Self-Distillation from the Last Mini-Batch for Consistency Regularization

Yiqing Shen[1*†], Liwu Xu[2*], Yuzhe Yang[2], Yaqian Li[2‡], Yandong Guo[2]
[1]Shanghai Jiao Tong University, [2]OPPO Research Institute
shenyq@sjtu.edu.cn, {xuliwu, liyaqian}@oppo.com, ipllewis@gmail.com, yandong.guo@live.com

Abstract

Knowledge distillation (KD) shows a bright promise as a powerful regularization strategy to boost generalization ability by leveraging learned sample-level soft targets. Yet, employing a complex pre-trained teacher network or an ensemble of peer students in existing KD is both time-consuming and computationally costly. Various self KD methods have been proposed to achieve higher distillation efficiency. However, they either require extra network architecture modification or are difficult to parallelize. To cope with these challenges, we propose an efficient and reliable self-distillation framework, named Self-Distillation from Last Mini-Batch (DLB). Specifically, we rearrange the sequential sampling by constraining half of each mini-batch coinciding with the previous iteration. Meanwhile, the rest half will coincide with the upcoming iteration. Afterwards, the former half mini-batch distills on-the-fly soft targets generated in the previous iteration. Our proposed mechanism guides the training stability and consistency, resulting in robustness to label noise. Moreover, our method is easy to implement, without taking up extra run-time memory or requiring model structure modification. Experimental results on three classification benchmarks illustrate that our approach can consistently outperform state-of-the-art self-distillation approaches with different network architectures. Additionally, our method shows strong compatibility with augmentation strategies by gaining additional performance improvement. The code is available at https://github.com/Meta-knowledge-Lab/DLB.

1. Introduction

Knowledge Distillation (KD), first introduced by Bucilua et al. [2], was later popularized by Hinton et al. [10]. Numerous previous researches have demonstrated the success of KD in various learning tasks to boost the generalization ability. For example, in the case of network compression, the two-stage offline KD is widely used to transfer dark knowledge from a cumbersome pre-trained model to a light student model that learns from teachers' intermediate feature maps [21], logits [10], attention maps [40], or auxiliary outputs [43]. However, training a high-capacity teacher network heavily relies on large computational sources and run-in memory. To alleviate the time-consuming preparation of static teachers, online distillation is introduced [44], where an ensemble of peer students learns mutually from each other. Online KD achieves equivalent performance improvement, compared with offline KD, with higher computational efficiency. Thus, this line is subsequently extended by many following works to a more capable self-ensemble teacher [3,8,14,33]. Other applications of KD include semi-supervised learning, domain adaptation, transfer learning and etc [18, 26, 28]. The main scope of this paper focuses on the KD paradigm itself.

Conventional KD approaches, both online and offline, have achieved satisfying empirical performance [24]. Yet, existing KD approaches suffer from an obstacle in the low knowledge transferring efficiency [35]. Additionally, high computation and run-in memory costs restrict their deployment onto the end devices, such as mobile phones, digital cameras [4]. To cope with these limitations, self-knowledge distillation has received increasing popularity, which enables a student model to distill knowledge from itself. The absence of a complex pre-trained teacher and an ensemble of peer students in self-KD contributes to a marginal improvement in the training efficiency.

One popular formulation of self KD, such as Be Your Own Teacher (BYOT), requires heavy network architecture modifications, which largely increases their difficulty to generalize onto various network structures [19,32,43]. In another line, history information, including previous training logits or model snapshots, is utilized to construct a virtual teacher for extra supervision signals as self distillation. Initially, born again networks (BAN) sequentially distill networks with identical parameters as its last generation [7]. An advancement achieved by snapshot distillation is to take the secondary information from the prior mini-generation i.e. a couple of epochs, within one generation [36]. This virtual teacher update frequency is further

[*]Equal Contribution. [‡]Corresponding Author.
[†]This work was done during the internship at OPPO Research Institute.

Table 1. A comparison with the state-of-the-arts in terms of computational cost and smoothed granularity. We compare our method with label smoothing regularization [27], teacher-free knowledge distillation (Tf-KD$_{self}$, Tf-KD$_{reg}$) [37], class-wise self-knowledge distillation (CS-KD) [39], progressive self-knowledge distillation (PS-KD) [12], memory-replay knowledge distillation (Mr-KD) [30], data-distortion guided self-knowledge distillation (DDGSD) [35], be your own teacher (BYOT) [43].

Characteristic	LSR	Tf-KD$_{self}$	Tf-KD$_{reg}$	CS-KD	PS-KD	Mr-KD	DDGSD	BYOT	**Ours**
Sample-level smoothing	✗	✓	✓	✓	✓	✓	✓	✓	✓
No pre-trained teacher	✓	✗	✓	✓	✓	✓	✓	✓	✓
No Architecture modification	✓	✓	✓	✓	✓	✓	✓	✗	✓
Forward times per batch	1	2	1	2	2	≥ 2	2	1	1
Backward times per batch	1	1	1	2	1	1	2	$\geq$2	1
Number of involved networks	1	2	1	1	2	1	1	1	1
Label update frequency	-	epoch	-	-	epoch	epoch	-	-	batch

improved to epoch-level in progressive self-knowledge distillation [12] and learning with retrospection [5]. Yet, existing self KD methods have the following setbacks to be tackled. Firstly, the most instant information from the last iteration is discarded. Moreover, storing a snapshot of past models consumes an extra run-in memory cost, and subsequently increases the difficulty to parallelize [36]. Finally, computation of the gradient in each time backward prorogation is associated with twice the forward process on each batch of data, resulting in a computational redundancy and low computational efficiency.

