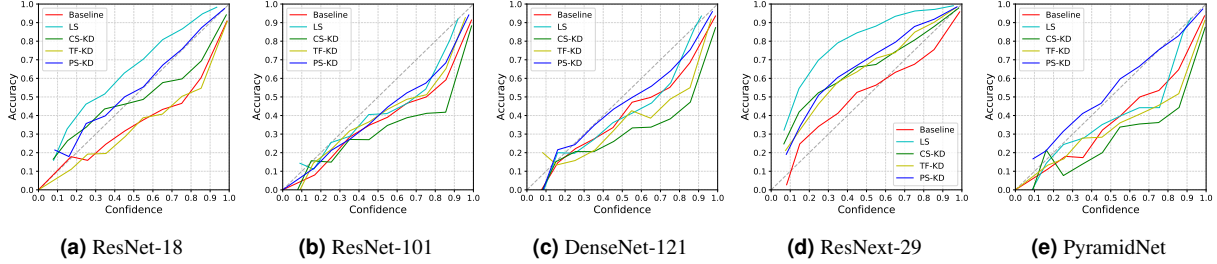
Figure 3. Calibration plots of all architectures on CIFAR-100. A diagonal dashed line represents perfect calibration.

Model + Method	Top-1 Err (%)	Top-5 Err (%)	NLL	ECE (%)	AURC ($\times 10^3$)
DenseNet-264* [18]	22.15	6.12	-	-	-
ResNeXt-101* [44]	21.20	5.60	-	-	-
ResNet-152	22.19	6.19	0.88	3.84	61.79
+ LS	21.73	5.85	0.92	3.91	68.24
+ CS-KD	21.61	5.92	0.90	5.79	62.12
+ TF-KD	22.76	6.43	0.91	4.70	65.28
+ PS-KD	21.41	5.85	0.84	2.51	61.01
+ CutMix	21.04	5.56	0.81	2.19	58.43
+ CutMix + LS	20.77	5.36	0.85	1.90	63.45
+ CutMix + CS-KD	21.08	5.53	0.83	1.56	59.02
+ CutMix + TF-KD	22.00	5.93	0.85	2.18	62.57
+ CutMix + PS-KD	**20.76**	**5.34**	**0.80**	**0.54**	**58.25**

Table 4. Top-1/top-5 error, NLL, ECE and AURC results on ImageNet validation dataset. '*' denotes results reported in the original papers. The best result is in bold.

4.2. ImageNet Classification

In the case of a large-scale dataset like ImageNet [4], the knowledge (i.e., predictions) from the previous snapshot model at $(t-1)$-th epoch might be too outdated since the model at t-th epoch further learns from a large number of samples during a single epoch. Nevertheless, we observe that the model benefits from PS-KD even for such a large-scale dataset.

Experimental settings. As a baseline, we train PS-KD using ResNet (depth=152) with standard data augmentation schemes including random resize cropping, random horizontal flip, color jittering, and lighting, following [49]. We train ResNet-152 for 90 epochs with a weight decay of 0.0001 and an initial learning rate of 0.1, followed by decaying the learning rate by a factor of 10 at 30 and 60 epochs. We employ SGD with a momentum of 0.9 as an optimizer and set the mini-batch size to 256. For LS, we set the hyperparameter ϵ as 0.1. Since the optimal hyperparameters of CS-KD and TF-KD on ImageNet are not reported in the literature, we conduct a random search over five runs to compare in a fair setting.[6]

Result. Table 4 shows performances evaluated by the metrics used in the previous section. Our method shows

---

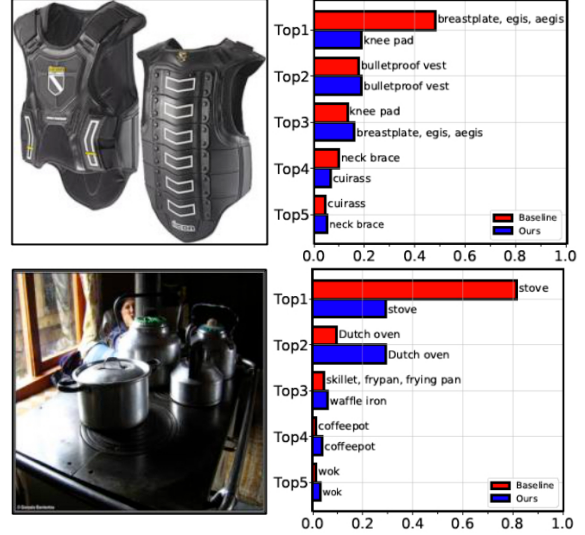[6]All results can be found in the supplementary material.

