3.Self-Knowledge Distillation_A Simple Way for Be

Abstract 1

1Introduction 1

regularization methods:

- L_1, L_2 weight decay
- dropout
- batch norm
- advanced data augmentation: Cutup, Mixup, etc
- Label smoothing: inceptionV2 (LS)

LS:

hard label $y_i = \begin{cases} 1 & (i=k) \\ 0 & (i \neq k) \end{cases}$

$\Downarrow$ soft

soft label $y_i = \begin{cases} 1-\epsilon & (i=k) \\ \frac{\epsilon}{K} & (i \neq k) \end{cases}$

ϵ is a small hyper-parameter

K is the amount of classes

2Self-Knowledge Distillation 2

- 2.1Knowledge Distillation as Softening Targets 2
- 2.2Distilling Knowledge from the Past Predictions 3

3Experimental Results 4

- 3.1CIFAR-100 Classification 4
- 3.2ImageNet Classification 6

  作者认为对于这种大尺度的数据集来说，上一个epoch训来的模型来作老师有点过时，他想用t-0.5这种

- 3.3Object Detection 7
- 3.4Machine Translation 7

4Related Works 8

5Conclusion 8

3.Self-Knowledge Distillation_A Simple Way for Be

- Abstract 1

- 1Introduction 1

regularization methods:

- L_1, L_2 weight decay
- dropout
- batch norm
- advanced data augmentation : Cutup, Mixup, etc
- Label smoothing : inception V_2 (LS)

LS:

hard label $y_i = \begin{cases} 1 & (i=k) \\ 0 & (i \neq k) \end{cases}$

$\Downarrow$ soft

soft label $y_i = \begin{cases} 1-\epsilon & (i=k) \\ \frac{\epsilon}{K} & (i \neq k) \end{cases}$

ϵ is a small hyper-parameter

K is the amount of classes

- 2Self-Knowledge Distillation 2
  - 2.1Knowledge Distillation as Softening Targets 2
  - 2.2Distilling Knowledge from the Past Predictions 3
- 3Experimental Results 4
  - 3.1CIFAR-100 Classification 4
  - 3.2ImageNet Classification 6

作者认为对于这种大尺度的数据集来说，上一个epoch训来的模型来作老师有点过时，他想用t-0.5这种

  - 3.3Object Detection 7

3.Self-Knowledge Distillation_A Simple Way for Be

Self-Knowledge Distillation: A Simple Way for Better Generalization

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 *self-knowledge distillation* (Self-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. 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, we show that Self-KD achieves not only better accuracy, but also provides high quality of confidence estimates. 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, and especially, it achieves state-of-the-art BLEU score of 30.0 and 36.2 on IWSLT15 English-to-German and German-to-English tasks, respectively.

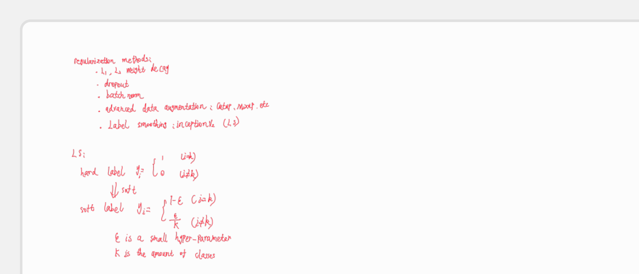
1 Introduction

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 [13, 16, 20, 35], object detection / segmentation [12, 30], machine translation [38] and language modeling [18]. Scaling up of DNNs [11, 13, 37] is widely adopted as a promising strategy to achieve higher performance. However, deeper networks require a large number of model parameters that need to be learned, which could make the model more prone to overfitting. Thus, DNNs typically produce overconfident predictions even for incorrect predictions, and this is because the predictions are highly miscalibrated [9, 24].

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 [21, 25] to restrict the function space, dropout [34] to inject randomness during training, batch normalization [17, 31] 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 [4], Mixup [46], AugMix [14] and CutMix [44] have shown to boost classification accuracy, improve robustness and uncertainty of a model. Another effective regularization method is to adjust the targets

[*]Corresponding author

Preprint. Under review.



3.Self-Knowledge Distillation_A Simple Way for Be

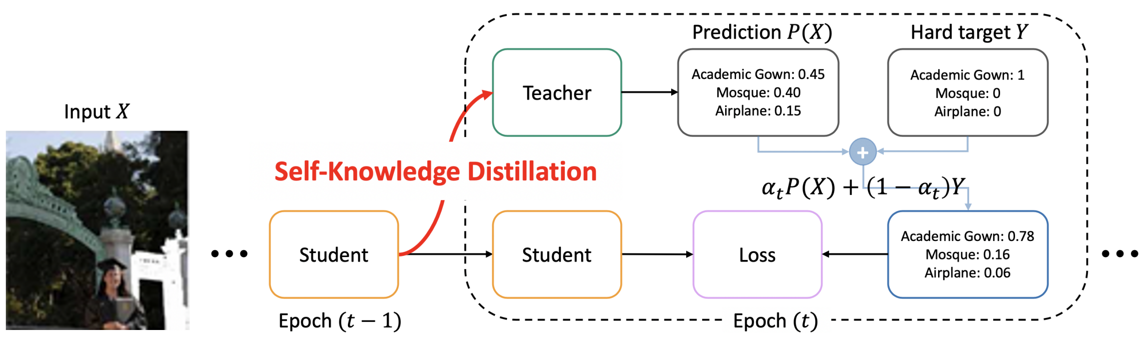


Figure 1: A schematic of Self-KD. At epoch t, the student at epoch $(t-1)$ becomes the teacher and a model at epoch t is trained with the soft targets computed as a linear combination of hard targets and the predictions from the teacher.

when they are given in a form of one-hot coded vectors (i.e., hard targets), including label smoothing (LS) [36], label perturbation [40], etc.

Among those methods about adjusting targets, LS [36] has been widely applied to many applications [29, 38, 48] 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 [23]. This method 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 LS 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 [2].

motivation

One natural question raised on LS could be: is there a more effective strategy to soften hard targets so as to obtain more informative labels? To answer this question, we propose a simple regularization technique named *self-knowledge distillation* (Self-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 the 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. 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. 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.