To address these challenges in existing self KD methods, we propose a simple but efficient self distillation approach, named as Self-Distillation from Last Mini-Batch (DLB). Compared with existing self KD approaches, DLB is computationally efficient and saves the run-in memory by only storing the soft targets generated in the last mini-batch backup, resulting in its simplicity in deployment and parallelization. Every forward process of data instances is associated with a once backpropagation process, mitigating the computational redundancy. Major differences compared with the state-of-the-arts are summarized in Table 1. DLB produces on-the-fly sample-level smoothed labels for self-distillation. Leveraging the soft predictions from the last iteration, our method provides the most instant distillation for each training sample. The success of DLB is attributed to the distillation from the most immediate historically generated soft targets to impose training consistency and stability. To be more specific, the target network plays a dual role as teacher and student in each mini-batch during the training stage. As a teacher, it provides soft targets to regularize itself in the next iteration. As a student, it distills smoothed labels generated from the last iteration and minimizes the supervision learning objective e.g. the cross-entropy loss.

We empirically illustrate the comprehensive effectiveness of our methods on three benchmark datasets, namely CIFAR-10 [13], CIFAR-100 [13], TinyImageNet. Our approach is both task-agnostic and model-agnostic i.e. with no requirement of model topological modifications. We select six representative backbone CNNs for evaluations, including ResNet-18, ResNet-110 [9], VGG-16, VGG-19 [25], DenseNet [11], WideResNet [41]. Experimental results demonstrate that our DLB can consistently improve the generalization ability. We also test the robustness of DLB on corrupted data. Training consistency and stability imposed by DLB on corrupted data results in higher generalization ability.

The major contributions are three-fold.

- We propose a simple but efficient consistency regularization scheme based on self-knowledge distillation, named as DLB. With no network architecture modifications, our method requires very few additional computation costs as well as the run-time memory to implement. Utilizing the latest update from the last iteration, our DLB is easy to implement for parallelization. Notably, the proposed method is also both model-agnostic and task-agnostic.

- Comprehensive experimental results on three popular classification benchmarks illustrate consistent generalization improvements on different models. We also empirically demonstrate the compatibility of DLB with various augmentation strategies.

- We systemically analyze the impact of our method on the training dynamics. Concretely, the success of its regularization effect is attributed to guidance towards training consistency, by leveraging on-the-fly sample-level smooth labels. The consistency effect is further amplified under label corruption settings, showing strong robustness to label noise. These empirical finds may shed light on a new direction to understand the effect of knowledge distillation.

2. Related Works

Knowledge distillations. Knowledge distillation (KD) targets to transfer 'knowledge', such as logits, or intermediate feature maps, from a high-capability teacher model

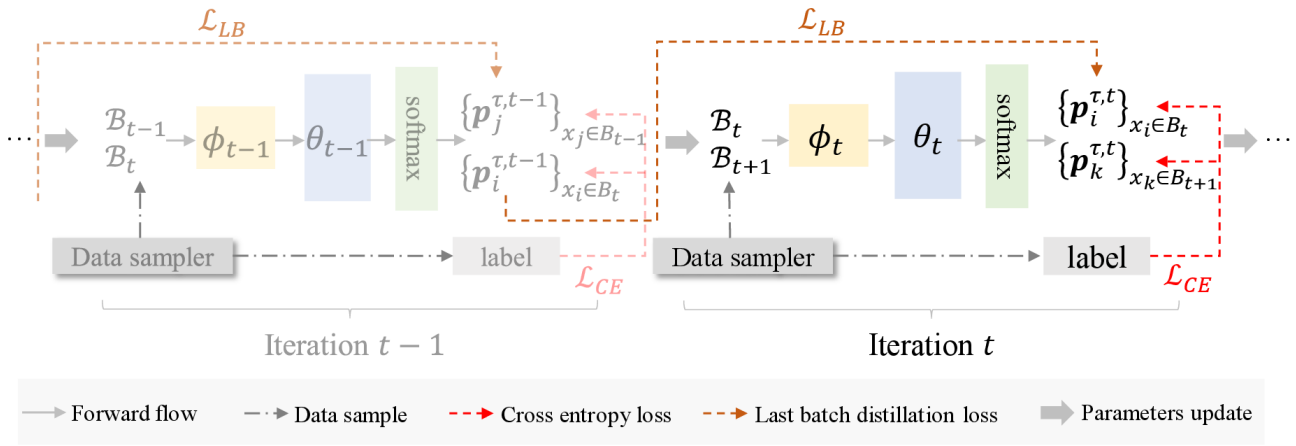


Figure 1. The overall architecture of our DLB. We write $\mathcal{B}_t$, ϕ_t, θ_t for a mini-batch of data samples, random augmentation and the trainable parameters indexed in the t^{th} iteration.

to a lightweight student network [2, 10]. Despite its competitive performance improvement to generalization, pre-training a complex teacher model requires extra training time and computational cost. Another way to form an economic distillation is called mutual learning, also known as online distillation, where an ensemble of students learn mutually from each other [44]. This idea was popularized by many following works [3, 14, 33]. But, the optimization in peer learning involves multiple numbers of networks, which requires extra memory to store all parameters.

Self Knowledge Distillation. To enhance efficiency and effectiveness in knowledge transferring, self knowledge distillation (SKD) is proposed to utilize knowledge from itself, without the involvement of an extra networks [31]. There are three popular ways to construct a SKD model i.e., 1) data distortion based self-distillation [15, 35], 2) use history information as a virtual teacher, 3) distilling across auxiliary heads [17, 43]. However, the first one marginally relies on the augmentation efficiency. The second one misses the latest update from the last mini-batch. And the last kind requires heavy network architecture modifications, which increase its difficulty for deployment.