Figure 4. Predicted probabilities for samples in the validation dataset from baseline and PS-KD. The ground-truth labels of these images are "bulletproof vest" (top), and "stove" (bottom).

better accuracy than LS and other self-KD methods, achieving a top-1 error of 21.41%. Also, PS-KD achieves better performance on confidence estimation, i.e., it reduces ECE by 3.28% for CS-KD and 2.19% from TF-KD, respectively. From our validation results provided in the supplementary material, we observe that CS-KD is sensitive to the hyperparameters while PD-KD is much more robust to the hyperparameter α_T. Additionally, PS-KD is further improved on all metrics when combined with CutMix, especially in terms of ECE. It is consistent with the results on CIFAR-100, which demonstrates that PS-KD provides additional benefits to the existing regularization methods. On the other hand, other self-KD methods in conjunction with CutMix are even worse than the vanilla CutMix. We expect that the performance improvement can be greater if the knowledge from the recent past model is utilized, for example, the predictions from the model at $(t-0.5)$-th epoch.

Examples of how our PS-KD improves the quality of predicted probability are shown in Fig. 4 (see the supplementary material for more examples). For the top image whose label is "bulletproof vest", both baseline and PS-

KD produce an incorrect prediction. However, PS-KD outputs the class probabilities distributed over the classes that have similar visual characteristics while the baseline outputs overconfident prediction on non-target class. The bottom image is labeled as "stove" while containing multiple objects including "coffee pot" and "stove". Both baseline and PS-KD correctly classify this image, however, PS-KD also produces a high probability on "Dutch oven" that is visually similar to the objects in the image. These quantitative and qualitative results support the advantage of PS-KD which acts as an effective and strong regularizer.

4.3. Object Detection

We also examine that other visual recognition tasks can benefit from PS-KD. For this, we perform the experiment on the task of object detection using PASCAL VOC [7] dataset. We use the 5k VOC 2007 *trainval* and 15k VOC 2012 *trainval* as training sets, and use the PASCAL VOC 2007 *test* as a test set, following [32, 49]. As a baseline, Faster R-CNN [32] is considered, and the improvement of detection performance is examined by replacing the original VGG-16 [34] backbone network with a ResNet-152 trained on ImageNet. We utilize six different backbones trained for the previous section: ResNet-152, ResNet-152 with LS, PS-KD, CS-KD, TF-KD, and PS-KD+CutMix. We then fine-tune Faster R-CNN with each backbone network for 10 epochs with a mini-batch size of 1, an initial learning rate of 0.001 decayed by a factor of 10 at 5 epochs.

Method	mAP (IoU > 0.5) (%)
ResNet-152	78.26
+ LS	78.44
+ CS-KD	78.33
+ TF-KD	78.28
+ PS-KD	79.50
+ PS-KD + CutMix	**79.72**

Table 5. Effect of PS-KD as a pretrained backbone network for Faster R-CNN. The mAP value is computed by averaging APs over classes.

As shown in Table 5, ResNet-152 with PS-KD significantly improves the detection performance by 1.06%, 1.17%, and 1.22% of the mean average precision (mAP) compared to ResNet-152 with LS, CS-KD, and TF-KD, respectively.[7] Furthermore, PS-KD shows better mAP when it is combined with CutMix. Note that this improvement is achieved by just replacing the backbone network. From this result, it is verified that training with PS-KD provides a strong backbone network, which provides generic representations that can be transferred to other visual recognition tasks.

---

[7]APs over all classes are presented in the supplementary material.

4.4. Machine Translation

To verify the effectiveness of PS-KD on other tasks rather than multi-class classification, a machine translation task where classification is performed on a token-level, not an input-level is considered.

We use two benchmark datasets including IWSLT15 English to German (EN-DE) and German to English (DE-EN) [2], and Multi30k [6] from WMT16 [35]. The original purpose of Multi30k is for multimodal learning, consisting of images and descriptions associated with them. For the experiment, we extract only image descriptions written in English and German translations by professional translators. This dataset consists of 29K train data, 1K validation data, and 10K test data with 9,521 vocabularies.