To rigorously evaluate the advantages of the proposed method, 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 Self-KD further enhances the state-of-the-art baselines. For image classification, our results show that Self-KD outperforms the state-of-the-art regularization techniques in terms of confidence estimation. In particular, on machine translation, we achieve state-of-the-art BLEU score on IWSLT15 English-to-German and German-to-English.

2 Self-Knowledge Distillation

2.1 Knowledge Distillation as Softening Targets

Knowledge distillation [15] 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 much smaller model, and show even better performance when the student has the same capacity with the teacher [7].

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. [15] suggests to utilize temperature scaling to soften these probabilities

3.Self-Knowledge Distillation_A Simple Way for Be

for better distillation:

$$\widetilde{p_i}(\mathbf{x};\tau) = \frac{\exp(z_i(\mathbf{x})/\tau)}{\sum_j \exp(z_j(\mathbf{x})/\tau)} \tag{1}$$

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:

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

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 $P^T(\mathbf{x})$:

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

Therefore, the existing methods that use the soft targets for regularization can be interpreted within the framework of knowledge distillation. For example, LS [23] is equivalent to distilling the knowledge from the teacher which produces uniformly distributed probabilities on any inputs.

2.2 Distilling Knowledge from the Past Predictions

We propose a new way of knowledge distillation, called *self-knowledge distillation* (Self-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:

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

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 knowledge distillation 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.

动态改变α

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 knowledge distillation, the teacher remains unchanged so the α is usually set to a fixed value during training. However, in Self-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 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:

$$\alpha_t = \alpha_T \times \frac{t}{T}, \tag{5}$$

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

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

Implementation. For Self-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 pass. The other is to save the past predictions on disk in advance during $(t-1)$-th epoch, and read these information to compute the soft targets at t-th epoch. These two approaches have the pros and cons. The former way needs more GPU memory and increases the computation time due to the extra forward pass[2]. 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 task 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 although it increases the training time. Note that softening targets via a moving average presented in [1] is not applicable to such large-scale datasets. 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.

3 Experimental Results

In this section, we show the effectiveness of Self-KD across a variety of tasks including image classification, object detection and machine translation. More details on datasets, evaluation metrics are available in the supplementary material. All experiments were performed on NVIDIA DGX-1 system with PyTorch [28].

3.1 CIFAR-100 Classification

On CIFAR-100 classification, we consider four popular CNN models: ResNet [13], ResNeXt [41], DenseNet [16], and PyramidNet [11]. First, we compare Self-KD with LS as a baseline on ResNet, ResNeXt, and DenseNet. Then, we also compare with the state-of-the-art regularization methods including Cutout [4], CutMix [44], and ShakeDrop [43] (SD) on PyramidNet to show that the combination of Self-KD with them can further enhance the performance of a classifier.

Experimental settings. The detailed architectures we consider are ResNet (depth=50,101) [13], ResNeXt-29 (cardinality=8, width=64) [41], DenseNet-BC (growth rate=12, depth=100) [16] and PyramidNet-200 (widening factor=240) [11]. 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 is set to 64, 0.0001, and 0.25, respectively, following to [11, 44].

The hyperparameters are set according to those reported in the corresponding studies. For LS, we use the smoothing parameter ϵ of 0.1 [36]. We set the hole size of Cutout to 8×8 pixels following to [4]. For CutMix, the parameter α of Beta distribution (i.e., combination ratio) is set to 1 [44]. Our Self-KD has one 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 optimal α_T to 0.7 which shows the best top-1 error on the validation dataset. We then train a model by using the whole dataset for a fair comparison.

Cutout and CutMix produce randomly synthesized images from two inputs at every iteration. In this case, applying Self-KD with them at the same time is not straightforward. Therefore, for the experiments where Self-KD is combined with Cutout or CutMix, each data selects the regularization method with a probability of 0.5. In other words, Self-KD is applied to 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) [27] and the area under the risk-coverage curve (AURC) [8] 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

[2]Note that the backward computation for the model at $(t-1)$-th epoch is not necessary.

Table 1: Top-1/top-5 error, NLL, ECE and AURC results on CIFAR-100 compared to other methods across popular architectures. The results are the average of three runs. Lower score indicates better performance and the best result is shown in boldface.

Model	Methods	Top-1 Error (%)	Top-5 Error (%)	NLL	ECE (%)	AURC ($\times 10^2$)
ResNet-50	Baseline	27.07	7.25	1.10	10.67	85.18
	+ Label Smoothing	26.97	7.87	1.07	2.08	93.18
	+ Self-KD	**24.95**	**5.98**	**0.87**	**1.70**	**72.42**
ResNet-101	Baseline	24.41	6.47	1.08	11.65	73.07
	+ Label Smoothing	24.80	7.37	1.06	2.96	97.90
	+ Self-KD	**22.77**	**5.40**	**0.81**	**1.62**	**62.71**
DenseNet-BC	Baseline	22.66	5.41	0.88	7.79	62.20
	+ Label Smoothing	22.75	6.21	0.96	5.01	72.65
	+ Self-KD	**21.26**	**4.62**	**0.77**	**3.16**	**57.08**
ResNeXt-29 (8×64d)	Baseline	18.65	4.47	0.74	**4.17**	44.27
	+ Label Smoothing	17.60	4.23	1.05	22.14	41.92
	+ Self-KD	**17.20**	**3.48**	**0.69**	6.03	**39.33**
PyramidNet-200	Baseline	16.80	3.69	0.73	8.04	36.95
	+ Label Smoothing	17.82	4.72	0.89	3.46	105.02
	+ Self-KD	**15.79**	**3.08**	**0.57**	**2.42**	**32.64**
	+ Cutout [4]	16.50	3.42	0.67	7.15	33.20
	+ Cutout + Self-KD	**14.82**	**2.87**	**0.54**	**2.02**	**29.57**
	+ CutMix [44]	15.62	3.38	0.68	8.16	34.60
	+ CutMix + Self-KD	**15.12**	**2.93**	**0.60**	**7.19**	**30.48**
	+ CutMix + SD [43]	14.07	2.38	0.51	3.96	28.65
	+ CutMix + SD + Self-KD	**13.61**	**2.28**	**0.49**	**1.72**	**26.48**

incorrect predictions can be well-separable based on confidence estimates. In these experiments, the maximum class probability is used as a confidence estimator.