Distillations as regularization. KD is extensively used in many tasks, such as model compression, semi-supervised learning, domain adaptation and etc [18, 26, 28]. However, theoretical analysis of the success of KD remains a big challenge. Recently, Yuan et al. attributed the success of KD to its regularization effect as providing sample-level soft targets from the LSR perspective [37]. It reveals the large promise of applying KD to the regularization domain. In this line, class-wise self-knowledge distillation (CS-KD) is designed by evacuating a consistency between predictions of two batches of samples identified with the same category [39]. Progressive self-knowledge distillation (PS-KD), more similar to our work, progressively distills past knowledge from the last epoch to soften hard targets in the current epoch [12]. Memory-replay knowledge distillation (Mr-KD) extends PS-KD by storing a series of abandoned network backups for distillation [30]. However, implementing PS-KD or Mr-KD both requires extra GPU memory to store the historical model parameters or the whole past predictions on the disk. The previous strategy is computationally costly for large models such as deep WRN [41], while the latter one is inefficient in training large datasets such as ImageNet [23]. The above-mentioned drawbacks result in low training efficiency, as well as the implementation difficulty on end devices such as mobile phones, digital cameras, etc [4], which restrict their applications for regularization. On the other hand, much recent information from the last several mini-batches is absent in these methods. To cope with these shortages, we propose a novel self-distillation framework named DLB, which will be elaborated on in the following section.

3. Methods

3.1. Preliminary

In this work, we focus on a supervised classification task as case study. For a clear notation, we write a K-classes labelled dataset as $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where N is the total number of training instances. In every mini-batch, a batch of n samples $\mathcal{B} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \subseteq \mathcal{D}$ is augmented by data distortion ϕ to derive the distorted images $\mathcal{B}^{\phi} = \{(\phi(\mathbf{x}_i), y_i)\}_{i=1}^{n}$. Afterwards, they are fed into target neural network h_θ to optimize the cross-entropy loss function, defined as follows

$$\mathcal{L}_{CE} = \frac{1}{n}\sum_{i=1}^{n} H\left(y_i, \mathbf{p}_i\right). \tag{1}$$

Formally, the predictive distribution $\mathbf{p}_i = (p_i(1), \cdots, p_i(K))$ in a softmax classifier for class $k \in [K]$ is formulated as

$$p_i(k) = \frac{\exp(f_k(\mathbf{x}_i;\theta)/\tau)}{\sum_{j=1}^{K}\exp(f_j(\mathbf{x}_i;\theta)/\tau)}, \tag{2}$$

where f_k writes for the k^{th} component of the logits from the backbone encoder parameterized by θ. Temperature τ is usually set to 1 in Eq. (2). To improve the generalization ability, vanilla knowledge distillation [10] transfers pre-trained teacher's knowledge by optimizing an additional Kullback-Leibler (KL) divergence loss between the softened outputs from teacher and student in every mini-batch i.e.

$$\mathcal{L}_{KD} = \frac{1}{n}\sum_{i=1}^{n} \tau^2 \cdot D_{KL}\left(\widetilde{\mathbf{p}}_i^{\tau} \| \mathbf{p}_i^{\tau}\right), \tag{3}$$

where $\mathbf{p}_i^{\tau}$, $\widetilde{\mathbf{p}}_i^{\tau}$ are the soften predictions, parameterized by temperature τ, from student and teacher respectively. A higher temperature results in a more uniform distribution, leading to a similar regularization effect as label smoothing [27, 37]. Compared with previous works where a complex network is pre-trained to generate $\widetilde{\mathbf{p}}_i^{\tau}$, our work uses historic information from the last batch to efficiently generate $\widetilde{\mathbf{p}}_i^{\tau}$ as a more instant smoothed labels for regularization.

3.2. Self-Distillation from Last Batch

The overall training process for the proposed self-distillation is visualized in Figure 1. Instead of adopting a complex pre-trained teacher model to provide sample-wise smoothed labels, our proposed framework utilizes the backup information from the last mini-batch to generate soft targets. It results in a regularization towards training consistency. For a clear notation, we denote the original batch of data sampled in the t^{th} iteration as $\mathcal{B}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^n$, and the network parameters as θ_t. Formally, we substitute the $\widetilde{\mathbf{p}}_i^{\tau}$ in Eq. (3) by the soften labels $\mathbf{p}_i^{\tau,t-1}$ generated by the identical network at $t-1^{\text{th}}$ iteration i.e. f parameterized by θ_{t-1}. Consequently, we introduce an extra last-batch consistency regularization loss to DLB as follows:

$$\mathcal{L}_{LB} = \frac{1}{n}\sum_{i=1}^{n} \tau^2 \cdot D_{KL}\left(\mathbf{p}_i^{\tau,t-1} \| \mathbf{p}_i^{\tau,t}\right). \tag{4}$$

Rather than storing the whole θ_{t-1} in the t^{th} iterations as designed in previous works [12], which is run-time memory consuming, we complete the computation of all $\mathbf{p}_i^{\tau,t-1}$ in $t-1^{\text{th}}$ iteration. We employ a data sampler to obtain $\mathcal{B}_t$ and $\mathcal{B}_{t-1}$ simultaneously at the $t-1^{\text{th}}$ iteration for implementation. Both the predictions from $\mathcal{B}_{t-1}$ and $\mathcal{B}_t$ in $t-1^{\text{th}}$ iteration update the $\mathcal{L}_{CE}$. Whereas predictions from $\mathcal{B}_t$ are smoothed by temperature τ and then stored for regularization in t^{th} iteration. Storage of a batch of smoothed labels requires very few extra memory cost, which is thus more efficient. Conclusively, the overall loss function is formulated by

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \cdot \mathcal{L}_{LB}, \tag{5}$$

where α is the coefficient to balance two terms. In short, we constrain half of each mini-batch coinciding with the previous iteration, and the rest half with the upcoming iteration. Afterwards, the former half mini-batch distills on-the-fly soft targets generated in the previous iteration. The overall training process is summarized in Algorithm 1.

Algorithm 1 Pseudo code for DLB.

