DisturbLabel: Regularizing CNN on the Loss Layer

Lingxi Xie[1] * Jingdong Wang[2] Zhen Wei[3] Meng Wang Qi Tian[5]
[1]Department of Statistics, University of California, Los Angeles, Los Angeles, CA, USA
[2]Microsoft Research, Beijing, China
[3]Department of Computer Science, Shanghai Jiao Tong University, Shanghai, USA
[4]School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China
[5]Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA
[1]198808xc@gmail.com [2]jingdw@microsoft.com
[3]94hazelnut@gmail.com [4]wangmeng@hfut.edu.cn [5]qitian@cs.utsa.edu

Abstract

During a long period of time we are combating over-fitting in the CNN training process with model regularization, including weight decay, model averaging, data augmentation, etc. In this paper, we present **DisturbLabel***, an extremely simple algorithm which randomly replaces a part of labels as incorrect values in each iteration. Although it seems weird to intentionally generate incorrect training labels, we show that DisturbLabel prevents the network training from over-fitting by implicitly averaging over exponentially many networks which are trained with different label sets. To the best of our knowledge, DisturbLabel serves as the first work which adds noises on the* **loss layer***. Meanwhile, DisturbLabel cooperates well with Dropout to provide complementary regularization functions. Experiments demonstrate competitive recognition results on several popular image recognition datasets.*

1. Introduction

Deep Convolutional Neural Networks (CNNs) [16] have shown significant performance gains in image recognition [14]. The large image repository, ImageNet [3], and the high-performance computational resources such as GPUs played very important roles in the resurgence of CNNs. Meanwhile, a number of research attempts on various aspects have been made to learn the deep hierarchical structure better [28][32] and faster [9].

Many regularization techniques that this paper is interested in have been developed to prevent neural network from over-fitting, *e.g.*, the ℓ_2-regularization over the *weights* (*a.k.a.*, weight decay) [13], Dropout [7] which sets a randomly-selected subset of *activations* to zero within the (hidden) layers during training, DropConnect [36] which sets randomly-selected *weights* within the network to zero during training, data argumentation which manipulates the *input* [2][14], and early stopping the iteration [24].

In this paper, we propose **DisturbLabel** which imposes the regularization within the **loss layer**. In each training iteration, DisturbLabel randomly selects a small subset of samples (from those in the current mini-batch) and randomly sets their ground-truth labels to be incorrect, which results in a noisy loss function and, consequently, noisy gradient back-propagation. To the best of our knowledge, this is the first attempt to regularize the CNN on the *loss layer*. We show that DisturbLabel is an alternative approach to combining a large number of models that are trained with different noisy data. Experimental results show that DisturbLabel achieves comparable performance with Dropout and that it, in conjunction with Dropout, obtains better performance on several image classification benchmarks.

The rest of this paper is organized as follows. Section 2 briefly introduces related works. The DisturbLabel algorithm is presented in Section 3. The discussions and the cooperation with Dropout are presented in Sections 4 and 5, respectively. Experimental results are shown in Section 6, and we conclude our work in Section 7.

2. Related Works

The recent great success of CNNs in image recognition has benefitted from and inspired a wide range of research efforts, such as designing deeper network structures [28][32], exploring or learning non-linear activation functions [4][20][6], developing new pooling operations [37][5][17], introducing better optimization techniques [18], regularization techniques preventing the network from over-fitting [7][36], and so on.

*This work was done when Lingxi Xie and Zhen Wei were interns at MSR.

Method	weight decay	Dropout	DropConnect	data augmentation	stochastic pooling	**DisturbLabel**
Units	weights	hidden nodes	weights	input nodes	pooling layer	**loss layer**

Table 1. Comparison with different CNN regularization techniques. Please refer to the texts for detailed references.

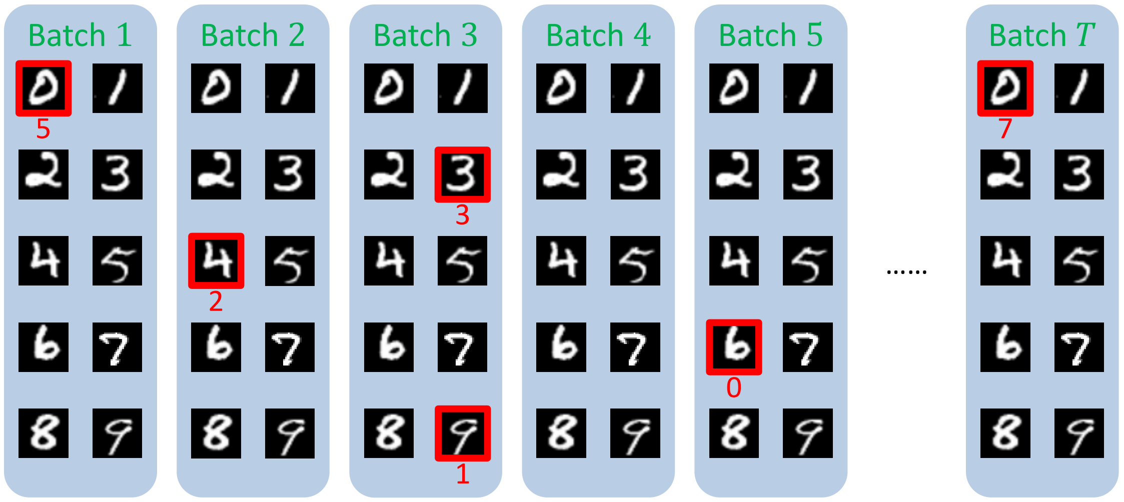

Figure 1. An illustration of the DisturbLabel algorithm ($\alpha = 10\%$). A mini-batch of 10 training samples is used as the toy example. Each sample is disturbed with the probability α. A disturbed training sample is marked with a red frame and the disturbed label is written below the frame. Even if a sample is disturbed, the label may remain unchanged (*e.g.*, the digit 3 in the 3rd mini-batch).

