Self-Adaptive Training: Bridging Supervised and Self-Supervised Learning

Lang Huang*, Chao Zhang*, Member, IEEE, and Hongyang Zhang*, Member, IEEE

Abstract—We propose self-adaptive training—a unified training algorithm that dynamically calibrates and enhances training processes by model predictions without incurring an extra computational cost—to advance both supervised and self-supervised learning of deep neural networks. We analyze the training dynamics of deep networks on training data that are corrupted by, e.g., random noise and adversarial examples. Our analysis shows that model predictions are able to magnify useful underlying information in data and this phenomenon occurs broadly even in the absence of any label information, highlighting that model predictions could substantially benefit the training processes: self-adaptive training improves the generalization of deep networks under noise and enhances the self-supervised representation learning. The analysis also sheds light on understanding deep learning, e.g., a potential explanation of the recently-discovered double-descent phenomenon in empirical risk minimization and the collapsing issue of the state-of-the-art self-supervised learning algorithms. Experiments on the CIFAR, STL, and ImageNet datasets verify the effectiveness of our approach in three applications: classification with label noise, selective classification, and linear evaluation. To facilitate future research, the code has been made publicly available at https://github.com/LayneH/self-adaptive-training.

Index Terms—Deep learning, supervised learning, self-supervised learning, generalization, robust learning under noise

1 INTRODUCTION

Deep neural networks have received significant attention in machine learning and computer vision, in part due to their impressive performance achieved by supervised learning approaches in the ImageNet challenge. With the help of massive labeled data, deep neural networks advance the state-of-the-art to an unprecedented level on many fundamental tasks, such as image classification [2], [3], object detection [4], and semantic segmentation [5]. However, data acquisition is notoriously costly, error-prone, and even infeasible in certain cases. Furthermore, deep neural networks suffer significantly from overfitting in these scenarios. On the other hand, the great success of self-supervised pre-training in natural language processing (e.g., GPT [6], [7], [8] and BERT [9]) highlights that learning universal representations from unlabeled data can be even more beneficial for a broad range of downstream tasks.

Regarding this, much effort has been devoted to learning representations without human supervision for computer vision. Several recent studies show promising results and largely close the performance gap between supervised and self-supervised learning. To name a few, the contrastive learning approaches [10], [11], [12] solve the instance-wise discrimination task [13] as a proxy objective of representation learning. Extensive studies demonstrate that self-supervisedly learned representations are generic and even outperform the supervised pre-trained counterparts when they are fine-tuned on certain downstream tasks.

Our work advances both supervised learning and self-supervised learning settings. Instead of designing two distinct algorithms for each learning paradigm separately, in this paper, we explore the possibility of a unified algorithm that bridges supervised and self-supervised learning. Our exploration is based on two observations on the learning of deep neural networks.

Observation 1. Under various supervised learning settings with noisy data, deep neural networks are able to magnify useful underlying information of data by their predictions.

Observation 2. Observation 1 extends to the extreme noise where the supervised signals are completely random, which is equivalent to self-supervised learning.

These two observations indicate that model predictions can magnify useful information in data, which further indicates that incorporating predictions into the training processes could significantly benefit model learning. With this in mind, we propose self-adaptive training—a carefully designed approach that dynamically uses model predictions as a...
guiding principle in the design of training algorithms—that bridges supervised learning and self-supervised learning in a unified framework. Our approach is conceptually simple yet calibrates and significantly enhances the learning of deep models in multiple ways.

1.1 Summary of Our Contributions
Self-adaptive training sheds light on understanding and improving the learning of deep neural networks.

- We analyze the Empirical Risk Minimization (ERM) training processes of deep models on four kinds of corruptions (see Fig. 1a). We describe the failure scenarios of ERM and observe that useful information from data has been distilled to model predictions in the first few epochs. We show that this phenomenon occurs broadly even in the absence of any label information (see Fig. 2). These insights motivate us to propose self-adaptive training—a unified training algorithm for both supervised and self-supervised learning—to improve the learning of deep neural networks by dynamically incorporating model predictions into training, requiring no modification to existing network architecture and incurring almost no extra computational cost.

- We show that self-adaptive training improves the generalization of deep networks under both label-wise and instance-wise random noise (see Figs. 1 and 3). Besides, self-adaptive training exhibits a single-descent error-capacity curve (see Fig. 4). This is in sharp contrast to the recently-discovered double-descent phenomenon in ERM, which might be a result of overfitting to noise. Moreover, while adversarial training may easily overfit adversarial noise, our approach mitigates the overfitting issue and improves the adversarial accuracy by \( \sim 3\% \) over the state-of-the-art (see Fig. 5).

- Self-adaptive training questions and alleviates the dependency of recent self-supervised algorithms on the dominant training mechanism that typically involves multiple augmented views of the same images at each training step: self-adaptive training achieves remarkable performance despite requiring only a single view of each image for training, which significantly reduces the heavy cost of data pre-processing and model training on extra views.

- Learning with noisy labels, where the goal is to improve the performance of deep networks on clean test data in the presence of training label noise. On the CIFAR datasets, our approach obtains up to 9\% absolute classification accuracy improvement over the state-of-the-art. On the ImageNet dataset, our approach...

Fig. 1. Accuracy curves of models on the CIFAR10 dataset with 40\% of corrupted data. All models can only have access to the noisy training set (corresponding to the red dashed curve) during training and are tested on the other three sets for illustration. The horizontal dotted line displays the percentage of clean data in the training sets. It shows that while ERM suffers from overfitting to noise, our method enjoys improved generalization and higher validation accuracy (e.g., the improvement is larger than 10\% in the left-most column).

Fig. 2. Linear evaluation accuracy on CIFAR10 of checkpoints of the model trained by fitting the noise as described in Section 2.1 (the red and green curves) and fitting the accumulated representation of the model (the blue curve). The horizontal dashed line indicates the accuracy of linear evaluation directly using a randomly initialized network. All the curves are smoothed for better demonstration.
approach improves over ERM by 3% under 40% noise rate.

- Selective classification, which trades prediction coverage for classification accuracy. Our approach achieves up to 50% relative improvement over the state-of-the-art on three datasets under various coverage rates.

- Linear evaluation, which evaluates the representations of a self-supervised pre-trained model using a linear classifier. Our approach performs on par with or even better than the state-of-the-art on various datasets. In particular, on the ImageNet dataset, self-adaptive training achieves 72.8% top1 linear accuracy using only 200 pre-training epochs, surpassing the other methods by 2.2% in absolute under the same setting.

2 SELF-ADAPTIVE TRAINING

2.1 Blessing of Model Predictions

On Corrupted Data. Recent works [14], [15] cast doubt on the ERM training; techniques such as uniform convergence might be unable to explain the generalization of deep neural networks because ERM easily overfits the training data even though the training data are partially or completely corrupted by random noise. To take a closer look at this phenomenon, we conduct the experiments on the CIFAR10 dataset [16], splitting the original training data into a training set (consists of the first 45,000 data pairs) and a validation set (consists of the last 5,000 data pairs). We measure four random noise schemes according to prior work [14], where the data are partially corrupted with probability $p$: 1) Corrupted labels. Labels are assigned uniformly at random; 2) Gaussian. Images are replaced by random Gaussian samples with the same mean and standard deviation as the original image distribution; 3) Random pixels. Pixels of each image are shuffled using independent random permutations; 4) Shuffled pixels. Pixels of each image are shuffled using a fixed permutation pattern. We consider the performance on both the noisy and the clean sets (i.e., the original uncorrupted data), while the models can only have access to the noisy training sets.

Fig. 1a displays the accuracy curves of ERMs that are trained on the noisy training sets under four kinds of random corruptions: ERM easily overfits noisy training data and achieves nearly perfect training accuracy. However, the four subfigures exhibit very different generalization behaviors which are indistinguishable if we only look at the accuracy curve on the noisy training set (the red curve). In Fig. 1a, the accuracy increases in the early stage, and the generalization errors grow quickly only after a certain number of epochs. Intuitively, early-stopping improves the generalization in the presence of label noise (see the first column in Fig. 1a); however, it remains unclear how to properly identify such an epoch without using validation data. Moreover, the early-stop mechanism may significantly hurt the performance on the clean validation sets, as we can see in the last three columns of Fig. 1a. Our approach is motivated by the failure scenarios of ERM and goes beyond ERM. We begin by making the following observations in the leftmost subfigure of Fig. 1a: the peak of the accuracy curve on the clean training set (>80%) is much higher than the percentage of clean data in the noisy training set (>60%). This finding was also previously reported by [17], [18], [19] under label corruption and suggested that model predictions might be able to magnify useful underlying information in data. We confirm this finding and show that the pattern occurs under various kinds of corruption more broadly (see the last three subfigures of Fig. 1a).