Result. The results are summarized in Table 1. First, we observe that training with Self-KD performs better than baseline and LS in terms of classification accuracy across all architectures, e.g., an improvement of 1.45% and 0.4% from baseline and LS on ResNeXt, respectively. Compared with Cutout or CutMix on PyramidNet, Self-KD shows slightly lower accuracy while significantly improving the performances on confidence estimation, for example, it reduces ECE by 4.73% and 5.74% from Cutout and CutMix, respectively[3]. This performance improvements on confidence estimation are consistently observed across all metrics (i.e., NLL, ECE, and AURC) except for the single case, ECE on ResNeXt.

To show that Self-KD can be used in conjunction with other advanced regularization methods, we present the detailed experimental results on PyramidNet. We observe the top-1 error of 14.82% when Cutout is combined with Self-KD, which is 1.68% improvement of Cutout. When Self-KD, CutMix, and SD are utilized simultaneously, the top-1 error from the combination of CutMix and SD is reduced by 0.46%. In this setting, it is confirmed again that Self-KD provides a positive effect on confidence estimation: all metrics, NLL, ECE, and AURC, are improved by Self-KD. For example, employing Self-KD jointly shows 5.13% (Cutout+Self-KD), 0.97% (CutMix+Self-KD), 2.35% (CutMix+SD+Self-KD) of ECE improvements compared to Cutout, CutMix, CutMix+SD, respectively. These results demonstrate that current state-of-the-art regularization methods benefit from Self-KD in terms of not only classification accuracy, but also confidence estimation. From the previous study [2], 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.

To examine the effect of our method more precisely, we conduct experiments with a fixed value of α_t so that the effect of adjusting α_t is excluded. From the curves of NLL and top-1 error in Fig. 2, we

[3]The reliability diagrams are provided in the supplementary material.

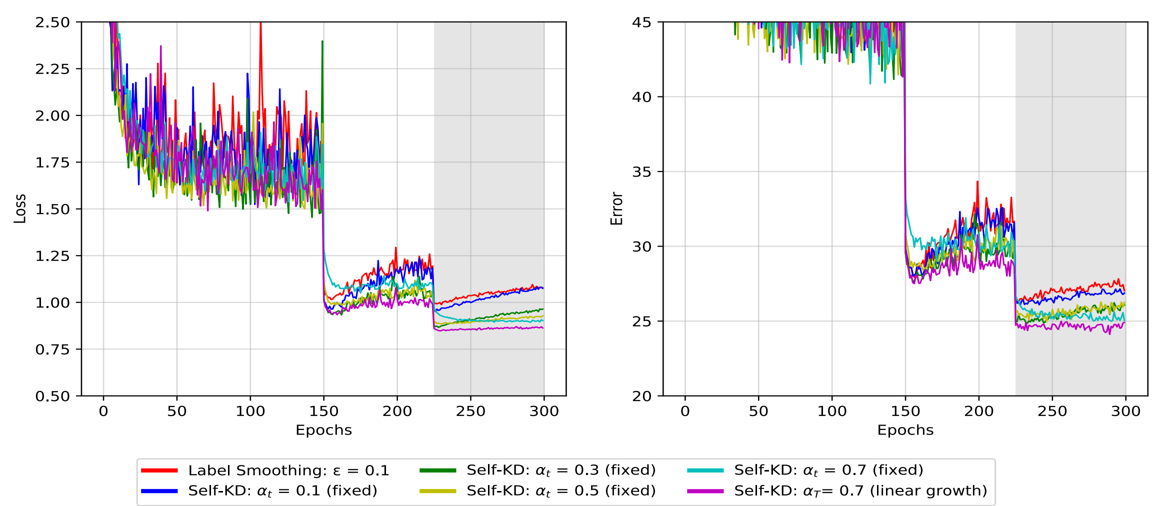


Figure 2: NLL (left) and top-1 error (right) curves with different α_t values from ResNet-50 on CIFAR-100. Linear growth with α_T = 0.7 achieves the lowest NLL and top-1 error.

Table 2: 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.

Model	Top-1 Error (%)	Top-5 Error (%)	NLL	ECE (%)	AURC ($\times 10^2$)
DenseNet-264* [16]	22.15	6.12	-	-	-
ResNeXt-101 (1x64d)* [41]	21.20	5.60	-	-	-
ResNet-152 Baseline	22.19	6.19	0.88	3.84	61.79
+ Label Smoothing	21.73	5.85	0.92	3.91	68.24
+ Self-KD	21.60	5.77	0.84	2.26	61.77
+ CutMix	21.09	**5.45**	0.82	2.16	60.06
+ CutMix + Self-KD	**20.96**	5.50	**0.81**	**0.59**	**58.67**

observe that Self-KD with a fixed $\alpha_t = 0.1$ shows lower NLL and top-1 error than LS with $\epsilon = 0.1$ (refer to the shaded area on the curves), and the performances are improved as a fixed α_t increases. To further investigate the effect of adjusting α_t, the curves from the linear growth strategy toward $\alpha_T = 0.7$ are also depicted. Compared to the curves from the fixed $\alpha_t = 0.7$, we conclude that the simplest approach, the linear growth, works surprisingly well for regularizing the model.

3.2 ImageNet Classification

In the case of a large-scale dataset like ImageNet [3], 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 learns from a large number of samples during a single epoch. Nevertheless, we observe that the model benefits from Self-KD even for such a large-scale dataset.