Input: balancing coefficient α
Input: distillation temperature τ
Require: data_loader samples batches as in Figure 1

```
last_logits = None # initialization
for (x,gt_labels) in data_loader do
    n̂ = gt_labels.size(0) # batch size
    logits = model.forward(x)
    loss = CELoss(logits, gt_labels)
    if last_logits != None then
        soft_targets = Softmax(last_logits/τ)
        pred = Softmax(logits[:n̂//2]/τ)
        loss += α*LBLoss(pred, soft_targets)*τ²
    end if
    loss.backward() # update parameters
    last_logits = logits[n̂//2:].detach() # no gradient
end for
```

4. Experiments

4.1. Datasets and Settings

Datasets. We employed three multi-class classification benchmark datasets for comprehensive performance evaluations. The **CIFAR-10/CIFAR-100** contain a total number of 60,000 RGB natural images of 32×32 pixels from 10/100 classes [13]. Each class includes 5,000/500 training samples and 1,000/100 testing samples. We followed the widely-used pre-processing from previous works [9,41]. More precisely, training samples were normalized by deviation and padded 4 pixels with zero-value on each side. A random crop of 32×32 region was generated from the padded image or its horizontal flip. The **TinyImageNet** is a subset of ILSVRC-2012, made up of 200 classes. Each class includes 500 training samples and 50 testing samples, scaled at 64×64. All training images were randomly cropped and resized to 32×32 after the normalization. The test images were only normalized.

Table 2. Performance comparison with label smoothing regularization and state-of-the-art self-distillation methods on CIFAR-10, CIFAR-100 and TinyImageNet in terms of average top-1 error rate (%). We calculated the average and deviation over three runs. The best and second best performance were highlighted in **Red** and Green respectively.

Dataset	Methods	Vgg-16	Vgg-19	ResNet-32	ResNet-110	WRN20-8	DenseNet-40-12
CIFAR-10	Baseline	6.03$_{\pm 0.22}$	6.04$_{\pm 0.10}$	6.54$_{\pm 0.10}$	5.21$_{\pm 0.24}$	5.47$_{\pm 0.08}$	7.09$_{\pm 0.28}$
	LSR	5.91$_{\pm 0.40}$	6.05$_{\pm 0.06}$	6.73$_{\pm 0.17}$	5.60$_{\pm 0.17}$	4.76$_{\pm 0.06}$	7.47$_{\pm 0.27}$
	Tf-KD$_{self}$	5.92$_{\pm 0.15}$	5.91$_{\pm 0.01}$	6.32$_{\pm 0.01}$	4.92$_{\pm 0.08}$	5.33$_{\pm 0.13}$	7.03$_{\pm 0.14}$
	Tf-KD$_{reg}$	6.12$_{\pm 0.07}$	6.25$_{\pm 0.18}$	6.60$_{\pm 0.05}$	5.48$_{\pm 0.18}$	4.79$_{\pm 0.01}$	7.38$_{\pm 0.16}$
	CS-KD	6.22$_{\pm 0.20}$	6.38$_{\pm 0.11}$	6.88$_{\pm 0.15}$	6.12$_{\pm 0.05}$	4.89$_{\pm 0.19}$	7.85$_{\pm 0.17}$
	PS-KD	5.90$_{\pm 0.04}$	6.07$_{\pm 0.57}$	5.96$_{\pm 0.06}$	5.09$_{\pm 0.68}$	4.99$_{\pm 0.01}$	6.77$_{\pm 0.28}$
	DLB	**5.38**$_{\pm 0.01}$ (0.65 ↑)	**5.58**$_{\pm 0.08}$ (0.46 ↑)	**5.85**$_{\pm 0.02}$ (0.70 ↑)	**4.85**$_{\pm 0.11}$ (0.37 ↑)	**4.46**$_{\pm 0.01}$ (1.01 ↑)	**6.57**$_{\pm 0.02}$ (0.53 ↑)
CIFAR-100	Baseline	26.37$_{\pm 0.19}$	27.16$_{\pm 0.42}$	28.26$_{\pm 0.18}$	23.64$_{\pm 0.25}$	22.42$_{\pm 0.23}$	28.31$_{\pm 0.35}$
	LSR	25.81$_{\pm 0.02}$	26.75$_{\pm 0.45}$	28.21$_{\pm 0.13}$	23.32$_{\pm 0.16}$	22.17$_{\pm 0.01}$	29.07$_{\pm 0.01}$
	Tf-KD$_{self}$	25.94$_{\pm 0.11}$	27.46$_{\pm 0.82}$	26.09$_{\pm 0.24}$	27.02$_{\pm 0.01}$	21.88$_{\pm 0.34}$	28.40$_{\pm 0.09}$
	Tf-KD$_{reg}$	25.85$_{\pm 0.30}$	26.82$_{\pm 0.59}$	28.29$_{\pm 0.07}$	23.54$_{\pm 0.04}$	22.28$_{\pm 0.05}$	28.92$_{\pm 0.04}$
	CS-KD	25.81$_{\pm 0.43}$	26.65$_{\pm 0.49}$	29.21$_{\pm 0.21}$	23.41$_{\pm 0.28}$	21.75$_{\pm 0.26}$	29.65$_{\pm 0.51}$
	PS-KD	25.95$_{\pm 0.74}$	26.36$_{\pm 0.76}$	27.49$_{\pm 0.75}$	22.85$_{\pm 0.65}$	21.26$_{\pm 0.11}$	28.48$_{\pm 0.53}$
	DLB	**23.88**$_{\pm 0.06}$ (2.50 ↑)	**24.53**$_{\pm 0.13}$ (2.63 ↑)	**26.00**$_{\pm 0.03}$ (2.26 ↑)	**21.82**$_{\pm 0.19}$ (1.83 ↑)	**20.79**$_{\pm 0.33}$ (1.63 ↑)	**27.48**$_{\pm 0.21}$ (0.83 ↑)
TinyImageNet	Baseline	48.83$_{\pm 0.33}$	50.02$_{\pm 0.12}$	50.38$_{\pm 0.38}$	43.20$_{\pm 0.56}$	43.72 $_{\pm 0.28}$	50.94$_{\pm 0.56}$
	LSR	47.95$_{\pm 0.32}$	48.86$_{\pm 0.78}$	52.75$_{\pm 0.69}$	42.18$_{\pm 0.57}$	43.51$_{\pm 0.13}$	51.01$_{\pm 0.16}$
	Tf-KD$_{self}$	46.76$_{\pm 0.10}$	47.25$_{\pm 0.06}$	49.48$_{\pm 0.16}$	41.31$_{\pm 0.01}$	41.13$_{\pm 0.11}$	50.78$_{\pm 0.01}$
	Tf-KD$_{reg}$	48.31$_{\pm 0.06}$	49.54$_{\pm 0.16}$	50.97$_{\pm 0.02}$	41.88$_{\pm 0.91}$	42.85$_{\pm 0.07}$	51.47$_{\pm 0.10}$
	CS-KD	46.95$_{\pm 0.15}$	47.89$_{\pm 0.13}$	53.99$_{\pm 0.03}$	43.06$_{\pm 0.22}$	42.04$_{\pm 0.42}$	54.93$_{\pm 0.25}$
	PS-KD	48.39$_{\pm 0.23}$	49.77$_{\pm 0.23}$	50.28$_{\pm 0.17}$	42.22$_{\pm 0.72}$	43.37$_{\pm 0.01}$	51.57$_{\pm 0.05}$
	DLB	**45.66**$_{\pm 0.01}$ (3.17 ↑)	**46.68**$_{\pm 0.09}$ (3.35 ↑)	**48.66**$_{\pm 0.07}$ (1.72 ↑)	**40.39**$_{\pm 0.01}$ (2.81 ↑)	**41.03**$_{\pm 0.02}$ (2.69 ↑)	**50.13**$_{\pm 0.01}$ (0.81 ↑)