On Unlabelled Data. We notice that supervised learning with 100% noisy labels is equivalent to unsupervised learning if we simply discard the meaningless labels. Therefore, it
would be interesting to analyze how deep models behave in such an extreme case. Here, we conduct experiments on the CIFAR10 [16] dataset and consider two kinds of random noise as the training (real-valued) targets for deep learning models: 1) the output features of another model on the same training images, where the model is randomly initialized and then frozen; 2) random noise that is drawn i.i.d. from standard Gaussian distribution and then fixed. The training of the deep model is then formulated as minimizing the mean square error between $f$-normalized model predictions and these two kinds of random noise. To monitor the training, we learn a linear classifier on the top of each model to evaluate its representation at each training epoch (see Appendix B.4, available online, for the detailed setup of this classifier).

Fig. 2 shows the linear evaluation accuracy curves of models trained on these two kinds of noise, i.e., the green and red curves (the explanation of the blue curve is deferred to Section 6.1). We observe that, perhaps surprisingly, the model trained by predicting fixed random Gaussian noise (the green curve) achieves 57\% linear evaluation accuracy, which is substantially higher than the 38\% accuracy of a linear classifier. This intriguing observation shows that deep neural networks are able to distill underlying information from data to its predictions, even when supervision signals are completely replaced by random noise. Furthermore, although predicting the output of another network can also improve the representation learning (the red curve), its performance is worse than that of the second scheme. We hypothesize that this might be a result of inconsistent training targets: the output of a network depends on the input training image, which is randomly transformed by the data augmentation at each epoch. This suggests that the consistency of training targets should also be taken into account in the design of the training algorithm.

**Inspiration for Our Methodology.** Based on our analysis, we propose a unified algorithm, **Self-Adaptive Training**, for both supervised and self-supervised learning. Self-adaptive training incorporates model predictions to augment the training processes of deep models in a dynamic and consistent manner. Our methodology is general and very effective: self-adaptive training significantly improves (a) the generalization of deep models on the corrupted data, and (b) the representation learning of deep models without human supervision.

### 2.2 Meta Algorithm: Self-Adaptive Training

Given a set of training images $\{x\}_n$ and a deep network $f_\theta(\cdot)$ parametrized by $\theta$, our approach records training targets $\{t\}_n$ for all data points accordingly. We first obtain the predictions of the deep network as

$$\hat{p}_i = \rho(f_\theta(x_i)).$$

where $\rho(\cdot)$ is a normalization function. Then, the training targets track all historical model predictions during training and are updated by Exponential-Moving-Average (EMA) scheme as

$$t_i \leftarrow \alpha \times t_i + (1 - \alpha) \times \hat{p}_i.$$  

(2)

The EMA scheme in Equation (2) alleviates the instability issue of model predictions, smooths out $t_i$ during the training process, and enables our algorithm to completely change the training labels if necessary. The momentum term $\alpha$ controls the weight of the model predictions. Finally, we can update the weights $\theta$ of the deep network $f_\theta$ by Stochastic Gradient Descent (SGD) on the loss function $L(p, t; f_\theta)$ at each training iteration.

We summarize the meta algorithm of self-adaptive training in Algorithm 1. The algorithm is conceptually simple, flexible, and has three components adapting to different learning settings: 1) the training targets initialization; 2) normalization function $\rho(\cdot)$; 3) loss function $L(p, t; f_\theta)$. In the following sections, we will elaborate on the instantiations of these components for specific learning settings.

**Algorithm 1. Self-Adaptive Training**

**Require:** Data $\{x\}_n$, deep network $f_\theta(\cdot)$ parametrized by $\theta$, momentum term $\alpha$, normalization function $\rho(\cdot)$

1. Initialize targets $\{t\}_n$
2. repeat
3. Fetch mini-batch data $\{(x, t)\}_m$
4. for $i = 1$ to $m$ (in parallel) do
5. $\hat{p}_i = \rho(f_\theta(x_i))$
6. $t_i \leftarrow \alpha \times t_i + (1 - \alpha) \times \hat{p}_i$
7. end for
8. Update $f_\theta$ by SGD on $L(p, t; f_\theta)$
9. until end of training

**Convergence Analysis.** To simplify the analysis, we consider a linear regression problem with data $\{x\}_n$, training targets $\{t\}_n$ and a linear model $f_\theta(x) = x^T \theta$, where $x, \theta \in \mathbb{R}^d$, $t_i \in \mathbb{R}$ and $\theta \in \mathbb{R}^d$. Let $X = [x_1 | x_2 | \cdots | x_n] \in \mathbb{R}^{d \times n}$, $t = [t_1 | t_2 | \cdots | t_n] \in \mathbb{R}^n$. Then the optimization for this regression problem (corresponding to $\min_\theta L(p, t; f_\theta)$ in Algorithm 1) can be written as

$$\arg \min_\theta \|X\theta - t\|^2_2.$$  

(3)

Let $\theta(k)$ and $t(k)$ be the model parameters and training targets at the $k$-th training step, respectively. Let $\eta$ denote the learning rate for gradient descent update. Algorithm 1 alternatively minimizes the problem (3) over $\theta(k)$ and $t(k)$ as

$$\theta(k) = \theta(k-1) - \eta \nabla f_\theta(X, \theta(k-1), t(k-1)),$$$$
t(k) = \alpha t(k-1) + (1 - \alpha) f_\theta(X, \theta(k)).$$

(4)

(5)

**Proposition 1.** Let $d_{\text{max}}$ be the maximal eigenvalue of the matrix $XX^T$, if the learning rate $\eta < \frac{1}{d_{\text{max}}}$, then

$$\lim_{k \to \infty} \|X\theta(k) - t(k)\|^2_2 = 0.$$  

(6)

**Corollary 1.** Under the same condition as in Proposition 1, we have
Combining Proposition 1 and Corollary 1, we see that, with a proper learning rate for gradient descent, the optimization of problem (3) converges at least Q-linearly to 0. The proofs of Proposition 1 and Corollary 1 are presented in Appendix A, available in the online supplemental material.

Limitations. We note that one disadvantage of self-adaptive training is the extra storage cost of the training targets, which might be a problem when training on an extremely large dataset. However, we find this cost is actually moderate in most realistic cases. Take the large-scale ImageNet dataset [20] as an example. The ImageNet consists of about 1.2 million images categorized into 1,000 classes. The storage of such vectors in the single precision format for the entire dataset requires $1.2 \times 10^9 \times 1000 \times 32$ bit, which is reduced to $\sim 1.12$GB under the self-supervised learning setting that records a 256-d feature for each image. The cost is acceptable since modern GPUs usually have 20GB or more dedicated memory, e.g., NVIDIA Tesla A100 has 40GB of memory. Moreover, the vectors can be (a) sharded to all GPU devices [21], (b) stored on the CPU memory, or even (c) stored on the disk and loaded along with the images, to further reduce the storage cost.

## 3 IMPROVED GENERALIZATION OF DEEP MODELS

### 3.1 Supervised Self-Adaptive Training

**Instantiation.** We consider a c-class classification problem and denote the images by $x$, labels by $y \in \{0,1\}^c$, $y' = 1$. Given a data pair $(x, y)$, our approach instantiates the three components of meta Algorithm 1 for supervised learning as follows:

1. **Targets initialization.** Since the labels are provided, the training target $t$ is directly initialized as $t \leftarrow y$.
2. **Normalization function.** We use the softmax function $\text{softmax}(u) = \exp(u) / \sum \exp(u)$ to normalize the model predictions into probability vectors $p = \text{softmax}(f(s(x)))$, such that $p \in [0,1]^c, p' = 1$.
3. **Loss function.** Following the common practice in supervised learning, the loss function is implemented as the cross entropy loss between model predictions $p$ and training targets $t$, i.e., $\sum t_{i,j} \log p_{i,j}$, where $t_{i,j}$ and $p_{i,j}$ represent the jth entry of $t$ and $p$, respectively.

During the training process, we fix $t$ in the first $E_s$ training epochs and update the training targets $t$ according to Equation (2) in each following training epoch. The number of initial epochs $E_s$ allows the model to capture informative signals in the data set and excludes ambiguous information that is provided by model predictions in the early stage of training.

**Sample Re-Weighting.** Based on the scheme presented above, we introduce a simple yet effective sample re-weighting scheme on each sample. Concretely, given training target $t$, we set

$$w_i = \max_j t_{i,j}. \quad (8)$$

The sample weight $w_i \in [0,1]$ reveals the labeling confidence of this sample. Intuitively, all samples are treated equally in the first $E_s$ epochs. As the target $t$ being updated, our algorithm pays less attention to potentially erroneous data and learns more from potentially clean data. This scheme also allows the corrupted samples to re-attain attention if they are confidently corrected.