作者认为对于这种大尺度的数据集来说，上一个epoch训来的模型来作老师有点过时，他想用t-0.5这种

Experimental settings. As a baseline, we train Self-KD using ResNet (depth=152) with standard data augmentation schemes including random resize cropping, random horizontal flip, color jittering, and lighting for the task of image classification on ImageNet [44]. We train ResNet 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. In this experiment, we set the optimal α_T to 0.3 which showed the best top-1 error on the validation dataset.

Result. Table 2 shows performances evaluated by the metrics used in the previous section. Our method shows better accuracy than LS, achieving top-1 error of 21.60%, consistent to the results of experiments on CIFAR-100. Furthermore, Self-KD shows the best top-1 error, NLL, and ECE when it is combined with CutMix. It also even achieves much lower ECE of 0.59% which is 1.57% improvements from the model that uses CutMix only. We expect that the performance improvement can be greater if the knowledge from the recent past model is utilized, for example, the predictions

6

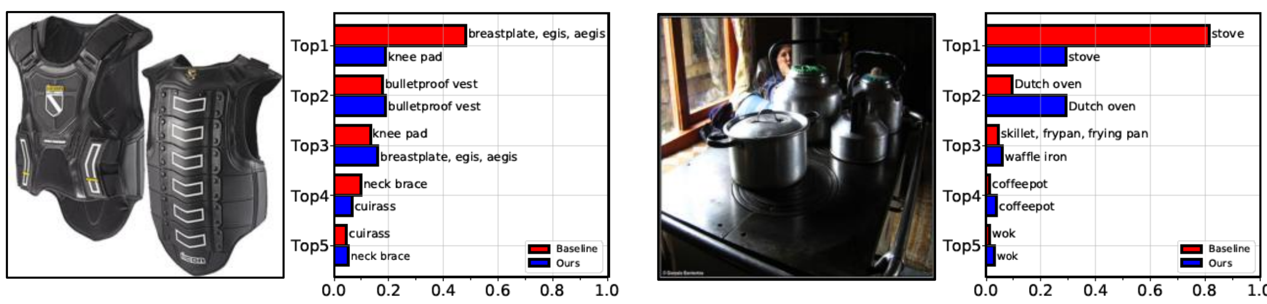


Figure 3: Predicted probabilities for samples in the validation dataset from baseline and Self-KD. The ground-truth labels of these images are "bulletproof vest" (left) and "stove" (right).

from the model at $(t-0.5)$-th epoch. Examples of how our Self-KD improves the quality of predicted probability are shown in Fig. 3. For the left image whose label is "bulletproof vest", both baseline and Self-KD produce an incorrect prediction. However, Self-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 right image contains multiple objects including "coffee pot" and "stove". Both baseline and Self-KD correctly classify this image, however, Self-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 Self-KD which acts as an effective and strong regularizer.

3.3 Object Detection

We also examine that other visual recognition tasks can benefit from Self-KD. For this, we perform the experiment on the task of object detection using PASCAL VOC [6] 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 testset, following to [30, 44]. As a baseline, Faster-RCNN [30] is considered, and the improvement of detection performance is examined by replacing the original VGG-16 [32] backbone network with a new one trained on ImageNet. We obtain three different backbones under the same training settings in the previous section: ResNet-152, ResNet-152 with LS, ResNet-152 with Self-KD. We then fine-tune Faster-RCNN 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.

Table 3: Effect of Self-KD as a pre-trained backbone network for Faster-RCNN. The mAP value is computed by averaging APs over classes.

Backbone	mAP (IoU > 0.5) (%)
ResNet-152	76.14
+ Label Smoothing	76.19
+ Self-KD	**77.31**

As shown in Table 3, ResNet-152 with Self-KD significantly improves the detection performance by 1.12% of the mean average precision (mAP) compared to ResNet-152 with LS. Note that this improvement is achieved by just replacing the backbone network. From this result, it is verified that training with Self-KD provides a strong backbone network, which provides discriminative representations that can be used for other visual recognition tasks. The detailed experimental results (e.g., APs over all classes) are presented in the supplementary material.

3.4 Machine Translation

Our Self-KD can be applied to any supervised learning tasks with hard targets. To verify the effectiveness of Self-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) [22], and Multi30k [5] from WMT16 [33]. IWSLT15 consists of 191K training sentence pairs[4], and 8,300 pairs of the training data is used for validation. We concatenate dev2010, dev2012, tst2010, tst2011, tst2012, tst2013 datasets for a testset. The original purpose of Multi30k is for multimodal learning, consisting of images and descriptions associated with them. For the

[4]The dataset can be downloaded from https://wit3.fbk.eu/mt.php?release=2015-01

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 [38] as our baseline model. All hyperparmeters involved in the architecture and training are set to those reported in [39]. 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 the maximum of 4,096 tokens per a mini-batch, and employ Adam optimizer [19] with $\beta_1 = 0.9$, $\beta_2 = 0.98$. As a metric, BLEU, commonly used one to evaluate the performance on machine translation, is used. The hyperparameter $\alpha_T = 0.7$ is determined through valiation. All experiments are conducted using PyTorch and fairseq[5] [26] toolkit.

The results are summarized in Table 4. Our Self-KD achieves the best BLEU scores on both datasets. Consistent with the results from image classification and object detection, Self-KD shows better performance than the baseline Transformer and that with LS. Especially, it achieves state-of-the-art BLEU scores of 30.0 and 36.2 on IWSLT15 English-to-German and German-to-English tasks, respectively.