Avoiding over-fitting is a major challenge against training robust CNN models. The early solutions include reducing the model complexity by using fewer parameters or sharing parameters [15], early stopping [24] in which training is stopped before convergence, weight decay [13] which can be interpreted as a way of constraining the parameters using the ℓ_2-regularization and is now widely-adopted.

Recently, various regularization methods have been introduced. Data augmentation generates more training data as the input of the CNN by randomly cropping, rotating and flipping the *input images* [14][2], and adding noises to the image *pixels* [25]. Dropout [7] randomly discards a part of neuron response (the *hidden neurons*) during the training and only updates the remaining weights in each mini-batch iteration. DropConnect [36] instead only updates a randomly-selected subset of *weights*. Stochastic Pooling [37] changes the deterministic *pooling operation* and randomly samples one input as the pooling result in probability during training. Probabilistic Maxout [29] instead turns the *Maxout operation* [4] to stochastic. In contrast, our approach (DisturbLabel) imposes the regularization at the *loss layer*. A comparison of different regularization methods is summarized in Table 1.

There are other research works that are related to noisy labeling. [31] explores the performance of discriminatively-trained CNNs on the noisy data, where there are some freely available labels for each image which may or may not be accurate. In contrast, our approach (DisturbLabel) assumes the labeling is correct and randomly changes the labels in a small probability in each mini-batch iteration, which means that the label of a training sample is correct in most iterations. Different from other works adding noise to the input unit [34][35][21][22], our work is a way of regularizing a neural network by adding noise to its loss layer (related to the output unit).

3. The DisturbLabel Algorithm

We start with the typical CNN training process. The image dataset is given as $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, in which the data point is a D-dimensional vector $\mathbf{x}_n \in \mathbb{R}^D$, and the label is a C-dimensional vector $\mathbf{y}_n = [0, \cdots, 0, 1, 0, \cdots, 0]^\top$ with the entry of the corresponding class being 1 and all the others being 0. The goal is to train a CNN model $\mathbb{M}$: $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^C$, in which $\boldsymbol{\theta}$ represents the model parameters.

$\boldsymbol{\theta}$ is often initialized as a set of white noises $\boldsymbol{\theta}_0$, then updated using stochastic gradient descent (SGD) [1]. The t-th iteration of SGD updates the current parameters $\boldsymbol{\theta}_t$ as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \gamma_t \cdot \frac{1}{|\mathcal{D}_t|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_t} \nabla_{\boldsymbol{\theta}_t} [l(\mathbf{x}, \mathbf{y})]. \quad (1)$$

Here, $l(\mathbf{x}, \mathbf{y})$ is a loss function, *e.g.*, softmax or square loss. $\nabla_{\boldsymbol{\theta}_t} [l(\mathbf{x}, \mathbf{y})]$ is computed using gradient back-propagation. $\mathcal{D}_t$ is a mini-batch randomly drawn from the training dataset $\mathcal{D}$, and γ_t is the learning rate.

DisturbLabel works on each mini-batch independently. It performs an extra sampling process, in which a disturbed label vector $\widetilde{\mathbf{y}} = [\widetilde{y}_1, \widetilde{y}_2, \ldots, \widetilde{y}_C]^\top$ is randomly generated for each data $(\mathbf{x}, \mathbf{y})$ from a Multinoulli (generalized Bernoulli) distribution $\mathcal{P}(\alpha)$:

$$\left\{ \begin{array}{rcl} \widetilde{c} & \sim & \mathcal{P}(\alpha), \\ \widetilde{y}_{\widetilde{c}} & = & 1, \\ \widetilde{y}_{\widetilde{i}} & = & 0, \qquad \forall i \neq \widetilde{c}. \end{array} \right. \tag{2}$$

The Multinoulli distribution $\mathcal{P}(\alpha)$ is defined as $p_c = 1 - \frac{C-1}{C} \cdot \alpha$ and $p_i = \frac{1}{C} \cdot \alpha$ for $i \neq c$. α is the *noise rate*, and c is the ground-truth label (*i.e.*, in the true label vector $\mathbf{y}$, $y_c = 1$). In other words, we disturb each training sample with the probability α. For each disturbed sample, the label is randomly drawn from a uniform distribution over $\{1, 2, \ldots, C\}$, regardless of the true label.

Algorithm 1 DisturbLabel

```
1: Input: $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, noise rate $\alpha$.
2: Initialization: a network model $\mathbb{M}$: $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_0) \in \mathbb{R}^C$;
3: for each mini-batch $\mathcal{D}_t = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^M$ do
4:     for each sample $(\mathbf{x}_m, \mathbf{y}_m)$ do
5:         Generate a disturbed label $\widetilde{\mathbf{y}}_m$ with Eqn (2);
6:     end for
7:     Update the parameters $\boldsymbol{\theta}_t$ with Eqn (1);
8: end for
9: Output: the trained model $\mathbb{M}'$: $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_T) \in \mathbb{R}^C$.
```

The pseudo codes of DisturbLabel are listed above. An illustration of DisturbLabel is shown in Figure 1.

3.1. The Effect of the Noise Rate

The noise rate α determines the expected fraction ($\frac{C-1}{C} \cdot \alpha$) of training data in a mini-batch which are assigned incorrect labels. When $\alpha = 0\%$, there are no noises involved, and the algorithm degenerates to the ordinary case. When $\alpha \to 100\%$, we are actually discarding most of the labels and the training process becomes nearly unsupervised (as the probability assigned to any class is nearly $\frac{1}{C}$). It is often necessary to set a relatively small α, although it is possible to obtain an efficient network with a rather large α (*e.g.*, training the **LeNet** with $\alpha = 90\%$ achieves $< 2\%$ testing error rate on **MNIST**).