We consider Transformer [41] as our baseline model.[8] All hyperparameters involved in the architecture and training are set to those reported in [42]. In specific, we use the architecture with $N = 6$, $d_{model} = 512$, $h = 4$, $d_k = 64, d_{ff} = 1024$. We train the model for 150 epochs with a maximum of 4,096 tokens per a mini-batch, and employ Adam optimizer [21] with $\beta_1 = 0.9$, $\beta_2 = 0.98$. As a metric, BLEU, commonly used to evaluate the performance on machine translation, is used. The hyperparameter $\alpha_T = 0.7$ is determined through validation.

Model	IWSLT15		Multi30k
+ Method	EN-DE	DE-EN	DE-EN
Transformer	28.5	34.6	29.0
+ LS	29.3	35.6	29.3
+ PS-KD	**30.0**	**36.2**	**32.3**

Table 6. BLEU scores on Transformer with LS or PS-KD

The results are summarized in Table 6. Our PS-KD achieves the best BLEU scores on all datasets. Consistent with the results from image classification and object detection, PS-KD shows better performance than the baseline Transformer and that with LS.

5. Conclusion

We propose a simple way to improve the generalization performance of DNNs, which distills the knowledge of a model itself to generate more informative targets for training. The targets are softened by using past predictions about data from the model at the previous epoch. We also provide theoretical justification, which shows that our method performs hard example mining implicitly during training. From the experimental results conducted across diverse tasks, we observe that the proposed method is effective to improve the generalization capability of DNNs.

---

[8]Experiments were conducted using Fairseq (https://github.com/pytorch/fairseq) toolkit [28].

References

[1] A. Aghajanyan. Softtarget regularization: An effective technique to reduce over-fitting in neural networks. In *The IEEE International Conference on Cybernetics*, 2017. 4

[2] M. Cettolo, J. Niehues, S. Stuker, L. Bentivogli, R. Cattoni, and M. Federico. The iwslt 2015 evaluation campaign. In *International Workshop on Spoken Language Translation*, 2015. 8

[3] Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods. In *ICLR Workshop on Uncertainty and Robustness in Deep Leaning*, 2019. 1, 6

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 7

[5] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 1, 5, 6

[6] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *ACL Workshop on Vision and Language*, 2016. 8

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 8

[8] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, 2018. 2, 3

[9] Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*, 2019. 2, 5

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017. 1

[11] Sangchul Hahn and Heeyoul Choi. Self-knowledge distillation in natural language processing. *arXiv preprint arXiv:1908.01851*, 2019. 2, 3

[12] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 5

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision*, 2017. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 5

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. 5

[16] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020. 1

[17] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Workshop on Deep Learning and Representation Learning*, 2015. 2, 3

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 5, 7

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Learning Representations*, 2015. 1

[20] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 1

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 8

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 1

[23] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, 1992. 1

[24] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. In *International Conference on Machine Learning*, 2020. 2

[25] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 2019. 1, 3

[26] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1

[27] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992. 1

[28] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *North American Chapter of the Association for Computational Linguistics*, 2019. 8

[29] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015. 5

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,

Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. 5

[31] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, 2019. 1

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 1, 8

[33] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, 2018. 1

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 8

[35] Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. A shared task on multimodal machine translation and crosslingual image description. In *Association for Computational Linguistics*, 2016. 8

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 1

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1

[38] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[39] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 2019. 1

[40] Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H. Chi, and Sagar Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2021. 4

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 1, 8

[42] Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. 8

[43] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing CNN on the loss layer. *CoRR*, abs/1605.00055, 2016. 1

[44] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5, 7

[45] Ting-Bing Xu and Cheng-Lin Liu. Data-distortion guided self-distillation for deep neural networks. In *AAAI Conference on Machine Learning*, 2019. 2, 3, 4

[46] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019. 6

[47] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L. Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 3

[48] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020. 2, 3, 5

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *The IEEE International Conference on Computer Vision*, 2019. 1, 5, 6, 7, 8

[50] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. Regularizing class-wise predictions via self-knowledge distillation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 4, 5

[51] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 1

[52] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *The IEEE International Conference on Computer Vision*, 2019. 2, 3

[53] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1