Table 4: BLEU scores on Transformer with LS or Self-KD

Model	IWSLT15		Multi30k
	EN-DE	DE-EN	DE-EN
Transformer(Baseline)	28.5	34.6	29.0
+ Label Smoothing	29.3	35.6	29.3
+ Self-KD	**30.0**	**36.2**	**32.3**

4 Related Works

Recently, several training methods named self-knowledge distillation have been introduced in literature. They commonly use some knowledge extracted from a model itself to enhance the generalization performance of it. Hahn *et al.* [10] suggests the self-knowledge distillation method can be applied to NLP tasks. To obtain the knowledge to be distilled (i.e., soft targets), they use the Euclidean distance between two words in the embedding space and introduce a weight parameter to gradually transfer the knowledge during training. For computer vision tasks, Xu *et al.* [42] proposes a mechanism based on data distortion. Given an image, it generates two separate distorted images using random mirroring and cropping. Then, a model is trained by feeding these two images with the loss consisting of three terms: the maximum mean discrepancy, the Kullback-Leibler (KL) divergence and cross entropy, to make the model robust to data distortion. These methods have a limitation on applicability since they are designed specifically for a particular task. Yun *et al.* [45] present a method called class-wise self-knowledge distillation (CS-KD) which focuses on distilling knowledge between samples in the same class. For an input x, another data x' with the same label is randomly sampled, and the KL divergence between predictive distributions from them is minimized during training. They show that CS-KD reduces overconfident predictions and intra-class variations. Zhang *et al.* [47] propose a general training framework named self-distillation, which divides a network into several components and attaches auxiliary classifiers to them independently. These classifiers are trained using three supervisions including the hard targets, final softmax output, and activations in the deepest layer. Despite the improvement in performance, these methods only consider a multi-class classification task, i.e., applying those methods to other tasks such as machine translation is not straightforward.

Compared to the existing approaches named self-knowledge distillation, our method can be universally utilized for any supervised learning task as long as the hard targets are given for training.

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. From the experimental results conducted across diverse tasks, we observe that the proposed method is effective to improve the generalization capability of DNNs.

[5] Facebook AI Research Sequence-to-Sequence: https://github.com/pytorch/fairseq

8

References

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

[2] 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.

[3] 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.

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

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

[6] 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.

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

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

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

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

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

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

[13] 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.

[14] 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.

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

[16] 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.

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

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

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

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

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

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

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

[24] 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.

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

9

[26] 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.

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

[28] 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.

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

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

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

[41] 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.

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

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

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

A Evaluation Metrics

A.1 Image Classification

ECE Expected calibration error (ECE) [27] is a widely used metric for evaluating confidence calibration performance. To estimate the expected gap between accuracy and confidence, it partitions samples into total M bins, B_m for $m = 1, ..., M$, by confidence. Then, each bin B_m contains samples with confidence within $(\frac{m-1}{M}, \frac{m}{M}]$. With this binning, ECE is defined as follows,

$$ECE = \frac{1}{n}\sum_{m=1}^{M} |B_m| \times |\mathrm{Acc}(B_m) - \mathrm{Conf}(B_m)|$$

where n is the number of samples, $\mathrm{Acc}(B_m)$ represents accuracy of samples in B_m, and $\mathrm{Conf}(B_m)$ represents average confidence of samples in B_m. The lower value of ECE indicates that a model is well-calibrated.

AURC Area under risk-coverage curve (AURC) [8] measures how well predictions are ordered by confidence values. Given a classifier, we can define a selective classifier with a threshold which covers only samples with higher confidence than the threshold. Then, coverage can be defined as the proportion of covered samples (i.e., not rejected samples by the selective classifier) to the entire dataset. Risk is defined as an error rate computed by using the covered samples. Therefore, as coverage increases from 0 to 1, the risk approaches to the top-1 error on the whole data. AURC is defined as the area under the risk-coverage curve. If a model has a low AURC value, it means that correct and incorrect predictions from the model are well-separable by confidence values.

A.2 Machine Translation

BLEU BLEU (Bilingual Evaluation Understudy) is an algorithm for numerically measuring the quality of machine translation results. By using human translation as a reference, BLEU evaluates the quality of machine translation via two aspects. One is how many n-grams in the translated output of a model appears in the reference. If more n-grams appear in both machine translation and human translation, the quality of machine translation is considered as better. We set n to 4, which is generally used for the evaluation. Another aspect of BLEU is the length of machine translated sentence. If we evaluate the performance by using only n-grams, very short sentence with only few words in the reference will have nearly a perfect score. To prevent this, an additional term comparing the length of machine translation and human translation is considered in the calculation of BLEU.

B Image Classification

B.1 Datasets and Methods

Datasets CIFAR-100 is a dataset for multi-class image classification. It consists of 50K training images and 10K test images of 32×32 resolutions with 100 classes, and has the same number of images per class. The ImageNet is a large-scale dataset. It consists of 1.2M training images and 50K validation images of various resolutions with 1K classes. It contains some images that have multiple objects. In training, we use an input image that is resized to 256×256, and it is randomly cropped to have a size of 224×224. For inference, we resize an image as 256×256 and perform the center crop to have a 224×224 sized input.

Label Smoothing Szegedy et al. [36] proposes a method named label smoothing which improves the performance of deep learning models by adjusting one-hot targets to be soft targets. Soft targets $\mathbf{y}_{LS}$ are computed as a weighted sum of the hard targets $\mathbf{y}$ and the uniform distribution over classes, i.e.,

$$\mathbf{y}_{LS} = (1-\epsilon)\mathbf{y} + \frac{\epsilon}{K}$$

where ϵ is a smoothing parameter and K is the number of classes.

Cutout Cutout [4] is a simple regularization method designed for image classification. Motivated by dropout and image augmentation, Cutout generates a partially occluded version of input samples, which can be interpreted as an augmented data by applying the structured dropout to an input space. In detail, a square-shaped region with the predefined size is randomly selected on an input image, and that region is zeroed-out during training.