We evaluate the recognition accuracy on both **MNIST** and **CIFAR10**, by training the **LeNet** with different noise rates α, We summarize the results in Figures 2 and 3, respectively. It is observed that DisturbLabel with a relatively small α (e.g., 10% or 20%) achieves higher recognition accuracy than the model without regularization ($\alpha = 0\%$). This verifies that DisturbLabel does improve the generalization ability of the trained CNN model (Section 3.2 provides an empirical verification that the improvement comes from preventing over-fitting). When the noise rate α goes up to 50%, DisturbLabel significantly causes the network to converge slower, meanwhile produces lower recognition accuracy compared to smaller noises, which is reasonable as the labels of the training data are not reliable enough.

3.2. DisturbLabel as a Regularizer

We empirically show that DisturbLabel is able to prevent the network training from over-fitting. Figures 4 and 5 show the results on the **MNIST** and **CIFAR10** datasets using the normal training without regularization and the DisturbLabel algorithm over the same CNN structure. In addition, we also report the results when training the CNN with Dropout which is an alternative approach of CNN regularization.

We can observe that without regularization, the training error quickly drops to quite a low level, *e.g.*, almost 0% on **MNIST** and close to 3% on **CIFAR10**, but the testing error stops decreasing at a high level. In contrast, the training error with DisturbLabel drops slower and is consistently larger than that without regularization. However, the testing error becomes lower, verifying that the improvement comes from preventing over-fitting. Similar results are also obtained in the case with the Dropout regularization. This reveals that training a CNN with regularization (either DisturbLabel or Dropout) yields stronger generalization ability.

4. Discussions

In this section, we provide discussions on the DisturbLabel algorithm. We show the difference between DisturbLabel and the strategies that training networks with noisy data or using soft labels, and interpret DisturbLabel as an alternative way of model ensemble and data augmentation.

4.1. Difference from Training on Noisy Data

We show that DisturbLabel is different from training directly with noisy data. By "noisy data" we mean to augment the original training set $\mathcal{D} = \left\{(\mathbf{x}_n, \mathbf{y}_n)_{n=1}^N\right\}$ to a larger set $\widetilde{\mathcal{D}} = \left\{\left(\mathbf{x}_n, \mathbf{y}_n^{(1)}\right)_{n=1}^N, \ldots, \left(\mathbf{x}_n, \mathbf{y}_n^{(K)}\right)_{n=1}^N\right\}$, so that for each data point $\mathbf{x}_n$, among $\left\{\mathbf{y}_n^{(1)}, \ldots, \mathbf{y}_n^{(K)}\right\}$, an expectation of $K \cdot (1 - \frac{C-1}{C} \cdot \alpha)$ labels are equal to the correct label $\mathbf{y}_n^{(c)}$ (assuming that K is large enough to cover all the possible incorrect labels) where α is the noise level. Then we can normally train the CNN over the augmented noisy data, *e.g.*, with mini-batch stochastic gradient descent.

There is a clear difference between these two methods: one mini-batch in training with noisy data may contain a data point $\mathbf{x}$ with both correct and incorrect labels, while one mini-batch in DisturbLabel only contains a training one time with an either correct or incorrect label.

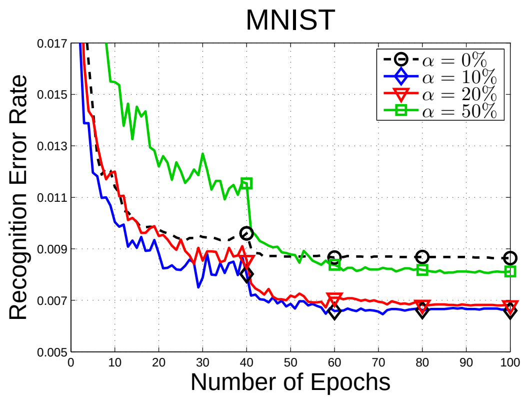

Figure 2. **MNIST** recognition error rate with respect to the noise level α (using **LeNet**).

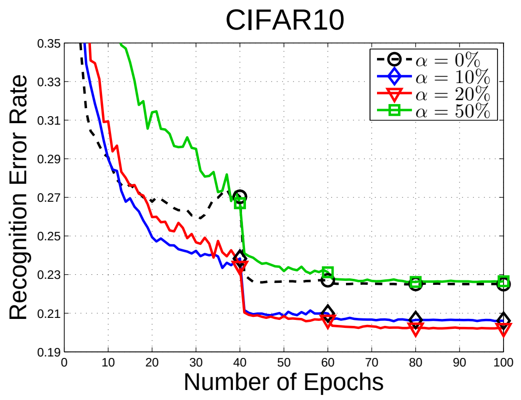

Figure 3. **CIFAR10** recognition error rate with respect to the noise level α (using **LeNet**).

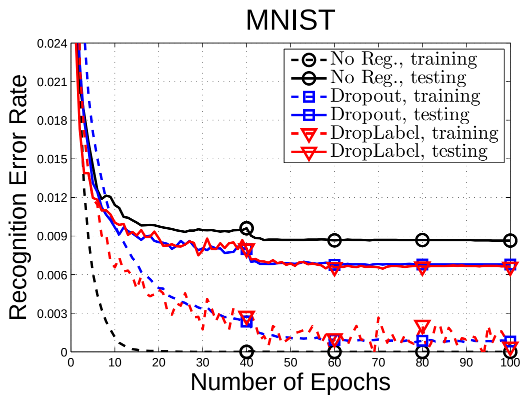

Figure 4. **MNIST** training vs. testing error on the **LeNet** with: no regularization, Dropout and DisturbLabel.