**Putting Everything Together.** We use stochastic gradient descent to minimize:

$$L(f) = -\frac{1}{\sum_i w_i \sum_j w_j \sum t_{i,j} \log p_{i,j}} \quad (9)$$
during the training process. Here, the denominator normalizes per sample weight and stabilizes the loss scale. We summarize the *Supervised Self-Adaptive Training* and display the pseudocode in Algorithm 2. Intuitively, the optimal choice of hyper-parameter $E_s$ should be related to the epoch where overfitting occurs, which is around the 60th epoch according to Fig. 1a. For convenience, we directly fix the hyper-parameters $E_s = 60, \alpha = 0.9$ by default if not specified. Experiments on the sensitivity of our algorithm to hyper-parameters are deferred to Section 4.2. Our approach requires no modification to existing network architecture and incurs almost no extra computational cost.

**Algorithm 2. Supervised Self-Adaptive Training**

**Require:** Data $\{(x, y)_i\}_n$; classifier $f_0$, initial epoch $E_s = 60$; momentum term $\alpha = 0.9$.
1. Initialize training targets by $t_i = \{y_i\}_n$.
2. **repeat**
3. Fetch mini-batch data $\{(x, t)_i\}_m$ at current epoch $e$.
4. for $i = 1$ to $m$ (in parallel) do
5. $p_i = \text{softmax}(f_0(x_i))$
6. if $e \geq E_s$ then
7. $t_i \leftarrow \alpha \times t_i + (1 - \alpha) \times p_i$
8. end if
9. $w_i = \max_j t_{i,j}$
10. end for
11. $L(p_i, t_i; f_0) = -\frac{1}{\sum_i w_i \sum_j t_{i,j} \log p_{i,j}}$
12. Update $f_0$ by STGD on $L(p_i, t_i; f_0)$
13. **until** end of training

**Methodology Differences With Prior Work.** Supervised self-adaptive training consists of two components: a) label correction; b) sample re-weighting. With the two components, our algorithm is robust to both instance-wise and label-wise noise and is ready to combine with various training schemes such as natural and adversarial training, without incurring multiple rounds of training. In contrast, the vast majority of works on learning from corrupted data follow a preprocessing-training scheme with an emphasis on the label-wise noise only: this line of research either discards samples based on the disagreement between noisy labels and model predictions [22], [23], [24], [25], or corrects noisy labels [26], [27], [28] investigated a more generic approach that corrects both label-wise and instance-wise noise. However, their approach inherently suffers from extra computational overheads. Besides, unlike the general scheme in robust statistics [29] and other re-weighting methods [30], [31] that used an additional optimization step to update the sample weights, our approach directly obtains the weights based on accumulated model predictions and thus is much more efficient.
3.2 Improved Generalization Under Random Noise

We consider the noise scheme (including noise type and noise level) and model capacity as two factors that affect the generalization of deep networks under random noise. We analyze self-adaptive training by varying one of the two factors while fixing the other.

Varying Noise Schemes. We use ResNet-34 [3] and rerun the same experiments as in Fig. 1a by replacing ERM with our approach. In Fig. 1b, we plot the accuracy curves of models trained with our approach on four corrupted training sets and compare them with those in Fig. 1a. We highlight the following observations.

- Our approach mitigates the overfitting issue in deep networks. The accuracy curves on noisy training sets (i.e., the red dashed curves in Fig. 1b) nearly converge to the percentage of clean data in the training sets and do not reach perfect accuracy.
- The generalization errors of self-adaptive training (the gap between the red and blue dashed curves in Fig. 1b) are much smaller than Fig. 1a. We further confirm this observation by displaying the generalization errors of the models trained on the four noisy training sets under various noise rates in the leftmost subfigure of Fig. 3. The generalization errors of ERM consistently grow as we increase the injected noise level. In contrast, our approach significantly reduces the generalization errors across all noise levels from 0% (no noise) to 90% (overwhelming noise).
- The accuracy on the clean sets (cyan and yellow solid curves in Fig. 1b) is monotonically increasing and converges to higher values than that of the counterparts in Fig. 1a. We also show the clean validation errors in the right two subfigures in Fig. 3. The figures show that the error of self-adaptive training is consistently much smaller than that of ERM.

Varying Model Capacity. We notice that such analysis is related to a recently discovered intriguing phenomenon [32], [33], [34], [35], [36], [37], [38] in modern machine learning models: as the capacity of the model increases, the test error initially decreases, then increases, and finally shows a second descent. This phenomenon is termed double descent [32] and has been widely observed in deep networks [33]. To evaluate the double-descent phenomenon on self-adaptive training, we follow exactly the same experimental settings as [33]: we vary the width parameter of ResNet-18 [3] and train the networks on the CIFAR10 dataset with 15% of training labels being corrupted at random (details are given in Appendix B.1, available in the online supplemental material).

Fig. 4 shows the test error curves. It shows that self-adaptive training overall achieves a much lower test error than that of ERM except when using extremely small models that underfit the training data. This suggests that our approach can improve the generalization of deep networks especially when the model capacity is reasonably large. Besides, we observe that the curve of ERM clearly exhibits the double-descent phenomenon, while the curve of our approach is monotonously decreasing as the model capacity increases. Since the double-descent phenomenon may vanish when label noise is absent [33], our experiment indicates that this phenomenon may be a result of overfitting to noise and we can bypass it by a properly designed training process such as self-adaptive training.

3.3 Improved Generalization Under Adversarial Noise

Adversarial noise [39] is different from random noise in that the noise is model-dependent and imperceptible to humans. We use the state-of-the-art adversarial training algorithm TRADES [40] as our baseline to evaluate the performance of self-adaptive training under adversarial noise. Algorithmically, TRADES minimizes

$$E_x y \left\{ CE(p(x), y) + \max_{\|z-x\|_2 \leq \epsilon} \frac{KL(p(z), p(y))/\lambda}{10} \right\}$$

where \(p(\cdot)\) is the model prediction, \(\epsilon\) is the maximally allowed perturbation, CE stands for the cross entropy, KL stands for the Kullback–Leibler divergence, and the hyperparameter \(\lambda\) controls the trade-off between robustness and accuracy. We replace the CE term in TRADES loss with our method. The models are evaluated using robust accuracy [41] with \(\epsilon = 0.031\) (the evaluation of projected gradient descent attack [42] is given in Fig. 14 of Appendix C, available in the online supplemental material). We set the initial learning rate as 0.1 and decay it by a factor of 0.1 in epochs 75 and 90, respectively. We choose \(1/\lambda = 6.0\) as suggested by [40] and use \(E_s = 70, \alpha = 0.9\) for our approach. Experimental details are given in Appendix B.2, available in the online supplemental material.

We display the robust accuracy on the CIFAR10 test set after \(E_s = 70\) epochs in Fig. 5. It shows that the robust accuracy of TRADES reaches its highest value around the epoch of the first learning rate decay (epoch 75) and decreases later, which suggests that overfitting might happen if we train the model without early stopping. On the other hand, our method considerably mitigates the overfitting issue in the adversarial training and consistently improves the robust accuracy of TRADES by 1%-3%, which indicates that self-adaptive training can improve the generalization in the presence of adversarial noise.

4 Application I: Learning With Noisy Label

Given the improved generalization of self-adaptive training over ERM under noise, we show the applications of our approach which outperforms the state-of-the-art with a significant gap.

4.1 Problem Formulation

Given a set of noisy training data \(\{(x_i, \tilde{y}_i)\} \in D\), where \(D\) is the distribution of noisy data and \(\tilde{y}_i\) is the noisy label for each uncorrupted sample \(x_i\), the goal is to be robust to the label noise in the training data and improve the classification performance on clean test data that are sampled from the clean distribution \(D\).

4.2 Experiments on CIFAR Datasets

Setup. We consider the cases that, with different noise rates, the labels are (a) assigned uniformly at random (i.e., the
symmetric noise), and (b) flipped according to the class-conditioned rules [44] (i.e., the asymmetric noise): for CIFAR10, the label noise is generated by mapping Truck → Automobile, Bird → Airplane, Deer → Horse, Cat → Dog; for CIFAR100, the noise is generated by flipping each class to the next class circularly. Following previous work [46], [48], we conduct the experiments on the CIFAR10 and CIFAR100 datasets [16] and use the ResNet-34 [3] /Wide ResNet-28-10 [50] (WRN28-10) as our base classifier. The networks are implemented on PyTorch [51] and are optimized using SGD with an initial learning rate of 0.1, a momentum of 0.9, a weight decay of 0.0005, a batch size of 256, and the total training epochs of 200. The learning rate is decayed to zero using the cosine annealing schedule [52]. We use the standard data augmentation with random horizontal flipping and cropping. We report the average performance over 3 trials.

Main Results. We summarize the experiments in Table 1. Most of the results are directly cited from original papers with the same experiment settings; the results of Label Smoothing [43], Mixup [45], Joint Opt [27], and SCE [47] are reproduced by rerunning the official open-sourced implementations. From the table, we can see that our approach outperforms the state-of-the-art methods in most entries by 1% ~ 5% on both CIFAR10 and CIFAR100 datasets, using different backbones. We observe that the improvements by our approach are consistent under both symmetric and asymmetric noise, demonstrating the robustness of self-adaptive training against various kinds of noise. Notably, unlike the Joint Opt, DAC, and SELF methods that require multiple iterations of training, our method enjoys the same computational budget as ERM.

