DisturbLabel: Regularizing CNN on the Loss Layer

Lingxi Xie[1], Jingdong Wang[2], Zhen Wei[3], Meng Wang[4] and Qi Tian[5]

[1]University of California, Los Angeles

[2]Microsoft Research [3]Shanghai Jiao Tong University

[4]Hefei University of Technology [5]University of Texas at San Antonio

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.

CONTRIBUTION

In this paper, we present a novel algorithm named **DisturbLabel**, which regularizes CNN by intentionally introducing incorrect labels in the training process. In each training iteration, each training sample is randomly picked up (with probability α), then assigned a random label. To the best of our knowledge, this is the first work to add noise on the *loss layer*.

DisturbLabel prevents over-fitting in the CNN training process. It can be explained as two different ways, *i.e.*, model ensemble and data augmentation, both of which are performed in a latent manner.

- Like Dropout, another popular regularization algorithm, DisturbLabel can be explained as a latent way of averaging a large number of models. In Dropout, models are trained with the *same* data and *different* network structures; but in DisturbLabel, models are trained with the *same* network structure and *different* data. Therefore, DisturbLabel can be used with Dropout.
- DisturbLabel can also be explained as data augmentation (see the figure to the right). It is equivalent to generating many *difficult* training samples that increase the ability of the network. Therefore, DisturbLabel can be used in the scenarios with fewer training data or with imbalanced training data, since it shares data among different categories.

The implementation of DisturbLabel is very easy, with only few lines of codes. Experimental results on several popular image classification benchmarks verify that DisturbLabel produces competitive recognition results.

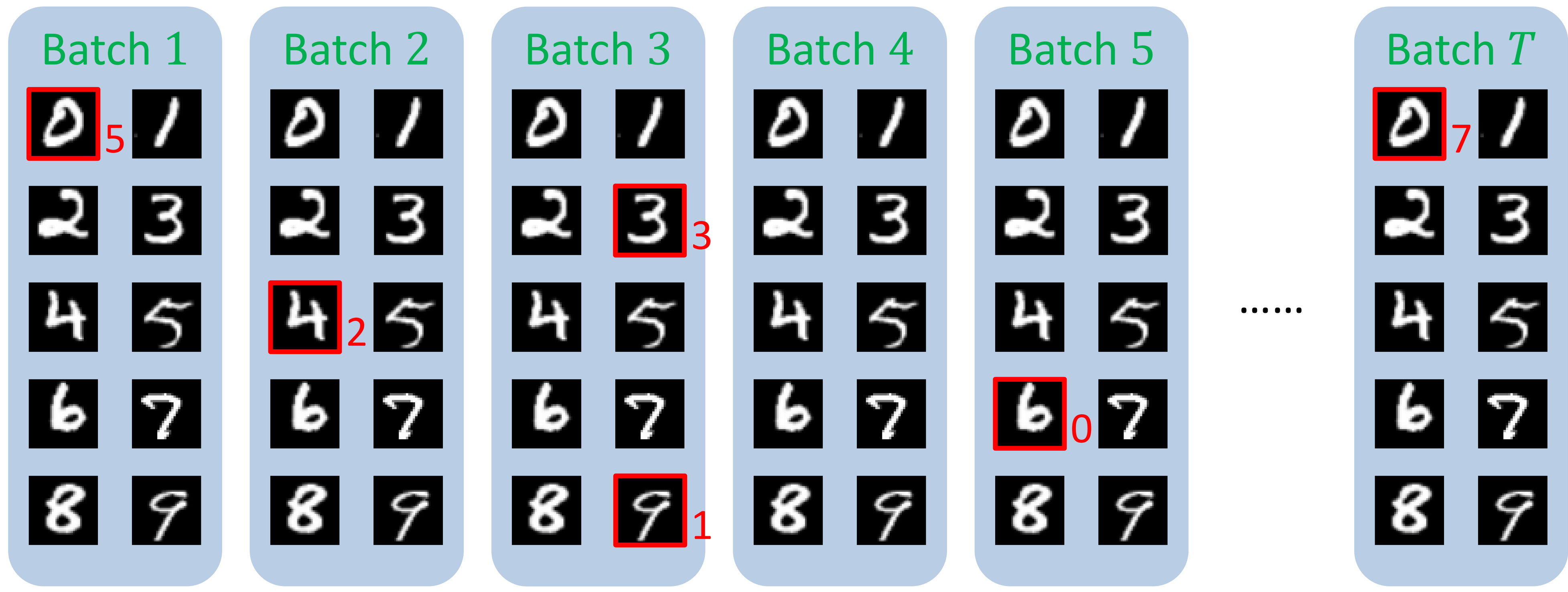
THE PROPOSED ALGORITHM

Pseudo codes for **DisturbLabel**

Input: a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, noise rate α.
Initialization: a network $\mathbb{M}$: $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_0) \in \mathbb{R}^C$;
for each mini-batch $\mathcal{D}_t = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M}$ **do**
  for each training sample $(\mathbf{x}_m, \mathbf{y}_m)$ **do**
    Generate a disturbed label $\tilde{\mathbf{y}}_m$;
  end for
  Update the parameter $\boldsymbol{\theta}_t$ with SGD;
end for
Output: the trained model $\mathbb{M}$: $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_T) \in \mathbb{R}^C$.