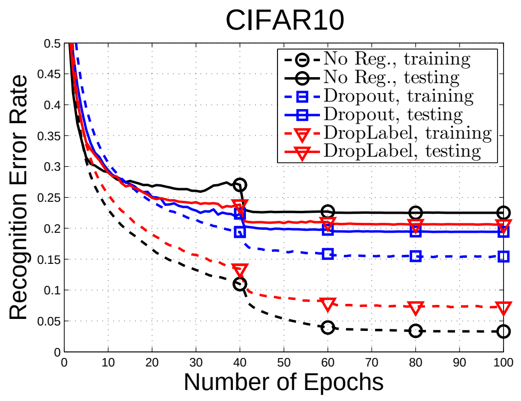

Figure 5. **CIFAR10** training vs. testing error on the **LeNet** with: no regularization, Dropout and DisturbLabel.

4.2. Difference from Soft Labeling

Consider the soft labeling problem, where each data point $\mathbf{x}$ is assigned to a label with probability. Denote the label vector by $\mathbf{y}'$, which is a C-dimensional vector, with the c-th dimension being $1 - \frac{C-1}{C} \cdot \alpha$ and all others being $\frac{\alpha}{C}$, and α is the noise level. Then we can normally train the CNN over the same data $\mathbf{x}$ and the soft label $\mathbf{y}'$, using which the loss function is changed. Here we consider the standard cross-entropy loss function, and the derivation of other choices (*e.g.*, logistic loss, square loss, *etc.*) is similar.

The gradient for a training sample $(\mathbf{x}, \mathbf{y}')$ is:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \left|\mathbf{y}_n' - \mathbf{f}\right|_1 = -\left[\frac{\partial \left|\mathbf{y}_n' - \mathbf{f}\right|_1}{\partial \left(\mathbf{y}_n' - \mathbf{f}\right)}\right]^\top \cdot \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \tag{3}$$

However in DisturbLabel, the gradient is computed as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \left|\widetilde{\mathbf{y}}_n - \mathbf{f}\right|_1 = -\left[\frac{\partial \left|\widetilde{\mathbf{y}}_n - \mathbf{f}\right|_1}{\partial \left(\widetilde{\mathbf{y}}_n - \mathbf{f}\right)}\right]^\top \cdot \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \tag{4}$$

The empirical evaluation on the **MNIST** and **CIFAR10** datasets is shown in Figures 6 and 7, respectively. One can observe that using soft labels generates nearly the same accuracy compared to ordinary training, whereas DisturbLabel significantly improves recognition accuracy. It is not surprising that DisturbLabel is not equivalent to soft label though the expectation of the gradient in DisturbLabel is equal to the gradient in soft label (as $\mathbb{E}(\widetilde{\mathbf{y}}) = \mathbf{y}'$). This reveals that using soft labels does not bring the regularization ability as DisturbLabel does, which is also validated in the distillation solution [8], similar to the soft labeling, though it brings an advantage, making the network converge faster.

4.3. Interpretation as Model Ensemble

We show that DisturbLabel can be interpreted as an implicit model ensemble. Consider a normal noisy dataset $\widetilde{\mathcal{D}} = \left\{(\mathbf{x}_n, \widetilde{\mathbf{y}}_n)_{n=1}^N\right\}$, which is generated by assigning an incorrect label $\widetilde{\mathbf{y}}_n \neq \mathbf{y}_n$ to $\mathbf{x}_n$ with a probability α for each data point in $\mathcal{D}$. Combining neural network models that are trained on different noisy sets is usually helpful [8]. However, separately training nets is prohibitively expensive as there are exponentially many noisy datasets. Even if we

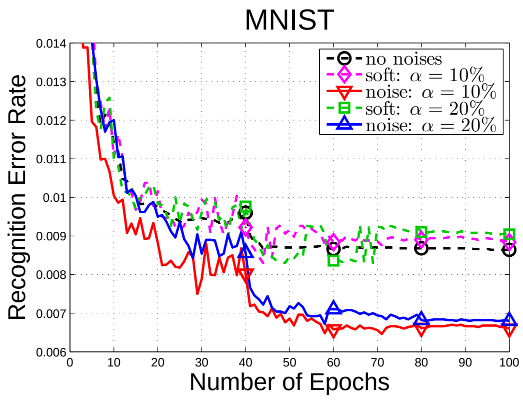

Figure 6. **MNIST** recognition error rate with soft labels and noisy labels (using the **LeNet**).

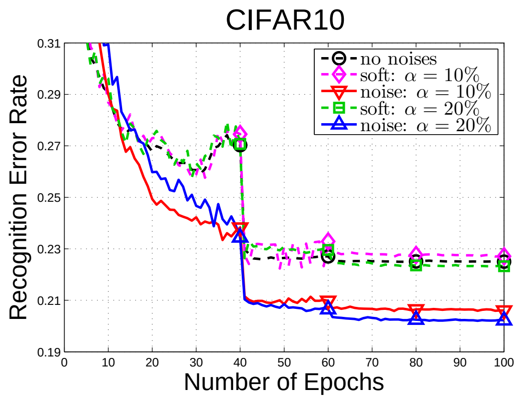

Figure 7. **CIFAR10** recognition error rate with soft labels and noisy labels (using the **LeNet**).

have already trained many different networks, combining them at the testing stage is very costly and often infeasible.