CutMix Yun et al. [44] suggests a method inspired by Cutout [4] and Mixup [46]. This method generates a new training sample $(\tilde{x}, \tilde{y})$ from two samples (x_a, y_a) and (x_b, y_b). From x_a, a rectangular region with bounding box coordinates (r_x, r_y, r_w, r_h) will be sampled as a patch. Then, the region of the same coordinates in x_b will be replaced by the patch to generate $\tilde{x}$. For the generated sample $\tilde{x}$, its target $\tilde{y}$ is defined as

$$\tilde{y} = \lambda y_a + (1 - \lambda) y_b.$$

ShakeDrop ShakeDrop [43] is a regularization technique designed for ResNet and its variants. This method gives regularization effect by replacing residual blocks to ShakeDrop blocks. Let an input x and an output of residual block $F(x)$, then the output of l-th ShakeDrop block $G(x)$ is defined as,

$$G(x) = \begin{cases} x + (b_l + \alpha - b_l\alpha)F(x), & \text{for the train-forward phase} \\ x + (b_l + \beta - b_l\beta)F(x), & \text{for the train-backward phase} \\ x + E[b_l + \beta - b_l\beta]F(x), & \text{for test phase} \end{cases}$$

where α, β are independent uniform random variables and b_l is a Bernoulli random variable with probability $P(b_l = 1) = p_l$, which is a parameter with linear decaying according to the block index l:

$$p_l = 1 - \frac{l}{L}(1 - P_L)$$

where L is the total number of building blocks and P_L is an initial parameter. In our experiments, we use $P_L = 0.5$ as suggested in [43].

B.2 Reliability Diagrams

The reliability diagram is a visualization tool to show how well confidence of a model is calibrated by plotting accuracy against confidence values. Fig. 4, 5, 6, and 7 show the reliability diagrams of the methods on CIFAR-100 and ImageNet. From these figures, we can observe that the model trained with Self-KD shows nearly perfect calibration.

B.3 Additional Samples on ImageNet

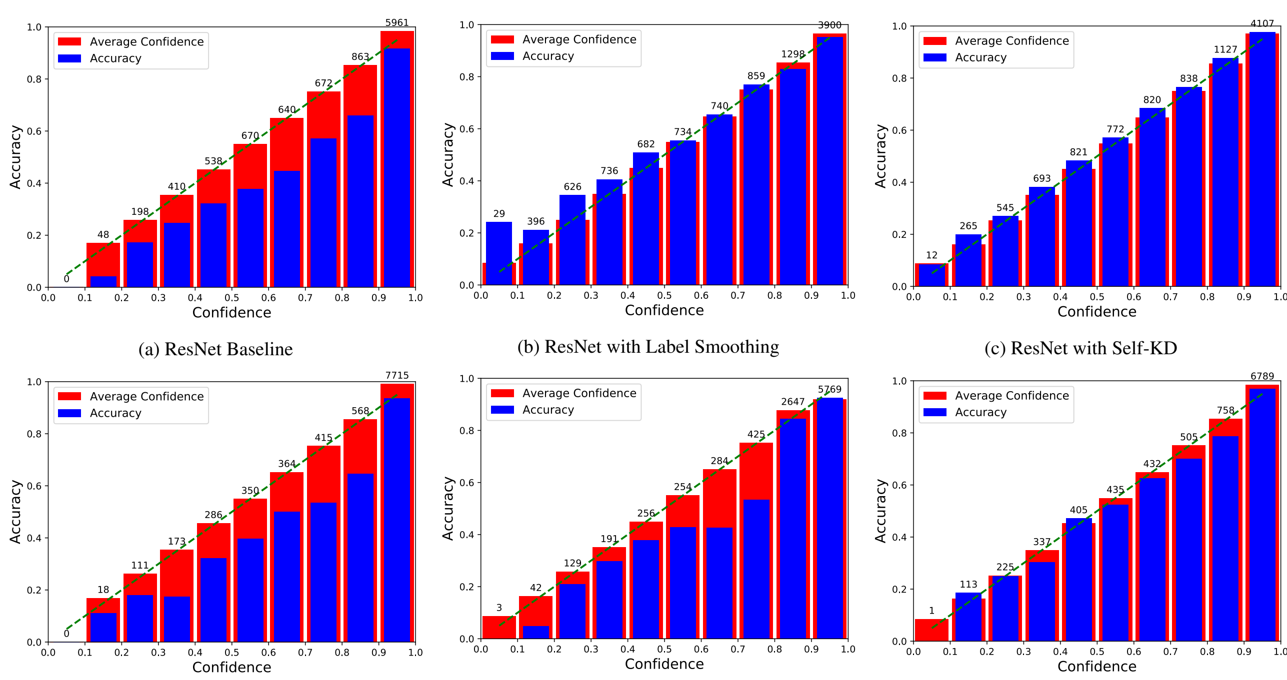
In Fig. 8, additional samples from ImageNet validation dataset and their predicted probabilities are presented. From these samples, we observe that Self-KD provides better outputs in the sense of human interpretation.

C Object Detection

Table 5 shows the values of average precision (AP) over all classes. Self-KD shows higher AP values than baseline and label smoothing for 12 classes out of 20 classes.

Table 5: APs over all classes on PASCAL VOC 2007 testset. The best result is in bold.

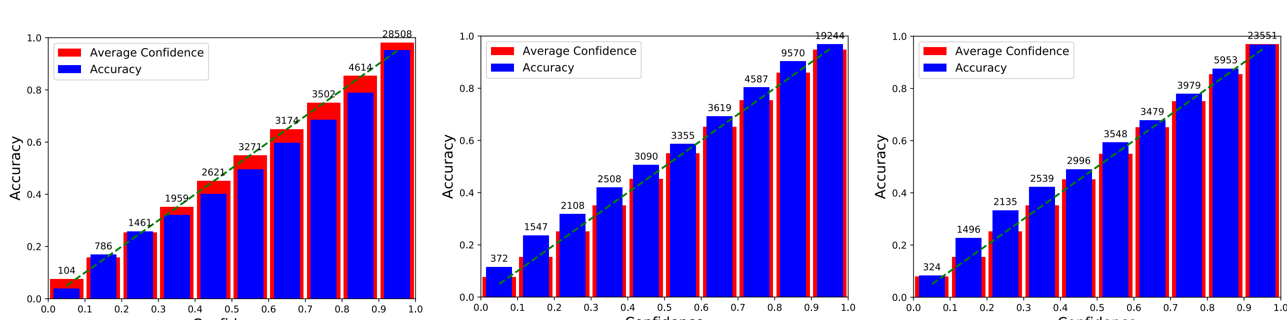
Method	Average Precision										mAP
	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	
Baseline	76.48	82.96	76.08	**68.18**	62.53	**85.34**	85.40	88.11	57.48	81.55	76.14
+Label smoothing	**78.83**	80.80	76.79	67.50	62.15	83.43	85.64	**88.44**	60.76	84.69	76.19
+Self-KD	78.35	**85.34**	**78.39**	66.11	**63.03**	84.68	**87.02**	86.49	**61.86**	**84.97**	**77.31**
	Dining Table	Dog	Horse	Mortor Bike	Person	Potted Plant	Sheep	Sofa	Train	TV Monitor	
Baseline	70.16	85.81	**85.69**	78.59	78.71	**47.58**	76.91	75.37	**84.71**	75.24	
+Label smoothing	**73.69**	85.96	84.53	78.86	78.60	44.13	79.56	76.27	81.65	71.63	
+Self-KD	73.23	**86.48**	85.10	**81.49**	**78.93**	45.32	**81.90**	**77.37**	84.42	**75.43**	