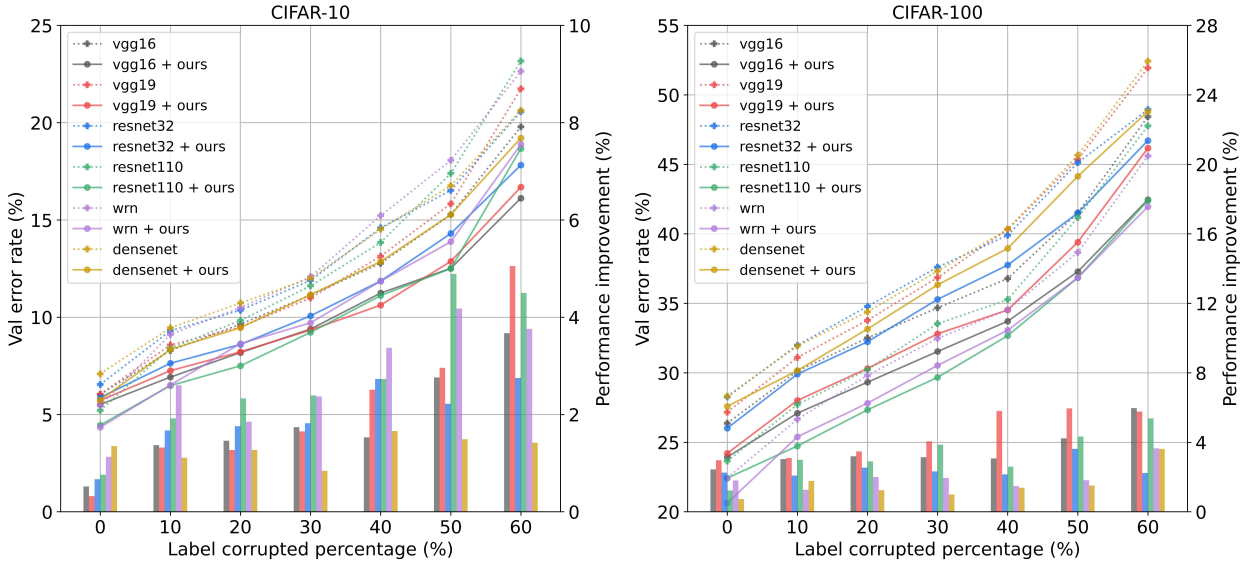
Implementations. All experiments were performed on one NVIDIA Tesla V100 GPU with 32Gb memory. The proposed DLB and compared methods were all implemented on Pytorch 1.6.0 in Python 3.7.0 environment. The hyper-parameters for the training scheme followed a consistent setting for a fair comparison. **CIFAR-100/10**: We followed the settings in previous works [29, 34]. Specifically, every backbone network was trained for 240 epochs with a batch size of 64. The initial learning rate was set to 0.05, and decayed by a factor of 10% at 150$^{\text{th}}$, 180$^{\text{th}}$ and 210$^{\text{th}}$ epoch. We employed a stochastic gradient descent (SGD) optimizer with 0.9 Nesterov momentum, where the weight decay rate was set to 5×10^{-4}. **TinyImageNet**: We also followed the settings in previous works [42]. Concretely, we set the maximal epoch number to 200, the batch size to 128. The initial learning rate was set to 0.2 with a decayed factor of 10% at 100$^{\text{th}}$, 150$^{\text{th}}$ epoch. The weight decay rate and the momentum in the SGD optimizer were set to 1×10^{-4} and 0.9 respectively. We fixed the temperature τ at 3 and coefficient α in DLB to 1, where the dependence of hyper-parameters is explored in the next subsection. The temperature setting in DLB followed suggests from previous work [44], and we did not manually tune it for our approach. We used the top-1 error rate (%) on the test set as the evaluation metric. For reproducibility, we fixed the random seed at 95 in all experiments. We also measured the average and the associated standard deviation over three runs. Importantly, for a fair comparison in terms of computation cost, the proposed method (DLB) was evaluated on half of the total training interactions/epochs than the compared approaches. Although our method comes to a doubled batch size by duplicating half of the last mini-batch, it brings no extra computation cost with halving total training iterations.

Backbone architectures. We employed six representative architectures [3, 33] for evaluation, namely Vgg-16, Vgg-19 [25], ResNet-32, ResNet-110 [9], WRN20-8 [41], DenseNet-40-12 [11].