Each iteration in the DisturbLabel training process is like an iteration when the network is trained over a different noisy dataset $\widetilde{\mathcal{D}}$ where a mini-batch of samples $\widetilde{\mathcal{D}}_t$ are drawn. Thus, training a neural network with DisturbLabel can be regarded as training many networks with massive weight sharing but over different training data, where each training sample is used very rarely.

It is interesting that Dropout can be interpreted as a way of approximately combining exponentially many *different* neural network architectures trained on the *same* data efficiently, while DisturbLabel can be regarded as a way of approximately combining exponentially many neural networks with the *same* architecture but trained on *different* noisy data efficiently. In Section 5, we will show that DisturbLabel cooperates with Dropout to produce better results than individual models.

4.4. Interpretation as Data Augmentation

We analyze DisturbLabel from a data augmentation perspective. Considering a data point $(\mathbf{x},\mathbf{y})$ and its incorrect label $\widetilde{\mathbf{y}}$, the contribution to the loss is $|f(\mathbf{x})-\widetilde{\mathbf{y}}|_1$. This contribution can be rewritten as $|(f(\mathbf{x})-\widetilde{\mathbf{y}}+\mathbf{y})-\mathbf{y}|_1$, where $\widetilde{f}(\mathbf{x}) \doteq f(\mathbf{x})-\widetilde{\mathbf{y}}+\mathbf{y}$ can be viewed as a noisy output. Inspired by [12], we can project the noisy output $\widetilde{f}(\mathbf{x})$ back into the input space by minimizing the squared error $\left\|\widetilde{f}(\mathbf{x})-f(\widetilde{\mathbf{x}})\right\|_2^2$, where $\widetilde{\mathbf{x}}$ is the augmented sample. In summary, the data point with a disturbed label $(\mathbf{x},\widetilde{\mathbf{y}})$ can be transformed to an augmented data point $(\widetilde{\mathbf{x}},\mathbf{y})$.

To verify that DisturbLabel indeed acts as data augmentation, we evaluate the algorithm on the **MNIST** dataset [15] with only 1% (600) and 10% (6000) training samples, meanwhile keep the total number of iterations unchanged, *i.e.*, each training sample is used $100\times$ and $10\times$ times as it is used in the original training process. With the **LeNet** [16], we obtain 10.92% and 2.83% error rates on the original testing set, respectively, which are dramatic compared to 0.86% when the network is trained on the complete set. Meanwhile, in both cases, the training error rates quickly decrease to 0, implying that the limited training data cause over-fitting. DisturbLabel significantly decreases the error rates to 6.38% and 1.89%, respectively. As a reference, the error rate on the complete training set is further decreased to 0.66% by DisturbLabel. This indicates that DisturbLabel improves the quality of network training with implicit data augmentation, thus it serves as an effective algorithm especially in the case that the amount training data is limited.

4.5. Relationship to Other CNN Training Methods

We briefly discuss the relationship between DisturbLabel and other network training algorithms.

- **Other regularization methods.** There exist other network regularization methods, including DropConnect [36], Stochastic Pooling [37], Probabilistic Maxout [29], *etc*. Like Dropout [7] which regularizes CNNs on the hidden neurons, these methods add regularization on other places such as neuron connections and pooling operations. DisturbLabel regularizes CNNs on the loss layer, which, to the best of our knowledge, has never been studied before. As we will show in the next section, DisturbLabel cooperates well with Dropout to obtain superior results to individual modules. We believe that DisturbLabel can also provide complementary information to other network regularization methods.

- **Other methods dealing with noises.** Some previous works [8][31] aim at training CNNs with noisy labels. We emphasize that DisturbLabel is intrinsically dif-

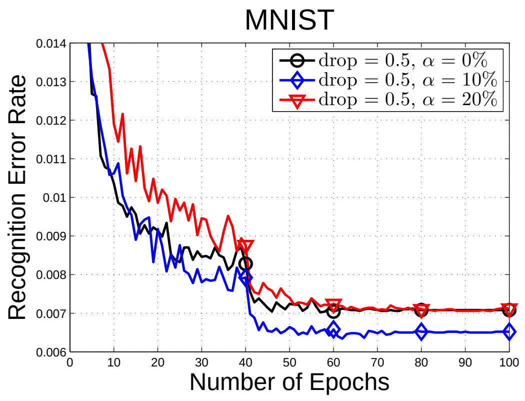

Figure 8. **MNIST** recognition error rate with drop rate 0.5 and different noise levels α (using the **LeNet**).

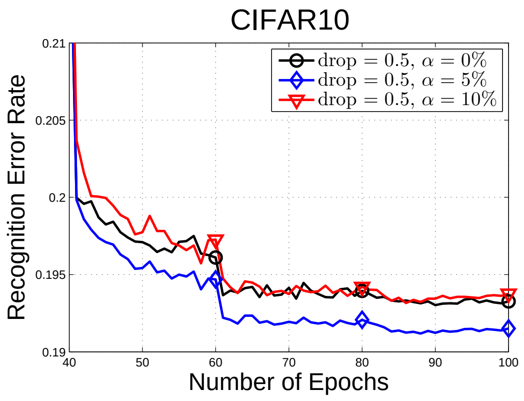

Figure 9. **CIFAR10** recognition error rate with drop rate 0.5 and different noise levels α (using the **LeNet**).

ferent with these methods since the problem settings are completely different. In these problems, training data suffer from noises (incorrect annotations), and researchers discuss the possibility of overcoming the noises towards an accurate training process. DisturbLabel, on the other hand, assumes that all the ground-truth labels are correct, and intentionally generates incorrect labels on a small fraction of data to prevent the network from over-fitting. In summary, inaccurate annotations may be harmful, but factitiously introducing noises is helpful to training a robust network.