(a) ResNet Baseline (b) ResNet with Label Smoothing (c) ResNet with Self-KD

(d) PyramidNet Baseline (e) PyramidNet with Label Smoothing (f) PyramidNet with Self-KD

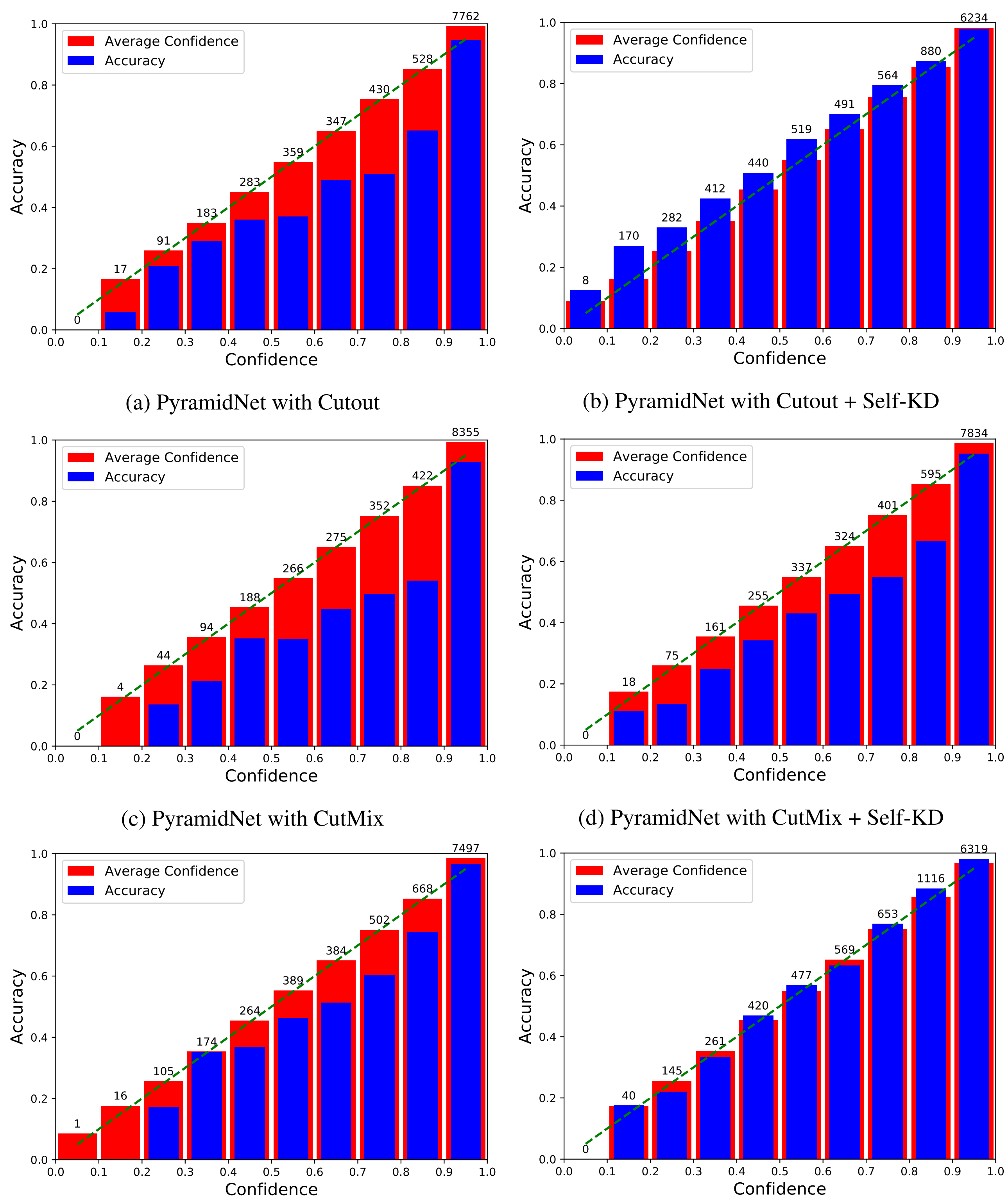
Figure 4: Reliability diagrams with ResNet-50 and PyramidNet on CIFAR-100