Ablation Study and Hyper-Parameter Sensitivity. First, we report the performance of ERM equipped with the simple early stopping scheme in the first row of Table 1. We observe that our approach achieves substantial improvements over this baseline. This demonstrates that simply early stopping the training process is a sub-optimal solution. Then, we further report the influences of the two individual components of our approach: Exponential Moving Average (EMA) and sample re-weighting scheme. As displayed in Table 2, removing any component considerably hurts the performance under all noise rates, and removing the EMA scheme leads to a significant performance drop. This suggests that properly incorporating model predictions is important in our approach. Finally, we analyze the sensitivity of our approach to the parameters \( \alpha \) and \( E_s \) in Table 3, where we fix one parameter while varying the other. The performance is stable for various choices of \( \alpha \) and \( E_s \), indicating that our approach is not sensitive to hyperparameter tuning.

### 4.3 Experiments on ImageNet Dataset

The work of [53] suggested that the ImageNet dataset [20] contains annotation errors on its own even after several rounds of cleaning. Therefore, in this subsection, we use ResNet-50/101 [3] to evaluate self-adaptive training on the large-scale ImageNet under both the standard setup (i.e., using original labels) and the case that 40% of the training labels are corrupted. We provide the experimental details in Appendix B.3, available in the online supplemental material, and report model performance on the ImageNet validation set in terms of top1 accuracy in Table 4. We can see that self-adaptive training consistently improves the ERM.

| Backbone | CIFAR10 | CIFAR100 |
|----------|---------|----------|
| Ours     | 94.12   | 92.64    |
| ERM + Early Stop | 85.57   | 81.82    |
| Label Smooth [43] | 85.64   | 71.59    |
| Forward T [44] | 87.99   | 83.25    |
| Mixup [45] | 93.58   | 89.46    |
| Trunc D [46] | 89.70   | 87.62    |
| Joint Opt [27] | 92.25   | 90.79    |
| SCE [47] | 90.15   | 86.74    |
| DAC [48] | 92.91   | 90.71    |
| SELF [25] | 91.13   | 91.13    |
| ELR [49] | 92.12   | 94.13    |
| Ours     | 94.14   | 92.64    |

| Backbone | CIFAR10 | CIFAR100 |
|----------|---------|----------|
| Ours     | 94.84   | 93.23    |
| ERM + Early Stop | 87.86   | 83.40    |
| MentorNet [30] | 92.0    | 89.0     |
| DAC [48] | 93.25   | 90.93    |
| SELF [25] | 93.34   | 67.41    |

| Label Noise Rate | CIFAR10 | CIFAR100 |
|------------------|---------|----------|
| 0.4              | 92.64   | 78.58    |
| 0.8              | 71.38   | 38.72    |
| - Re-weighting   | 92.49   | 78.10    |
| - Exponential Moving Average | 72.00   | 28.17    |

We compare with previous works under exactly the same experiment settings. It shows that, in all settings, self-adaptive training improves over the state-of-the-art by at least 1%, and sometimes the improvement is as significant as 9%. The best entries are bold-faced.
baseline by a considerable margin under all settings using different models. Specifically, the improvement can be as large as 3% in absolute for the larger ResNet-101 when 40% of the training labels are corrupted. The results validate the effectiveness of our approach on the large-scale dataset and larger model.

4.4 Label Recovery of Self-Adaptive Training

We demonstrate that our approach is able to recover the true labels from noisy training labels: we obtain the recovered labels by the moving average targets $\hat{t}_i$ and compute the recovered accuracy as $\sum_{i} \frac{1}{n} \{\text{argmax } y_i = \text{argmax } \hat{t}_i\}$, where $y_i$ is the clean label of each training sample. When 40% of labels are corrupted in the CIFAR10 and ImageNet training set, our approach successfully recovers a huge amount of labels and obtains recovered accuracy of 94.6% and 81.1%, respectively. We also display the confusion matrix of recovered labels w.r.t the clean labels on CIFAR10 in Fig. 6, where we can see that our approach performs well for all classes.

4.5 Investigation of Sample Weights

We further inspect the re-weighting scheme of self-adaptive training. Following the procedure in Section 4.4, we display the average sample weights in Fig. 7. In the figure, the $(i,j)$th block contains the average weight of samples with clean label $i$ and recovered label $j$, the white areas represent the case that no sample lies in the cell. We see that the weights on the diagonal blocks are clearly higher than those on non-diagonal blocks. The figure indicates that, aside from its impressive ability to recover the correct labels, self-adaptive training could properly down-weight the noisy examples.

5 APPLICATION II: SELECTIVE CLASSIFICATION

5.1 Problem Formulation

Selective classification, a.k.a. classification with rejection, trades classifier coverage for accuracy [57], where the coverage is defined as the fraction of classified samples in the dataset; the classifier is allowed to output “don’t know” for certain samples. The task focuses on the noise-free setting and allows the classifier to abstain from potential out-of-distribution samples or samples that lies in the tail of data distribution, that is, making prediction only on the samples with confidence.

Formally, a selective classifier is a composition of two functions $(f_0, g)$, where $f_0$ is the conventional $c$-class classifier and $g$ is the selection function that reveals the underlying uncertainty of inputs. Given an input $x$, selective classifier outputs

\[
(f_0, g)(x) = \begin{cases} 
\text{Abstain}, & g(x) > r; \\
 f_0(x), & \text{otherwise},
\end{cases}
\]

for a given threshold $r$ that controls the trade-off.

5.2 Approach

Inspired by [48], [54], we adapt our presented approach in Algorithm 1 to the selective classification task. We introduce an extra $(c+1)$th class (represents abstention) during training and replace selection function $g(\cdot)$ in Equation (11) by $f_0(\cdot)$. In this way, we can train a selective classifier in an end-to-end fashion. Besides, unlike previous works that provide no explicit signal for learning abstention class, we use model predictions as a guideline when designing the learning process.

Given a mini-batch of data pairs $\{(x_i, y_i)\}_{i=1}^n$, the model prediction $p_i$ and its exponential moving average $\hat{t}_i$ for each

```plaintext
| Label Noise Rate | 0.0 | 0.4 | 0.0 | 0.4 |
|------------------|-----|-----|-----|-----|
| ERM              | 76.8| 69.5| 78.2| 70.2|
| Ours             | 77.2| 71.5| 78.7| 73.5|
```

Table 3: Parameters Sensitivity When Label Noise of 40% is Injected Into the CIFAR10 Training Set
by minimizing:

$$\mathcal{L}(f_{\theta}) = - \frac{1}{m} \sum_{i} [t_{i,y_{i}} \log p_{i,y_{i}} + (1 - t_{i,y_{i}}) \log p_{i,c}],$$

(12)

where $y_{i}$ is the index of the non-zero element in the one hot label vector $y$. The first term measures the cross entropy loss between prediction and original label $y_{i}$, in order to learn a good multi-class classifier. The second term acts as the selection function and identifies uncertain samples in datasets. $t_{i,y_{i}}$ dynamically trades-off these two terms: if $t_{i,y_{i}}$ is very small, the sample is deemed as uncertain and the second term enforces the selective classifier to learn to abstain from this sample; if $t_{i,y_{i}}$ is close to 1, the loss recovers the standard cross entropy minimization and enforces the selective classifier to make perfect predictions.

5.3 Experiments

Setup. We conduct the experiments on three datasets: CIFAR10 [16], SVHN [58], and Dogs versus Cats [59]. We compare our method with previous state-of-the-art works on selective classification, including Deep Gamblers [54], SelectiveNet [55], Softmax Response (SR), and MC-dropout [56]. The experiments are based on the official open-sourced implementation of Deep Gamblers to ensure a fair comparison. We use the VGG-16 network [2] with batch normalization [60] and dropout [61] as the base classifier in all experiments. The network is optimized using SGD with an initial learning rate of 0.1, a momentum of 0.9, a weight decay of 0.0005, a batch size of 128, and a total training epoch of 300. The learning rate is decayed by 0.5 in every 25 epochs. For our method, we set the hyper-parameters $E_{\alpha} = 0, \alpha = 0.99$.

Main Results. The results of prior methods are cited from original papers and are summarized in Table 5. We see that our method achieves up to 50% relative improvements compared with all other methods under various coverage rates, on all datasets. Notably, Deep Gamblers also introduces an extra abstention class in their method but without applying model predictions. The improvement of our method comes from the use of model predictions in the training process.