- **Other network structures.** We are also interested in other sophisticated network structures, such as the Maxout Networks [4], the Deeply-Supervised Nets [18], the Network-in-Network [20] and the Recurrent Neural Networks [19]. Thanks to the generality, DisturbLabel can be adopted on these networks to improve their generalization ability.

5. Cooperation with Dropout

We have shown that DisturbLabel regularizes the CNN on the loss layer. This is different from Dropout [7], which regularizes the CNN on hidden layers. DisturbLabel is an approximate ensemble of many CNN models with the *same* structure trained over *different* noisy datasets, while Dropout is an approximate ensemble of many CNN models with *different* structures trained over the *same* data. These two regularization strategies are complementary. We empirically discuss the combination of DisturbLabel with Dropout, which leads to an ensemble of many CNN models with *different* structures trained over *different* noisy data.

We report the results with various noise levels α for DisturbLabel and a fixed drop rate for Dropout over the **MNIST** and **CIFAR10** datasets in Figures 8 and 9, respectively. In general, the proper combination of Dropout with DisturbLabel benefits the recognition accuracy improvement. In the MNIST dataset, the best result is obtained when $\alpha = 10\%$, meanwhile $\alpha = 20\%$ performs much worse. We note that in Figure 2, without Dropout, $\alpha = 10\%$ and $\alpha = 20\%$ produce comparable results. In the CIFAR10 dataset, $\alpha = 10\%$ does not help to improve the accuracy (close to baseline), while in Figure 3, we get comparable better results using $\alpha = 10\%$ and $\alpha = 20\%$. The above experiments show that both DisturbLabel and Dropout add regularization to network training. If both strategies are adopted, we need to reduce the regularization power properly to prevent "under-fitting".

In the later experiments, if both Dropout and DisturbLabel are used, we will reduce the noise level α by half to prevent the regularization on network training from being too strong. In the case that DisturbLabel provides strong regularization, *e.g.*, in the case of **ImageNet** training where the wrong label is distributed over all the 1000 categories, we will slightly decrease the drop rate of Dropout for the same purpose.

6. Experiments

We evaluate DisturbLabel on five popular datasets, *i.e.*, **MNIST** [15] and **SVHN** [23] for digit recognition, **CIFAR10/CIFAR100** [13] for natural image recognition, and **ImageNet** [3] for large-scale visual recognition.

6.1. The MNIST Dataset

MNIST [15] is one of the most popular datasets for handwritten digit recognition. This dataset consists of 60000 training images and 10000 testing images, uniformly distributed over 10 classes (0–9). All the samples are 28×28 grayscale images.

We use a modified version of the **LeNet** [16] as the baseline. The input image is passed through two units consisting of convolution, ReLU and max-pooling operations.

MNIST	w/o DA	with DA	**SVHN**	w/o DA	with DA
Jarrett [10]	0.53	–	Sermanet [27]	5.15	–
Zeiler [37]	0.47	–	Zeiler [37]	2.80	–
Lin [20]	0.47	–	Goodfellow [4]	2.47	–
Goodfellow [4]	0.45	–	Lin [20]	2.35	–
Lee [18]	0.39	–	Wan [36]	–	1.94
Liang [19]	0.31	–	Lee [18]	1.92	–
Wan [36]	0.52	0.21	Liang [19]	1.77	–
LeNet, no Reg.	0.86	0.48	**LeNet**, no Reg.	3.93	3.48
LeNet, Dropout	0.68	0.43	**LeNet**, Dropout	3.65	3.25
LeNet, DisturbLabel	0.66	0.45	**LeNet**, DisturbLabel	3.69	3.27
LeNet, both Reg.	0.63	0.41	**LeNet**, both Reg.	3.61	3.21
BigNet, no Reg.	0.69	0.39	**BigNet**, no Reg.	2.87	2.35
BigNet, Dropout	0.36	0.29	**BigNet**, Dropout	2.23	2.08
BigNet, DisturbLabel	0.38	0.32	**BigNet**, DisturbLabel	2.28	2.21
BigNet, both Reg.	0.33	0.28	**BigNet**, both Reg.	2.19	2.02

Table 2. Recognition error rates ($\%$) on the **MNIST** and **SVHN** datasets. DA: data augmentation (random cropping).

In which, the convolutional kernels are of the scale 5×5, the spatial stride 1, and max-pooling operators are of the scale 2×2 and the spatial stride 2. The number of convolutional kernels are 32 and 64, respectively. After the second max-pooling operation, a fully-connect layer with 512 filters is added, followed by ReLU and Dropout. The final layer is a 10-way classifier with the softmax loss function. We use a set of abbreviation to represent the above network configuration as: [C5(S1P0)@32-MP2(S2)]-[C5(S1P0)@64-MP2(S2)]-FC512-D0.5-FC10.

To obtain higher recognition accuracy, we also train a more complicated **BigNet**. The cross-map normalization [14] is adopted after each pooling layer, and the parameter K for normalization is proportional to the logarithm of the number of kernels. The network configuration is abbreviated as: [C5(S1P2)@128-MP3(S2)]-[C3(S1P1)@128-D0.7-C3(S1P1)@256-MP3(S2)]-D0.6- [C3(S1P1)@512]-D0.5-[C3(S1P1)@1024-MPS(S1)]-D0.4-FC10. Here, the number S is the map size before the final (global) max-pooling, before which the down-sampling rate is 4. Therefore, if the input image size is $W\times W$, $S=\lfloor W/4\rfloor$. The **BigNet** is feasible for data augmentation based on image cropping as the input image size is variable.