Compared methods. Hard labels were utilized in the baseline to train the target network directly. We also compared the proposed method with label smoothing regulariza-

Table 3. Performance compatibility with augmentation-based regularization methods including CutOut [6], CutMix [38] and DDGSD [35] on CIFAR-10/100. We calculated the average top-1 error rate (%), standard deviation of three runs, written in the form of 'avg ± std'. The best result in each category was highlighted in **boldface**.

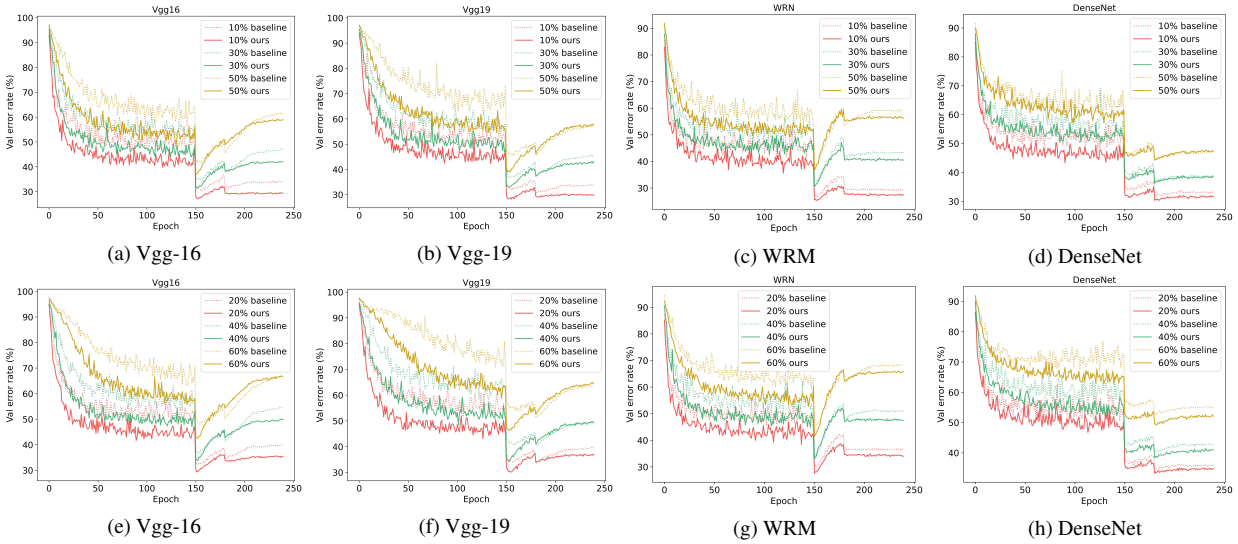
Dataset	Methods	Vgg-16	Vgg-19	ResNet-32	ResNet-110	WRN20-8	DenseNet-40-12
C10	Baseline	6.03±0.22	6.04±0.10	6.54±0.10	5.21±0.24	5.47±0.08	7.09±0.28
	+ DLB	**5.38**±0.01	**5.58**±0.08	**5.85**±0.02	**4.85**±0.11	**4.46**±0.01	**6.57**±0.02
	+ CutOut	4.93±0.04	5.10±0.05	5.91±0.12	4.64±0.14	4.88±0.11	6.68±0.11
	+ CutOut + DLB	**4.48**±0.03	**4.46**±0.40	**5.18**±0.05	**3.78**±0.09	**4.12**±0.09	**5.54**±0.04
	+ CutMix	5.29±0.04	5.41±0.03	5.96±0.28	4.82±0.10	4.89±0.31	6.76±0.17
	+ CutMix + DLB	**5.20**±0.13	**5.21**±0.01	**5.22**±0.66	**4.41**±0.25	**4.29**±0.20	**5.28**±0.06
	+ DDGSD	5.65±0.14	5.79±0.08	5.96±0.04	4.77±0.04	4.72±0.13	6.83±0.21
	+ DDGSD + DLB	**5.31**±0.07	**5.52**±0.05	**5.75**±0.25	**4.45**±0.59	**4.14**±0.03	**5.74**±0.23
C100	Baseline	26.37±0.19	27.16±0.42	28.26±0.18	23.64±0.25	22.42±0.23	28.31±0.35
	+ DLB	**23.88**±0.06	**24.53**±0.13	**26.00**±0.03	**21.82**±0.19	**20.79**±0.33	**27.48**±0.21
	+ CutOut	25.76±0.16	26.19±0.69	27.72±0.02	22.09±0.18	21.14±0.37	28.58±0.47
	+ CutOut + DLB	**23.02**±0.08	**23.62**±0.17	**25.93**±0.28	**20.27**±0.33	**20.30**±0.01	**27.04**±0.43
	+ CutMix	24.07±1.02	25.67±0.04	27.31±0.42	21.42±0.1	20.57±0.27	28.27±0.42
	+ CutMix + DLB	**23.48**±0.29	**24.06**±0.01	**25.77**±0.64	**20.93**±0.04	**20.10**±0.11	**26.82**±0.12
	+ DDGSD	24.36±0.04	24.69±0.01	26.32±0.23	22.55±0.17	20.83±0.08	27.69±0.32
	+ DDGSD + DLB	**23.81**±0.11	**24.20**±0.28	**25.98**±0.04	**21.18**±0.42	**20.28**±0.01	**27.25**±0.28



(a) CIFAR-100 (b) CIFAR-10

Figure 2. Performance on training with label noise. (a) CIFAR-100. (b) CIFAR-10. The line shows the top-1 validation error rate (%) on corrupted data at different percentage (p) of the noisy label i.e., $p \in \{0, 10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$. The bar illustrates the improvement of our approach, compared with baseline.

tion (LSR) [27] and self-knowledge distillation regularization approaches, including teacher-free knowledge distillation (Tf-KD$_{self}$,Tf-KD$_{reg}$) [37], class-wise self-knowledge distillation (CS-KD) [39], progressive self-knowledge distillation (PS-KD) [12]. The above methods focus on logit-level regularization. Data-distortion guided self-knowledge distillation (DDGSD) is a data augmentation based distillation approach [35], which was compared and tested the compatibility with DLB. We removed the feature-level supervision in DDGSD [35] i.e. MMD loss, for a fair comparison. As DDGSD is an augmentation based method, we explore the performance compatibility between DLB and