6 IMPROVED REPRESENTATIONS LEARNING

6.1 Self-Supervised Self-Adaptive Training

Instantiation. We consider training images $\{x_{i}\}_{i}$, without label and use a deep network followed by a non-linear projection as encoder $f_{\theta}$. Then, we instantiate the meta Algorithm 1 for self-supervised learning as follows:

1) Target initialization. Since the labels are absent, each training target $t_{i}$ is randomly and independently drawn from a standard normal distribution.
2) Normalization function. We directly normalize each representation by dividing its $\ell_{2}$ norm.
3) Loss function. The loss function is implemented as the Mean Square Error (MSE) between the normalized model prediction $p_{i}$ and training target $t_{i}$.

The above instantiation of our meta algorithm, i.e., fitting models’ own accumulated representations, suffices to learn decent representations, as exhibited by the blue curve in Fig. 2. However, as discussed in Section 2.1 that the consistency of the training target also plays an essential role, especially in self-supervised representation learning, we introduce two components that further improve the representation learning with self-adaptive training.

Momentum Encoder and Predictor. We follow prior works [10], [62] to employ a momentum encoder $f_{\theta_{m}}$, whose parameters $\theta_{m}$ is also updated by the EMA scheme as

$$\theta_{m} = \beta \times \theta_{m} + (1 - \beta) \times \theta_{t}$$

(13)

With the slowly-evolving $f_{\theta_{m}}$, we can obtain a representation $z_{i} = f_{\theta_{m}}(x_{i})$,

(14)

and construct the target $t_{i}$, following the EMA scheme in Equation (2).

Furthermore, to prevent the model from outputting the same representation for every image in each iteration (i.e.,

| Dataset          | Coverage | Ours [54] | Deep Gamblers [54] | SelectiveNet [55] | SR [56] | MC-dropout [56] |
|------------------|----------|-----------|---------------------|-------------------|---------|-----------------|
| CIFAR10          | 100      | 6.05±0.20 | 6.12±0.09           | 6.79±0.03         | 6.79±0.03 | 6.79±0.03       |
|                  | 95       | 3.37±0.05 | 3.49±0.15           | 4.16±0.09         | 4.55±0.07 | 4.58±0.05       |
|                  | 90       | 1.93±0.09 | 2.19±0.12           | 2.43±0.08         | 2.89±0.03 | 2.92±0.01       |
|                  | 85       | 1.15±0.18 | 1.09±0.15           | 1.43±0.08         | 1.78±0.09 | 1.82±0.09       |
|                  | 80       | 0.67±0.10 | 0.66±0.11           | 0.86±0.06         | 1.05±0.07 | 1.08±0.05       |
|                  | 75       | 0.44±0.03 | 0.52±0.03           | 0.48±0.02         | 0.63±0.04 | 0.66±0.05       |
|                  | 70       | 0.34±0.06 | 0.43±0.07           | 0.32±0.01         | 0.42±0.06 | 0.43±0.05       |
|                  | 100      | 2.75±0.09 | 3.24±0.09           | 3.21±0.08         | 3.21±0.08 | 3.21±0.08       |
| SVHN             | 95       | 0.96±0.09 | 1.36±0.02           | 1.40±0.01         | 1.39±0.05 | 1.40±0.05       |
|                  | 90       | 0.60±0.05 | 0.76±0.05           | 0.82±0.01         | 0.89±0.04 | 0.90±0.04       |
|                  | 85       | 0.45±0.02 | 0.57±0.07           | 0.60±0.01         | 0.70±0.03 | 0.71±0.03       |
|                  | 80       | 0.43±0.01 | 0.51±0.05           | 0.53±0.01         | 0.61±0.02 | 0.61±0.01       |
|                  | 100      | 3.01±0.17 | 2.93±0.17           | 3.58±0.04         | 3.58±0.04 | 3.58±0.04       |
|                  | 95       | 1.25±0.05 | 1.23±0.12           | 1.62±0.05         | 1.91±0.08 | 1.92±0.06       |
| Dogs vs. Cats    | 90       | 0.59±0.04 | 0.59±0.13           | 0.93±0.01         | 1.10±0.08 | 1.10±0.05       |
|                  | 85       | 0.25±0.11 | 0.47±0.10           | 0.56±0.02         | 0.82±0.06 | 0.78±0.06       |
|                  | 80       | 0.15±0.06 | 0.46±0.08           | 0.35±0.09         | 0.68±0.05 | 0.55±0.02       |

Mean and standard deviation are calculated over 3 trials. The best entries and those overlap with them are marked bold.
collapsing), we further use a predictor $g$ to transform the output of encoder $f_0$ to prediction

$$p_i = g(f_0(x_i)), \quad (15)$$

where $g$ has the same number of output units as $f_0$.

Putting Everything Together. We $L_2$-normalize $p_i$ to $\tilde{p}_i = p_i/\|p_i\|_2$ and $t_i$ to $\tilde{t}_i = t_i/\|t_i\|_2$. Finally, the MSE loss between the normalized predictions and accumulated representations

$$L(p_i, t_i; f_0, g) = \frac{1}{m} \sum \|\tilde{p}_i - \tilde{t}_i\|^2_2 \quad (16)$$

is minimized to update the encoder $f_0$ and predictor $g$. We term this variant Self-Supervised Self-Adaptive Training, and summarize the pseudo-code in Algorithm 3 and the overall pipeline in Fig. 8. Our approach is straightforward to implement in practice and requires only single-view training, which significantly alleviates the heavy computation burden of data augmentation operations.

**Algorithm 3. Self-Supervised Self-Adaptive Training**

```plaintext`
Require: Data $\{x_i\}_i$; encoder $f_0$ and momentum encoder $f_{thrs}$; predictor $g$; momentum terms $\alpha = 0.7$, $\beta = 0.99$
1: Initialize $\theta_0 \leftarrow \theta$
2: Randomly initialize training target $\{t_i\}_i \sim \mathcal{N}(0, 1)$
3: repeat
4: Fetch augmented mini-batch data $\{x_i\}_i$
5: for $i = 1$ to $m$ (in parallel) do
6: $p_i = g(f_0(x_i))$
7: $z_i = f_{thrs}(x_i)$
8: $\tilde{z}_i = z_i/\|z_i\|_2$; $\tilde{t}_i = t_i/\|t_i\|_2$
9: $t_i \leftarrow \alpha \times \tilde{t}_i + (1 - \alpha) \times \tilde{z}_i$
10: end for
11: $L(p_i, t_i; f_0, g) = \frac{1}{m} \sum \|\tilde{p}_i - \tilde{t}_i\|^2_2$
12: Update $f_0$ and $g$ by SGD on $L(p_i, t_i; f_0, g)$
13: Update $\theta_m \leftarrow \beta \times \theta_m + (1 - \beta) \times \theta$
14: until end of training
```

Methodology Differences With Prior Works. BYOL [62] formulated the self-supervised training of deep models as predicting the representation of one augmented view of an image from the other augmented view of the same image. The self-supervised self-adaptive training shares some similarities with BYOL since both methods do not need to contrast with negative examples to prevent the collapsing issue. Instead of directly using the outputs of a momentum encoder as training targets, our approach uses the accumulated predictions as training targets, which contain all historical view information for each image. As a result, our approach requires only a single view during training, which is much more efficient as shown in Section 6.3 and Fig. 9. Besides, NAT [63] used an online clustering algorithm to assign noise as the training target for each image. Unlike NAT that fixes the noise while updating the noise assignment during training, our approach uses the noise as the initial training target and updates the noise by model predictions in the subsequent training process. InstDisc [13] used a memory bank to store the representation of each image in order to construct positive and negative samples for the contrastive objective. By contrast, our method gets rid of the negative samples and only matches the prediction with the training target of the same image (i.e., positive sample).

6.2 Bypassing Collapsing Issues

We note that there exist trivial local minima when we directly optimize the MSE loss between predictions and training targets due to the absence of negative pairs: the encoder $f_0$ can simply output a constant feature vector for every data point to minimize the training loss, a.k.a. collapsing issue [62]. Despite the existence of a collapsing solution, self-adaptive training intrinsically prevents the collapsing. The initial targets $\{t_i\}_i$ are different for different classes (under the supervised setting) or even for different images (under the self-supervised setting), enforcing the models to learn different representations for different classes/images. Our empirical studies in Fig. 1, Fig. 2 of the main body and Fig. 11 of the Appendix, available in the online supplemental material, strongly support that deep neural networks are able to learn meaningful information from corrupted data or even random noise, and bypass the model collapse. Based on this, our approach significantly improves the supervised learning (see Fig. 1a) and self-supervised learning (see the blue curve in Fig. 2) of deep neural networks.

6.3 Is Multi-View Training Indispensable?

The success of state-of-the-art self-supervised learning approaches [10], [62] largely hinges on the multi-view...
training scheme: they essentially use strong data augmentation operations to create multiple views (crops) of the same image and then match the representation of one view with the other views of the same image. Despite their promising performance, these methods suffer heavily from the computational burden of data pre-processing and the training on extra views. Concretely, as shown in Fig. 9, prior methods BYOL [62] and MoCo [10] incur doubled training time compared with standard supervised cross entropy training. In contrast, since our method requires only single-view training, its training is only slightly slower than the supervised method and is much faster than MoCo and BYOL. See Appendix B.5, available in the online supplemental material, for the detailed setup of this comparison.