For data augmentation, we randomly crop input images into 24×24 pixels. We apply $(40,20,20,20)$ training epochs for the **LeNet**-based configurations with learning rates $\left(10^{-3},10^{-4},10^{-5},10^{-6}\right)$. For the **BigNet**-based configurations, the numbers are $(200,100,100,100)$ and $\left(10^{-2},10^{-3},10^{-4},10^{-5}\right)$, respectively.

We evaluate DisturbLabel with the noise level $\alpha=20\%$. According to the results shown in Table 2, DisturbLabel produces consistent accuracy gain over models without regularization, and also cooperates with Dropout to further improve the recognition performance. Train the **BigNet** using both Dropout and DisturbLabel achieves a 0.33% error rate without data augmentation, which outperforms several recently reported results [4][18]. In comparison with [2] which applies more complicated data augmentation (*e.g.*, image rotation), we only use randomly image cropping and obtain a comparable error rate (0.28% vs. 0.23%).

6.2. The SVHN Dataset

The **SVHN** dataset [23] is a larger collection of 32×32 RGB images, *i.e.*, 73257 training samples, 26032 testing samples, and 531131 extra training samples. We preprocess the data as in the previous methods [23], *i.e.*, selecting 400 samples per category from the training set as well as 200 samples per category from the extra set, using these 6000 images for validation, and the remaining 598388 images as training samples. We also use Local Contrast Normalization (LCN) for data preprocessing [4].

We use another version of the **LeNet**. A $32\times32\times3$ image is passed through three units consisting of convolution, ReLU and max-pooling operations. Using abbreviation, the network configuration can be written as: [C5(S1P2)@32-MP3(S2)]-[C5(S1P2)@32-MP3(S2)]-[C5(S1P2)@64-MP3(S2)]-FC64-D0.5-FC10. Padding of 2 pixels wide is added in each convolution operation to preserve the width and height of the data. The **BigNet** is also used to achieve higher accuracy. The training epochs, learning rates and data augmentation settings remain the same as in the **MNIST** experiments.

We evaluate DisturbLabel with the noise level $\alpha=20\%$, and summarize the results in Table 2. We can observe that DisturbLabel improves the recognition accuracy, either with or without using Dropout. With data augmentation and both regularization methods, we achieve a competitive 2.02% error rate.

CIFAR10	w/o DA	with DA	**CIFAR100**	w/o DA	with DA
Zeiler [37]	15.13	–	Zeiler [37]	42.51	–
Goodfellow [4]	11.68	9.38	Goodfellow [4]	38.57	–
Lin [20]	10.41	8.81	Srivastava [30]	36.85	–
Wan [36]	–	9.32	Lin [20]	35.68	–
Lee [18]	9.69	7.97	Lee [18]	34.57	–
Liang [19]	8.69	7.09	Liang [19]	31.75	–
LeNet, no Reg.	22.50	15.76	**LeNet**, no Reg.	56.72	43.31
LeNet, Dropout	19.42	14.24	**LeNet**, Dropout	49.08	41.28
LeNet, DisturbLabel	20.26	14.48	**LeNet**, DisturbLabel	51.83	41.84
LeNet, both Reg.	19.18	13.98	**LeNet**, both Reg.	48.72	40.98
BigNet, no Reg.	11.23	9.29	**BigNet**, no Reg.	39.54	33.59
BigNet, Dropout	9.69	7.08	**BigNet**, Dropout	33.30	27.05
BigNet, DisturbLabel	9.82	7.93	**BigNet**, DisturbLabel	34.81	28.39
BigNet, both Reg.	9.45	6.98	**BigNet**, both Reg.	32.99	26.63

Table 3. Recognition error rates (%) on the **CIFAR10** and **CIFAR100** datasets. DA: data augmentation (random cropping and flipping).

6.3. The CIFAR Datasets

The **CIFAR10** and **CIFAR100** datasets [13] are both subsets drawn from the 80-million tiny image database [33]. There are 50000 images for training, and 10000 images for testing, all of them are 32×32 RGB images. **CIFAR10** contains 10 basic categories, and **CIFAR100** divides each of them into a finer level. In both datasets, training and testing images are uniformly distributed over all the categories. We use exactly the same network configuration as in the **SVHN** experiments, and add left-right image flipping into data augmentation with the probability 50%.

We evaluate DisturbLabel with the noise level $\alpha = 10\%$. In **CIFAR-100**, we slightly modify DisturbLabel by only allowing disturbing the label among 10 finer-level categories. We compare our results with the state-of-the-arts in Table 3. Once again, DisturbLabel produces consistent accuracy gain in every single case, either with or without Dropout. On **CIFAR10**, the **BigNet** with Dropout produces an excellent baseline (7.08% error rate), and DisturbLabel further improves the performance (6.98% error rate) with a complementary regularization function to Dropout. The success on the **CIFAR** datasets verifies that DisturbLabel is generalized: it not only works well in relatively simple digit recognition, but also helps natural image recognition tasks.

6.4. The ImageNet Database

Finally we evaluate DisturbLabel on the **ILSVRC2012** classification task [26], a subset of the **ImageNet** database [3] which contains 1000 object categories. The training set, validation set and testing set contain 1.3M, 50K and 150K images, respectively. We use the **AlexNet** [14] (provided by the **CAFFE** library [11]) with the dropout rate 0.5 as the baseline. The **AlexNet** structure is abbreviated as: C11(S4)@96-MP3(S2)-LRN-C5(S1P2)@256-MP3(S2)-LRN- C3(S1P1)@384-C3(S1P1)@384-C3(S1P1)@256-MP3(S2)-FC4096-D0.5-FC4096-D0.5-FC1000.