(a) Vgg-16 (b) Vgg-19 (c) WRM (d) DenseNet

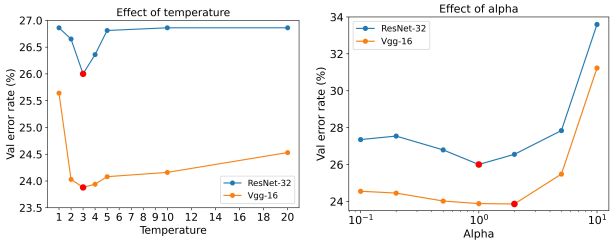
(e) Vgg-16 (f) Vgg-19 (g) WRM (h) DenseNet

Figure 3. The training performance of Vgg-16, Vgg-19, WRN, DenseNet on corrupted data. (a)-(d) Results on data corruption rate $p \in \{0.1, 0.3, 0.5\}$; (e)-(h) Results on data corruption rate $p \in \{0.2, 0.4, 0.6\}$.

DDGSD. All the extra hyper-parameters involved in the compared methods were retained as their original settings.

4.2. Classification Results

As illustrated in Table 2, our method (denoted as DLB) consistently improved the performance on various backbones (baseline). To be more specific, the averaged error rate improvement achieved by DLB ranged from 0.83% to 2.50% on CIAFR-100, 0.37% to 1.01% on CIFAR-10, and 0.81% to 3.17 on TinyImageNet. It shows the effectiveness of DLB that can significantly improve the generalization ability on various classification tasks. Moreover, DLB outperformed the state-of-the-art approaches, achieving the lowest top-1 error. The best and second-best performances on each set were highlighted in red and green respectively.



(a) Temperature (b) Alpha

Figure 4. Impact of hyper-parameters in DLB on CIFAR-100 to ResNet-32 and Vgg-16. (a) Impact of temperature $\tau \in \{1, 2, 3, 4, 5, 10, 20\}$, with a fixed $\alpha = 1$. (b) Impact of balancing coefficient $\alpha \in \{0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0\}$ with fixed temperature 3. The best performance in each controlled experiment is marked out by red.

We can observe that DLB on CIFAR-10 succeeded the state-of-the-arts by 0.30% with WRN20-8; whilst on CIFAR-100 by 0.47%. These improvements are attributed to the self-distillation regularization from the last mini-batch. We notice that DLB significantly outperformed Tf-KD [37], and PS-KD [12]. It demonstrates the performance advantage of DLB to generalize CNN. Furthermore, the identical teacher from the last mini-batch provides dynamically updated smoothed labels that fit the training process better than a pre-trained teacher or the last-epoch backup. Conclusively, DLB can efficiently serve as a universal regularization to normally train neural networks.

4.3. Compatibility with Augmentations

Previous work claims that data augmentation and distillation provide an orthogonal improvement [22]. To evaluate the compatibility of DLB with data augmentation based regularization, we combined our method with CutMix [38], CutOut [6] and DDGSD [35] on CIFAR-10/100.

CutOut. CutOut randomly masks a square region in the training samples [6], we set the number of holes to 1 and the hole size to 16. Combining DLB with CutOut is straightforward. As shown in Table 3, DLB can progressively improve CutOut by 0.45% to 1.14% on CIFAR-10 and 0.84% to 2.74% on CIFAR-100. The performance enhancement was slightly higher than itself on the baseline. It suggests that DLB works in conjunction with CutOut.

CutMix. CutMix randomly cuts and pasts patches in a mini-batch for regularization, where we follow the addi-

tional hyper-parameters setting as its original work [38] i.e., $\beta = 1$ for beta distribution and augmentation probability $p = 0.5$. CutMix improved the baseline by approximately 0.52% on CIFAR-10 and 1.48% on CIFAR-100. We plugged CutMix into our approach by performing self-distillation on a batch of images distorted by the same CutMix operation. It resulted in an extra 0.42% improvement on CIFAR-10 and 1.02% on CIFAR-100.

DDGSD. DDGSD is a self-distillation scheme, excavating a consistency between different distorted versions of the same images [35]. We plugged DLB into DDGSD by distilling the last mini-batch for both two predictions from different augmented versions. As demonstrated in Table 3, DLB slightly outperforms DDSGD. Moreover, combining DLB with DDSGD can obtain observable improvements.

Discussion. We empirically show that DLB and augmentation regularization work orthogonally. To be more specific, an extra performance gain can be achieved by combing DLB with augmentation based regularization. These finds indicate that using DLB as a plugged-in regularization can enhance the generation of other approaches.

4.4. Robustness to Data Corruption

In this section, we try to understand how DLB works by empirically exploring its regularization effect on a corrupted setting. DLB is expected to impose training stability and consistency. The superiority of DLB is its simplicity for implementation and parallelization. Additionally, DLB is expected to amplify the robustness of the target neural network, especially in noisy data. It results in a stronger tolerance to the corrupted data, by mitigating the over-fitting to label noise [16]. To prove this statement, we kept the experimental settings aligned with previous works [1, 20, 42] by randomly injecting symmetric label noise at different rate $p \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ to CIFAR-100/CIFAR-10 before the training process. Whilst, the test set was kept clean. In Figure 2 (a)-(b), we can observe a stable performance improvement on different models. For example, on CIFAR-100, DLB improved baseline by 2.38% at $p = 0.1$, and 4.44% at $p = 0.6$. The generalization improvement went increasingly higher with a higher label noise rate p. These observations demonstrate that DLB can effectively mitigate a neural network to fit label noise and improve the overall performance. The training performance of Vgg-16, Vgg-19, WideResNet, DenseNet are visualized in Figure 3.