We further conduct experiments to evaluate the performance of the multi-view training scheme on the CIFAR10 and STL10 datasets, which cast doubt on its necessity for learning a good representation. Concretely, we first pre-train a ResNet-18 encoder using different algorithms and then train a linear classifier on top of the encoder to evaluate the classification performance (see Section 7.1 for details). As shown in Table 6, although the performance of MoCo and BYOL is nearly halved on both datasets when using the single-view training scheme, the self-supervised adaptive training achieves comparable results under both settings. Moreover, our approach with single-view training even slightly outperforms the MoCo and BYOL with multi-view training. We attribute this superiority to the guiding principle of our algorithm: by dynamically incorporating the model predictions, each training target contains the relevant information about all the historical views of each image and, therefore, implicitly forces the model learning representations that are invariant to the historical views.

### Table 6: Necessity of the Multi-View Training Scheme in Self-Supervised Learning

| Method  | # Views | CIFAR10 | STL10 |
|---------|---------|---------|-------|
| MoCov2 [64] | 1       | 46.36   | 39.65 |
|         | 2       | 91.68   | 88.90 |
| BYOL [62]  | 1       | 44.36   | 42.60 |
|         | 2       | 91.78   | 89.60 |
| Ours    | 1       | 92.27   | 89.55 |
|         | 2       | 91.56   | 90.05 |

*Top 1 test accuracy (%) on CIFAR10 and STL10 datasets using the ResNet-18 backbone.*

### Table 7: Influence of the Momentum Encoder and Predictor

| Method  | Predictor | Momentum $\beta$ | Accuracy |
|---------|-----------|------------------|----------|
| BYOL [62] | ✓         | 0.0              | 25.64    |
|          |           | 0.99             | 26.87    |
|          |           | 0.999            | 72.44    |
| Ours    | ✓         | 0.0              | 85.22    |
|         |           | 0.99             | 91.78    |
|         |           | 0.999            | 90.68    |
|         | ✓         | 0.0              | 78.36    |
|         |           | 0.99 $\dagger$   | 79.68    |
|         |           | 0.999 $\dagger$  | 83.58    |
|         | ✓         | 0.0              | 90.18    |
|         |           | 0.99             | 92.27    |
|         |           | 0.999            | 90.92    |

*The performance is measured by the top1 test accuracy (%) on the CIFAR10 dataset using the ResNet-18 backbone. The entries with $\dagger$ indicate that $\alpha = 0.9$."

momentum can also improve the performance and bypass the collapsing issue. The results suggest that although both the predictor and the momentum encoder are indeed crucial for the performance of representation learning, either one of them with a proper configuration (i.e., the momentum term) suffices to avoid the collapse. We note that the latest version of [62] also found that the momentum encoder can be removed without collapse when carefully tuning the learning rate of the predictor. Our results, however, are obtained using the same training setting.

Moreover, we find that self-supervised self-adaptive training exhibits impressive resistance to collapsing, despite using only single-view training. Our approach can learn a decent representation even when the predictor and momentum encoder are both removed (see the seventh row of Table 7). We hypothesize that the resistance comes from the consistency of the training target due to our EMA scheme. This hypothesis is also partly supported by the observation that the learning of BYOL heavily depends on the slowly-evolving momentum encoder.

### 7 Application III: Linear Evaluation Protocol in Self-Supervised Learning

#### 7.1 Experimental Setup

**Datasets and Data Augmentations.** We conduct experiments on four benchmarks: CIFAR10/CIFAR100 [16] with 50k images; STL10 [65] with 105k images; ImageNet [20] with ~1.2m images. The choice of data augmentations follows prior works [10], [11]: we take a random crop from each image and resize it to a fixed size (i.e., $32 \times 32$ for CIFAR10/CIFAR100, $96 \times 96$ for STL10, and $224 \times 224$ for ImageNet); the crop is then randomly transformed by color jittering, horizontal flip, and grayscale conversion (and Gaussian blurring for ImageNet).

**Network Architecture.** The encoders $f_0$ and $f_m$ consist of a backbone of ResNet-18 [3]/ResNet-50 [3]/AlexNet [66] and a projector that is instantiated by a multi-layer perceptron (MLP). We use the output of the last global average pooling layer of the backbone as the extracted feature vectors. Following prior works [11], the output vectors of the backbone are transformed by the projector MLP to a dimension of 256.
Besides, the predictor \( g \) is also instantiated by an MLP with the same architecture as the projector in \( f_u \). In our implementation, all the MLPs have one hidden layer of size 4,096, followed by a batch normalization [60] layer and the ReLU activation [67].

**Self-Supervised Pre-Training Settings.** On the CIFAR10, CIFAR100, and STL10 datasets, we optimize the networks using an SGD optimizer with a momentum of 0.9 and a weight decay of 0.00005. We use a batch size of 512 for all methods in all experiments and train the networks for 800 epochs using 4 NVIDIA GTX 1080Ti GPUs. The hyper-parameters of our method are set to \( \alpha = 0.7, \beta = 0.99 \). On the ImageNet dataset, we use a batch size of 1024, a LARS [73] optimizer with a weight decay of 0.000001, and train the networks for 200 epochs on 8 GPUs. The base learning rate is set to 2.0 and is scaled linearly with respect to the base batch size 256 following [74]. For our method, we set the initial values of the momentum terms \( \alpha / \beta \) to 0.5/0.99, which are then increased to 0.8/1.0 following the prior work [62]. During training, the learning rate is warmed up for the first 30 epochs (10 epochs when using the ImageNet dataset) and is then adjusted according to the cosine annealing schedule [52].

**Linear Evaluation Protocol.** Following the common practice [10], [11], [62], we evaluate the representation learned by self-supervised pre-training using linear classification protocol. That is, we remove the projector in \( f_u \) and the predictor \( g \), fix the parameters of the backbone of the encoder \( f_u \), and train a supervised linear classifier on top of the features extracted from the encoder. On the ImageNet dataset, the linear classifier is trained for 100 epochs with a weight decay of 0.0, a batch size of 512, and an initial learning rate of 2.0 that is decayed to 0 according to the cosine annealing schedule [52]. On the rest datasets, the linear classifier is trained for 100 epochs with a weight decay of 0.0 and a batch size of 512. The initial learning rate is set to 0.4 and is decayed by a factor of 0.1 at the 60th and 80th training epochs. The performance is measured in terms of the top1 test accuracy of the linear classifier.

### 7.2 Comparison With the State-of-the-Art

We first conduct experiments on the CIFAR10/100 [16] and STL10 [65] datasets and compare our self-supervised self-adaptive training with three state-of-the-art methods, including the contrastive learning methods MoCo [10], SimCLR [11], and bootstrap method BYOL [62]. For fair comparisons, we use the same code base and the same experimental settings for all the methods, following their official open-sourced implementations. We carefully adjust the hyper-parameters on the CIFAR10 dataset for each method and use the same parameters on the rest datasets. Besides, we also conduct experiments using AlexNet as the backbone and compare the performance of our method with the reported results of prior methods. The results are summarized in Table 8. We can see that, despite using only single-view training, our self-supervised self-adaptive training consistently obtains better performance than all other methods on all datasets with different backbones.

Moreover, we follow the standard setups in self-supervised learning on the large-scale ImageNet [53] dataset (i.e., using ResNet-50 backbone, 200 pre-training epochs, and the linear evaluation protocol), and report the comparisons with the state-of-the-art methods in Table 9. We find that, while using 50% fewer augmented views during training, the single-view self-adaptive training achieves 67.4\% top1 accuracy, which is comparable with the two-view SimCLR and MoCoV2 and slightly worse than BYOL. We conjecture that, on the large-scale dataset, the training targets are updated less frequently than those on a smaller dataset, which results in performance degradation. Regarding this, to make a fair comparison, we also pre-train a ResNet-50 model based on the two-view variant of SAT, at a similar training cost as BYOL. From the results in Table 9. We can observe that the two-view variant of SAT (a) outperforms all other methods under the same setting by a large margin, and (b) even performs on par with BYOL and MoCoV3 trained by 300 epochs while outperforming the other methods with \( > 2 \times \) training epochs.