(a) ResNet Baseline (b) ResNet with Label Smoothing (c) ResNet with Self-KD

Figure 5: Reliability diagrams with ResNet-152 on ImageNet

13



(a) PyramidNet with Cutout

(b) PyramidNet with Cutout + Self-KD

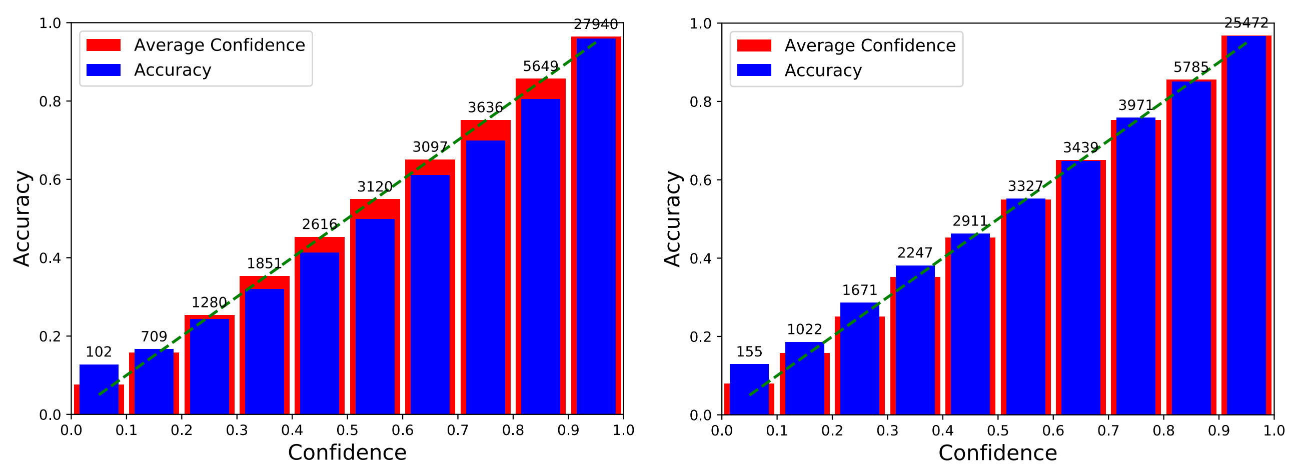
(c) PyramidNet with CutMix

(d) PyramidNet with CutMix + Self-KD

(e) PyramidNet with CutMix + ShakeDrop

(f) PyramidNet with CutMix + ShakeDrop + Self-KD

Figure 6: Reliability diagram for advanced regularization methods with PyramidNet on CIFAR-100. Self-KD provides additional benefits to existing methods in terms of calibration.



(a) ResNet with CutMix (b) ResNet with CutMix + Self-KD

Figure 7: Reliability diagrams for advanced regularization methods with ResNet-152 on ImageNet. Self-KD provides additional benefits to existing methods in terms of calibration.

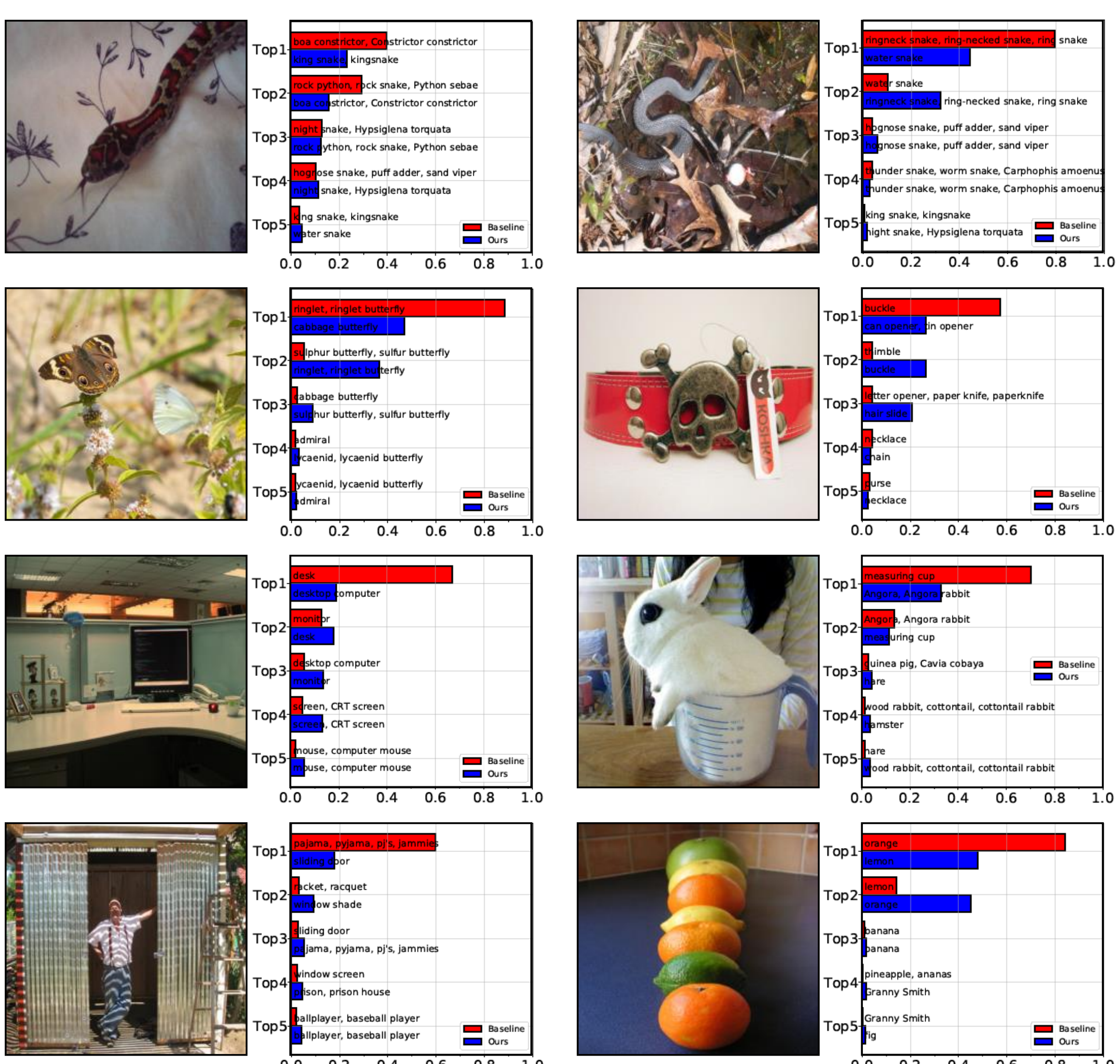


Figure 8: Predicted probabilities for sample images from baseline and Self-KD. From the top left, the ground-truth labels of these images are "king snake", "water snake", "cabbage butterfly", "buckle", "desk", "measuring cup", "sliding door" and "orange", respectively.

15