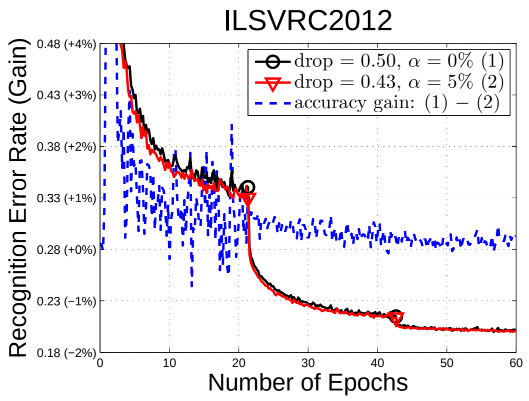

Figure 10. The error rate and accuracy gain curves on the **ILSVRC2012** validation set (the **AlexNet** [14] is used).

We note that when a training sample is chosen to be disturbed, the label will be uniformly distributed among all the 1000 classes, introducing strong noises to the training process, even when the noise level is relatively low ($\alpha = 5\%$ is used). Therefore, we decrease the dropout rate to 0.43 (less data are dropped) to perform weaker regularization. Figure 10 shows the error rate curve on the validation set. After about 20 epochs, DisturbLabel consistently improves the baseline accuracy by about 0.2%. We emphasize that the 0.2% gain is not so small as it seems (*e.g.*, the **VGGNet** [28] combines two individually trained nets to get a 0.1% gain), which once again verifies that DisturbLabel and Dropout cooperate well to provide regularization in different aspects.

While we only evaluate DisturbLabel on the **AlexNet**, we believe that it can also cooperate with other network architectures, such as the **GoogLeNet** [32] and the **VGGNet** [28], since regularization is a common requirement of deep neural networks.

7. Conclusions

In this paper, we present **DisturbLabel**, a novel algorithm which regularizes CNNs on the **loss layer**. DisturbLabel is surprisingly simple, which works by randomly choosing a small subset of training data, and intentionally setting their ground-truth labels to be incorrect. We show that DisturbLabel consistently improves the network training process by preventing it from over-fitting, and that DisturbLabel can be explained as an alternative solution of implicit model ensemble and data augmentation. Meanwhile, DisturbLabel cooperates well with Dropout, which regularizes CNNs on the hidden neurons. Experiments verify that DisturbLabel achieves competitive performance on several popular image classification benchmarks.

In the future, we will explore the use of DisturbLabel among more CNN-based applications, including object detection, image segmentation, textual analysis, *etc.*.

Acknowledgements

We would like to thank Prof. Alan Yuille for instructive discussions.

References

[1] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. *International Conference on Computational Statistics*, 2010.

[2] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*, 22(12):3207–3220, 2010.

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-scale Hierarchical Image Database. *Computer Vision and Pattern Recognition*, 2009.

[4] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout Networks. *International Conference on Machine Learning*, 2013.

[5] B. Graham. Fractional Max-Pooling. *arXiv preprint, arXiv: 1412.6071*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *International Conference on Computer Vision*, 2015.

[7] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv preprint, arXiv: 1207.0580*, 2012.

[8] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint, arXiv: 1503.0253*, 2014.

[9] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, 2015.

[10] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? *International Conference on Computer Vision*, 2009.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. CAFFE: Convolutional Architecture for Fast Feature Embedding. *ACM International Conference on Multimedia*, 2014.

[12] K. Konda, X. Bouthillier, R. Memisevic, and P. Vincent. Dropout as Data Augmentation. *arXiv preprint, arXiv: 1506.08700*, 2015.

[13] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*, 1(4):7, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 2012.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Y. LeCun, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 1990.

[17] C. Lee, P. Gallagher, and Z. Tu. Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. *arXiv preprint, arXiv: 1509.08985*, 2015.

[18] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. *International Conference on Artificial Intelligence and Statistics*, 2015.

[19] M. Liang and X. Hu. Recurrent Convolutional Neural Network for Object Recognition. *Computer Vision and Pattern Recognition*, 2015.

[20] M. Lin, Q. Chen, and S. Yan. Network in Network. *International Conference on Learning Representations*, 2014.

[21] L. Maaten, M. Chen, S. Tyree, and K. Weinberger. Learning with Marginalized Corrupted Features. *International Conference on Machine Learning*, 2013.

[22] L. Maaten, M. Chen, S. Tyree, and K. Weinberger. Marginalizing Corrupted Features. *arXiv preprint, arXiv: 1402.7001*, 2014.

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[24] D. Plaut et al. Experiments on Learning by Back Propagation. *Technical Report: CMU-CS-86-126, CMU*, 1986.

[25] R. Reed, S. Oh, R. Marks, et al. Regularization Using Jittered Training Data. *International Joint Conference on Neural Networks*, 3:147–152, 1992.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, pages 1–42, 2015.

[27] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional Neural Networks Applied to House Numbers Digit Classification. *International Conference on Pattern Recognition*, 2012.

[28] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2014.

[29] J. Springenberg and M. Riedmiller. Improving Deep Neural Networks with Probabilistic Maxout Units. *International Conference on Learning Representations*, 2013.

[30] N. Srivastava and R. Salakhutdinov. Discriminative Transfer Learning with Tree-based Priors. *Advances in Neural Information Processing Systems*, 2013.

[31] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training Convolutional Networks with Noisy Labels. *arXiv preprint, arXiv: 1406.2080*, 2014.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *Computer Vision and Pattern Recognition*, 2015.

[33] A. Torralba, R. Fergus, and W. Freeman. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. *International Conference on Machine Learning*, 2008.

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[36] L. Wan, M. Zeiler, S. Zhang, Y. Cun, and R. Fergus. Regularization of Neural Networks using DropConnect. *International Conference on Machine Learning*, 2013.

[37] M. Zeiler and R. Fergus. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *arXiv preprint, arXiv: 1301.3557*, 2013.