Finally, in addition to the above experiments, we also evaluate the pre-trained models on two downstream tasks, object detection and semantic segmentation on the Pascal

| Backbone | Method | CIFAR10 | CIFAR100 | STL10 |
|----------|--------|---------|----------|-------|
| SplitBrain [68] | 67.1 | 39.0 | - |
| DeepCluster [69] | 77.9 | 41.9 | - |
| InstDisc [13] | 70.1 | 39.4 | - |
| AND [70] | 77.6 | 47.9 | - |
| SeLa [71] | 83.4 | 57.4 | - |
| CMC [72] | - | - | 83.28 |
| Ours | 83.55 | 59.80 | 83.75 |
| MoCoV2 [64] | 93.20 | 69.48 | 91.95 |
| SimCLR [11] | 93.08 | 67.92 | 90.90 |
| BYOL [62] | 93.48 | 68.48 | 92.40 |
| Ours | 94.04 | 70.16 | 92.60 |

**TABLE 8** Linear Evaluation Protocol on CIFAR10, CIFAR100, and STL10 Datasets Using Different Backbones

| Method | # Views | Epochs | Linear Acc. (%) |
|--------|--------|--------|-----------------|
| InstDisc [13] | 2 | 200 | 56.5 |
| MoCo [10] | 2 | 200 | 60.6 |
| CPCv2 [76] | 2 | 200 | 63.8 |
| SimCLR [11] | 2 | 200 | 66.6 |
| MoCoV2 [64] | 2 | 200 | 67.5 |
| BYOL [62] | 2 | 200 | 70.6 |
| SwAV [12] | 2 | 200 | 69.1 |
| SimSiam [75] | 2 | 200 | 70.0 |
| Ours | 2 | 200 | 72.8 |
| CMC [72] | 2 | 240 | 60.0 |
| BYOL [62] | 2 | 300 | 72.4 |
| BarlowTwins [77] | 2 | 300 | 71.4 |
| MoCoV3 [78] | 2 | 300 | 72.8 |
| SeLa [71] | 2 | 400 | 61.5 |
| SeLa2 [71] | 2 | 400 | 67.2 |
| DeepClusterv2 [69] | 2 | 400 | 70.2 |
| SwAV [12] | 2 | 400 | 70.1 |
| PIRL [79] | 2 | 800 | 63.6 |
| SimCLRv2 [80] | 2 | 800 | 71.7 |

The entries marked with \( \dagger \) are cited from [75]; those marked by \( \ddagger \) are cited from [12].

**TABLE 9** Linear Evaluation Protocol on ImageNet Dataset Using the ResNet-50 [3] Backbone

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
VOC dataset [83]. We follow exactly the same settings as in prior works [10], [75] for fair comparisons, where all models are pre-trained for 200 epochs on ImageNet. We use the Faster R-CNN [81] architecture with the ResNet-50 C4 backbone [82] for detection and the ResNet-50 dilated FCN [5] architecture for segmentation. The experiments are conducted on the opensource codebases detectron2 [82]/mssegmentation [84] for the detection/segmentation tasks. According to the results in Table 10, self-adaptive training performs on par with or even better than state-of-the-art self-supervised learning methods on both transfer learning tasks. Moreover, we can also observe that SAT consistently achieves superior performance to the supervised baseline.

7.3 Sensitivity of Self-Supervised Self-Adaptive Training

Sensitivity to Hyper-Parameters. We study how the two momentum parameters $\alpha$ and $\beta$ affect the performance of our approach and report the results in Table 11. By varying one of the parameters while fixing the other, we observe that self-supervised self-adaptive training performs consistently well, which suggests that our approach is not sensitive to the choice of hyper-parameters.

Sensitivity to Data Augmentation. Data augmentation is one of the most essential ingredients of recent self-supervised learning methods: a large body of these methods, including ours, formulate the training objective as learning representations that encode the shared information across different views generated by data augmentation. As a result, prior methods, like MoCo and BYOL, fail in the single-view training setting. Self-supervised self-adaptive training, on the other hand, maintains a training target for each image, which contains all historical view information of this image. Therefore, we conjecture that our method should be more robust to data augmentations. To validate our conjecture, we evaluate our method and BYOL under the settings that some of the augmentation operators are removed. The results are shown in Fig. 10a. Removing any augmentation operators hurts the performance of both methods while our method is less affected. Specifically, when all augmentation operators except random crop and flip are removed, the performance of BYOL drops to 86.6% while our method still obtains 88.6% accuracy.

8 Related Works

Generalization of Deep Networks. Previous work [14] systematically analyzed the capability of deep networks to overfit random noise. Their results show that traditional wisdom fails to explain the generalization of deep networks. Another line of work [32], [33], [34], [35], [36], [37], [38] observed an intriguing double-descent risk curve from the bias-variance trade-off. [32], [33] claimed that this observation challenges the conventional U-shaped risk curve in the textbook. Our work shows that this observation may stem from overfitting to noise; the phenomenon vanishes by a properly designed training process such as self-adaptive training. To improve the generalization of deep networks, [43], [85] proposed label smoothing regularization that uniformly distributes $\epsilon$ of labeling weight to all classes and uses this soft label for training, [45] introduced mixup augmentation that extends the training distribution by dynamic interpolations between random paired input images and the associated targets during training. This line of research is similar to ours as both methods use soft labels in the training. However, self-adaptive training is able to recover true labels from noisy labels and is more robust to noise.

Robust Learning From Corrupted Data. Aside from the preprocessing-training approaches that have been discussed in the last paragraph of Section 3.1, there have also been many other works on learning from noisy data. To name a few, [19], [86] showed that deep neural networks tend to fit clean samples first and overfitting to noise occurs in the later stage of training, [19] further proved that early stopping can mitigate the issues that are caused by label noise. [87], [88] incorporated model predictions into training by simple interpolation of labels and model predictions. We demonstrate that our exponential moving average and sample re-weighting schemes enjoy superior performance. Other works [46], [47] proposed alternative loss functions to cross

### Table 10: Parameters Sensitivity on the CIFAR10 Dataset

| $\alpha$ | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|
| Fix $\beta = 0.99$ | 91.68 | 91.76 | 92.27 | 92.08 | 91.68 |
| Fix $\beta = 0.9$ | 0.9 | 0.95 | 0.99 | 0.995 | 0.999 |

**Sensitivity to Training Batch Size.** Recent contrastive learning methods require large batch size training (e.g., 1024 or even larger) for optimal performance, due to the need of comparing with massive negative samples. BYOL does not use negative samples and suggests that this issue can be mitigated. Here, since our method also gets rid of the negative samples, we make direct comparisons with BYOL at different batch sizes to evaluate the sensitivity of our method to batch size. For each method, the base learning rate is linearly scaled according to the batch size while the rest settings are kept unchanged. The results are shown in Fig. 10b. We can see that our method exhibits a smaller performance drop than BYOL at various batch sizes. Concretely, the accuracy of BYOL drops by 2% at batch size 128 while that of ours drops by only 1%.

| Method | # Views | VOC 07+12 Det. AP | VOC 12 Seg. AP$_50$ | VOC 12 Seg. AP$_75$ | mIoU |
|---|---|---|---|---|---|
| Supervised$^+$ | 1 | 53.5 | 81.3 | 58.8 | 67.7$^+$ |
| SimCLR [11]$^+$ | 2 | 55.5 | 81.8 | 61.4 | - |
| MoCov2 [64] | 2 | 57.0 | 82.4 | 63.6 | 66.7$^+$ |
| SwAV [12]$^+$ | 2 | 55.4 | 81.5 | 61.4 | - |
| BYOL [62]$^+$ | 2 | 55.3 | 81.4 | 61.1 | - |
| SimSiam [73]$^+$ | 2 | 56.4 | 82.0 | 62.8 | - |
| Ours | 1 | 55.2 | 81.5 | 60.9 | 64.8 |
| Ours | 2 | 56.4 | 82.5 | 63.0 | 68.2 |

All entries are based on the Faster R-CNN [81] architecture with the ResNet-50 C4 backbone [82] for detection, and are based on ResNet-50 FCN [5] for segmentation. $^+$: results cited from [75]. $^*$: reproduction based on the publicly available checkpoint.
entropy that are robust to label noise. They are orthogonal to ours and are ready to cooperate with our approach as shown in Appendix C.4, available in the online supplemental material. Beyond the corrupted data setting, recent works [89], [90] propose a self-training scheme that also uses model predictions as training targets. However, they suffer from the heavy cost of multiple iterations of training, which is avoided by our approach. Temporal Ensembling [91] incorporated the ‘ensemble’ predictions as pseudo-labels for training. Different from ours, Temporal Ensembling focuses on the semi-supervised learning setting and only accumulates predictions for unlabeled data.

Self-Supervised Learning. Aiming to learn powerful representations, most self-supervised learning approaches typically first solve a proxy task without human supervision. For example, prior works proposed recovering input using auto-encoder [92], [93], generating pixels in the input space [94], [95], predicting rotation [96] and solving jigsaw [97]. Recently, contrastive learning methods [10], [11], [13], [72], [98], [99] significantly advanced self-supervised representation learning. These approaches essentially used strong data augmentation techniques to create multiple views (crops) of the same image and discriminated the representation of different views of the same images (i.e., positive samples) from the views of other images (i.e., negative samples). Bootstrap methods eliminated the discrimination of positive and negative data pairs: the works of [69], [71] alternatively performed clustering on the representations and then used the cluster assignments as classification targets to update the model; [12] swapped the cluster assignments between the two views of the same image as training targets; [62] simply predicted the representation of one view from the other view of the same image; [63] formulated the self-supervised training objective as predicting a set of predefined noise. Our work follows the path of bootstrap methods. Going further than them, self-adaptive training is a general training algorithm that bridges supervised and self-supervised learning paradigms. Our approach casts doubt on the necessity of the costly multi-view training and works well with the single-view training scheme.