4.5. Impact of Hyper-parameters

To analyze the impact of two hyper-parameters in DLB, namely the temperature τ in Eq. (4) and balancing coefficient α in Eq. (5), we carried out controlled experiments. Firstly, we fixed α to 1 and assigned τ with different values ranged in $\{1, 2, 3, 4, 5, 10, 20\}$ to evaluate the performance of ResNet-32, Vgg-16 on CIFAR-100. As plotted in Figure 4 (a), DLB achieved lowest error rate with a temperature $\tau = 3$. The performance reliance on α is demonstrated in Figure 4 (b), with a fixed $\tau = 3$. We observe that DLB performs well when α changes in $[0.5, 2.0]$.

4.6. Ablation Study

We removed the last batch distillation loss and only retained the cross-entropy loss for ablation on CIFAR-100. Specifically, ResNet-32 achieved a 28.01% test top-1 error rate, which performs slightly better than the baseline (28.26%). However, it was far worse than our DLB (26.00%). These findings show the significance to include the distillation loss as well as the effectiveness of our method. Detailed ablation results on all models are reported in the Table 4.

Table 4. Ablation study on CIFAR-100.

Networks	Baseline	loss in (4) removed	All settings
Vgg-16	$26.37_{\pm 0.19}$	$25.45_{\pm 0.13}$	$23.88_{\pm 0.06}$
Vgg-19	$27.16_{\pm 0.42}$	$25.73_{\pm 0.28}$	$24.53_{\pm 0.13}$
ResNet-32	$28.26_{\pm 0.18}$	$28.01_{\pm 0.02}$	$26.00_{\pm 0.03}$
ResNet-110	$23.64_{\pm 0.25}$	$22.65_{\pm 0.23}$	$21.82_{\pm 0.19}$
WRN20-8	$22.42_{\pm 0.23}$	$21.18_{\pm 0.01}$	$20.79_{\pm 0.33}$
DenseNet-40-12	$28.31_{\pm 0.35}$	$28.30_{\pm 0.24}$	$27.48_{\pm 0.21}$

5. Limitation and Conclusion

In this research, we introduce an efficient self-distillation mechanism for consistency regularization. Without the involvement of a complex pre-trained teacher model or an ensemble of peer students, our method (DLB) distills the on-the-fly generated smooth labels in the previous iteration after rearranging the sampling sequence. DLB regularizes network by imposing training consistency, which is further amplified under data corruption settings. Thus, it boosts the robustness to label noise. Experimental results on three benchmark datasets suggest that our method can consistently outperform the state-of-the-art self distillation mechanism. Moreover, DLB, as a universal regularization approach, works in conjunction with augmentations techniques, bringing additional performance gain. However, due to the limitation of computational resources, we did not evaluate the performance on large-scale datasets such as ImageNet, which is left to future work. Additionally, as our method depends on the knowledge transmission between soft labels, in this paper, we focus primarily on classification, which yields another future direction in the extension to other tasks e.g. semantic segmentation, object detection.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019. 8

[2] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. 1, 3

[3] Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. Online knowledge distillation with diverse peers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3430–3437, 2020. 1, 3, 5

[4] Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine: Distilling with cheap convolutions. In *NeurIPS*, pages 2893–2903, 2018. 1, 3

[5] Xiang Deng and Zhongfei Zhang. Learning with retrospection. *arXiv preprint arXiv:2012.13098*, 2020. 2

[6] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 6, 7

[7] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018. 1

[8] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11020–11029, 2020. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4, 5

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 3, 4

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 5

[12] Kyungyul Kim, ByeongMoon Ji, Doyoung Yoon, and Sangheum Hwang. Self-knowledge distillation with progressive refinement of targets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6567–6576, 2021. 2, 3, 4, 6, 7

[13] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 2, 4

[14] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. *arXiv preprint arXiv:1806.04606*, 2018. 1, 3

[15] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, pages 5714–5724. PMLR, 2020. 3

[16] Mengchen Liu, Shixia Liu, Hang Su, Kelei Cao, and Jun Zhu. Analyzing the noise robustness of deep neural networks. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 60–71. IEEE, 2018. 8

[17] Yunteng Luan, Hanyu Zhao, Zhi Yang, and Yafei Dai. Msd: Multi-self-distillation learning via multi-classifiers within deep neural networks. *arXiv preprint arXiv:1911.09418*, 2019. 3

[18] Le Thanh Nguyen-Meidine, Atif Belal, Madhu Kiran, Jose Dolz, Louis-Antoine Blais-Morin, and Eric Granger. Unsupervised multi-target domain adaptation through knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1339–1347, 2021. 1, 3

[19] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364, 2019. 1

[20] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 8

[21] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 1

[22] Fabian Ruffy and Karanbir Chahal. The state of knowledge distillation for classification. *arXiv preprint arXiv:1912.10850*, 2019. 7

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3

[24] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Knowledge distillation beyond model compression. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6136–6143. IEEE, 2021. 1

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 5

[26] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019. 1, 3

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2, 4, 6

[28] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017. 1, 3

[29] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019. 5

[30] Jiyue Wang, Pei Zhang, and Yanxiong Li. Memory-replay knowledge distillation. *Sensors*, 21(8):2792, 2021. 2, 3

[31] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3

[32] Xinglu Wang and Yingming Li. Harmonized dense knowledge distillation training for multi-exit architectures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10218–10226, 2021. 1

[33] Guile Wu and Shaogang Gong. Peer collaborative learning for online knowledge distillation. In *AAAI*, 2021. 1, 3, 5

[34] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *European Conference on Computer Vision*, pages 588–604. Springer, 2020. 5

[35] Ting-Bing Xu and Cheng-Lin Liu. Data-distortion guided self-distillation for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5565–5572, 2019. 1, 2, 3, 6, 7, 8

[36] 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*, pages 2859–2868, 2019. 1, 2

[37] 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, 4, 6, 7

[38] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 6, 7, 8

[39] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13876–13885, 2020. 2, 3, 6

[40] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 1

[41] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2, 3, 4, 5

[42] Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30:5984–5996, 2021. 5, 8

[43] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019. 1, 2, 3

[44] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018. 1, 3, 5