9 Conclusion

In this paper, we explore the possibility of a unified framework to bridge the supervised and self-supervised learning of deep neural networks. We first analyze the training dynamic of deep networks under these two learning settings and observe that useful information from data is distilled to model predictions. The observation occurs broadly even in the presence of data corruptions and the absence of labels, which motivates us to propose Self-Adaptive Training—a general training algorithm that dynamically incorporates model predictions into the training process. We demonstrate that our approach improves the generalization of deep neural networks under various kinds of training data corruption and enhances the representation learning using accumulated model predictions. Finally, we present three applications of self-adaptive training on learning with noisy labels, selective classification, and linear evaluation protocol in self-supervised learning, where our approach significantly advances the state-of-the-art.

Acknowledgments

The material in this paper was presented in part at Thirty-fourth Conference on Neural Information Processing Systems, December 2020 [1].

References

[1] L. Huang, C. Zhang, and H. Zhang, “Self-adaptive training: Beyond empirical risk minimization,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, Art. no. 1624.
[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Representations, 2015.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186.
[5] G. Huang, Z. Liu, L. Sundaram, X. Tang, and M. Wang, “DenseNet: Convolutional networks with many feature maps in a single layer,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4700–4708.
[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, 2014, p. 1929.
[7] V. Kolchinsky, “Attention, please: Understanding transformer networks,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–16.
[8] K. He, Y. Sun, and X. Chen, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1026–1034.
[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186.
[10] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9729–9738.
P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. M. Baines et al., "FairScale: A general purpose modular pytorch framework for contrastive learning of visual representations," in Proc. Int. Conf. Mach. Learn., 2020, Art. no. 12115.

C. Szegedy et al., "Intriguing properties of neural networks," in Proc. Int. Conf. Mach. Learn. Representations, 2014.

J. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in Proc. Int. Conf. Mach. Learn., 2019, pp. 7472–7482.

F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in Proc. Int. Conf. Mach. Learn., 2020, pp. 2206–2216.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in Proc. Int. Conf. Learn. Representations, 2017.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2818–2826.

G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qi, "Making deep neural networks robust to label noise: A loss correction approach," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 1944–1952.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in Proc. Int. Conf. Learn. Representations, 2018.

Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in Proc. Int. Conf. Neural Inf. Process. Syst., 2018, pp. 8778–8788.

K. Wang, X. Ma, and Y. Qi, "Symmetric cross entropy for robust learning with noisy labels," in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 322–330.

S. Thulasidasan, T. Bhattacharyya, J. Bilmes, G. Chennupati, and J. Molyb-Yusof, "Combating label noise in deep learning using abstention," in Proc. Int. Conf. Mach. Learn., 2019, pp. 6234–6243.

S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, "Early-learning regularization prevents memorization of noisy labels," in Proc. Int. Conf. Learn. Representations, 2020, Art. no. 1707.

Z. Zagoruyko and N. Komodakis, "Wide residual networks," in Proc. Int. Conf. Learn. Representations, 2016, pp. 319–327.

A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 8024–8035.

I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in Proc. Int. Conf. Learn. Representations, 2017.

O. Russakovsky et al., "ImageNet large scale visual recognition challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–225, 2015.

Z. Liu, Z. Wang, P. P. Liang, R. Salakhutdinov, L.-P. Morency, and M. Ueda, "Deep gamblers: Learning to abstain with portfolio theory," in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 22287–22297.

Y. Geifman and R. El-Yaniv, "SelectiveNet: A deep neural network with an integrated reject option," in Proc. Int. Conf. Mach. Learn., 2019, pp. 2151–2159.

Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 4878–4887.

R. El-Yaniv and Y. Wiener, "On the foundations of noise-free selective classification," J. Mach. Learn. Res., vol. 11, no. May, pp. 1605–1614, 2010.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn., 2011.

F. Croce and M. Hein, "Statistical mechanics of learning: Generalization," in The Handbook of Brain Theory and Neural Networks. Cambridge, MA, USA: MIT Press, 1995, pp. 922–925.

M. Opper, "Learning to generalize," Front. Life, vol. 3, no. 2, pp. 763–775, 2001.

W. S. Geirhofer, M. A. Saxe, and H. Sompolinsky, "High-dimensional dynamics of generalization error in neural networks," New J. Phys., vol. 132, pp. 428–446, 2020.

S. Spigler, M. Geiger, S. d’Ascoli, L. Sagun, G. Birolti, and M. Wyart, "A jamming transition from under-to over-parametrization affects loss landscape and generalization," 2018, arXiv:1810.09665.
X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” 2020, arXiv:2003.04297.

A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in Proc. Int. Conf. Artif. Intell. Statist., 2011, pp. 215–223.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proc. Int. Conf. Mach. Learn., 2010, pp. 807–814.

R. Zhang, P. Isola, and A. A. Efros, “Split-brain autoencoders: Unsupervised learning by cross-channel prediction,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 645–654.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 132–149.

J. Huang, Q. Dong, S. Gong, and X. Zhu, “Unsupervised deep learning by neighbourhood discovery,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 2849–2858.

S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 91–99.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, “Exploring simple siamese representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 15745–15753.

O. Henaff, “Data-efficient image recognition with contrastive predictive coding,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 4182–4192.

Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 776–794.

Y. You, I. Gitman, and B. Ginsburg, “Large batch training of convolutional networks,” 2017, arXiv:1708.03588.

P. Goyal et al., “Accurate, large minibatch SGD: Training ImageNet in 1 hour,” 2017, arXiv:1706.02677.

X. Chen and K. He, “Exploring simple siamese representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6706–6716.

T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, pp. 2243–2255.

S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2015, pp. 1–9.

Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” 2019. [Online]. Available: https://github.com/facebookresearch/detectron2

M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” Int. J. Comput. Vis., vol. 118, no. 1, pp. 98–136, Jan. 2016.

Contributors, “MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark,” 2020. [Online]. Available: https://github.com/open-mmlab/mmsegmentation

G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton, “Regularizing neural networks by penalizing confident output distributions,” 2017, arXiv:1701.06548.

D. Arpit et al., “A closer look at memorization in deep networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 233–242.

S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “Training deep neural networks on noisy labels with bootstrapping,” in Proc. Workshop Int. Conf. Learn. Representation, 2015.

B. Dong, J. Hou, Y. Lu, and Z. Zhang, “Distillation $\approx$ early stopping? Harvesting dark knowledge utilizing anisotropic information retrieval for overparameterized neural network,” 2019, arXiv:1910.01255.

T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, “Born again neural networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1607–1616.

Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, “Self-training with noisy student improves ImageNet classification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 10 684–10 695.

S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in Proc. Int. Conf. Learn. Representations, 2017.

F. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in Proc. Int. Conf. Mach. Learn., 2008, pp. 1096–1103.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2536–2544.

D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in Proc. Int. Conf. Learn. Representation, 2014.

J. Goodfellow et al., “Generative adversarial nets,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2014, pp. 2672–2680.

S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” in Proc. Int. Conf. Learn. Representation, 2018.

M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving Jigsaw puzzles,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 69–84.

A. V. D. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.

J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi, “Prototypical contrastive learning of unsupervised representations,” in Proc. Int. Conf. Learn. Representations, 2021.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations, 2015.

D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in Proc. Int. Conf. Learn. Representations, 2018.

Lang Huang received the master’s degree from the Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, China, in 1995. He is currently a research professor with the Key Laboratory of Machine Perception (MOE), School of Intelligence Science and Technology, Peking University. His research interests include computer vision, image processing, machine learning, and pattern recognition.

Chao Zhang (Member, IEEE) received the PhD degree in electrical engineering from Beijing Jiaotong University, Beijing, China, in 1995. He is currently a research professor with the Key Laboratory of Machine Perception (MOE), School of Intelligence Science and Technology, Peking University. His research interests include computer vision, image processing, machine learning, and pattern recognition.

Hongyang Zhang (Member, IEEE) received the PhD degree from Machine Learning Department, Carnegie Mellon University, in 2019. He is currently an assistant professor with David R. Cheriton School of Computer Science, University of Waterloo, and a faculty affiliated with Vector Institute for AI. Before joining University of Waterloo, he was a postdoctoral research associate with Toyota Technological Institute at Chicago. His research interests include machine learning, AI security, and trustworthy AI.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.