Implementation details of label disturbation

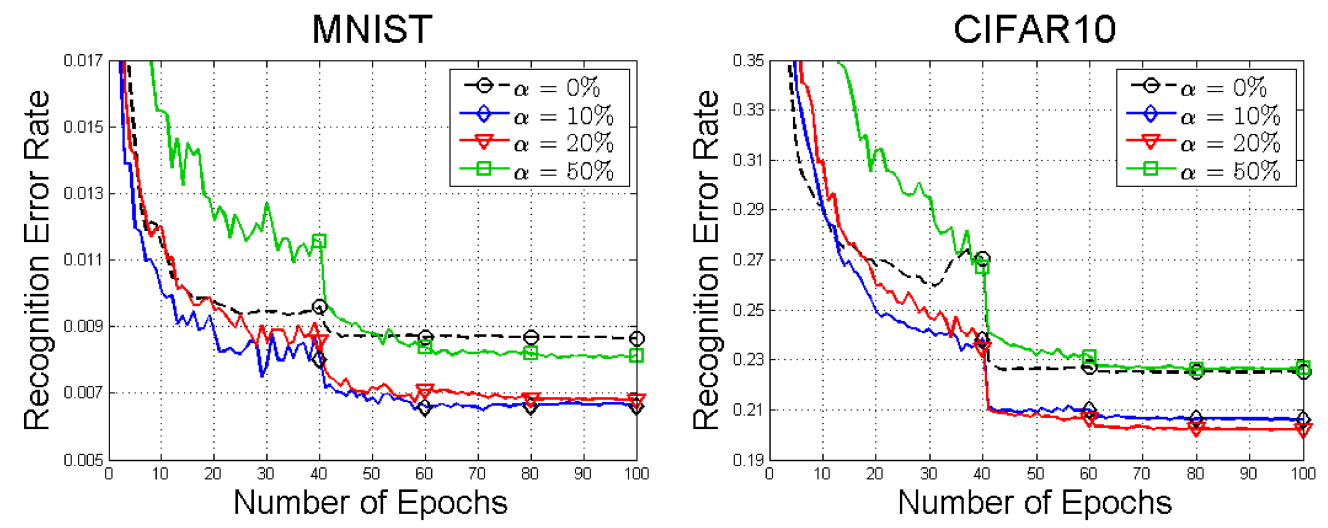
Each data sample $(\mathbf{x}, \mathbf{y})$ is sent into an extra sampling process, in which a disturbed label vector $\tilde{\mathbf{y}} = [\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y}_C]^{\mathrm{T}}$ is randomly generated from a Multinoulli (generalized Bernoulli) distribution $\mathcal{P}(\alpha)$. Suppose that the sampled integer is $\tilde{c}$, then we have $\tilde{y}_{\tilde{c}} = 1$ and $\tilde{y}_i = 1$ for all $i \neq \tilde{c}$.



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).

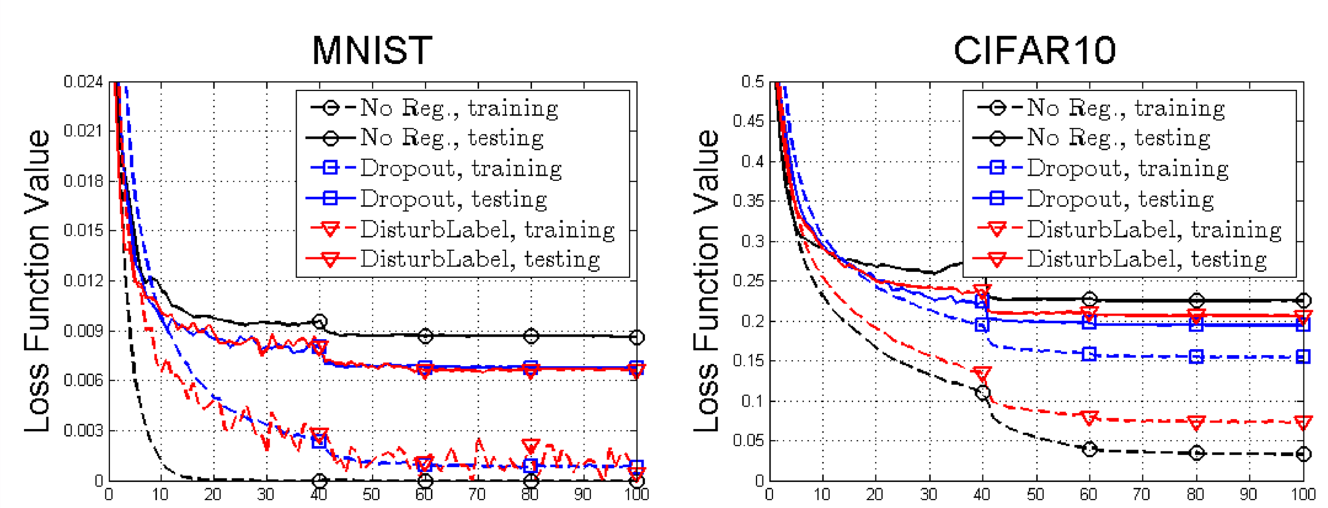
DisturbLabel produces higher accuracy

- Like in **Dropout**, adding proper noise in **DisturbLabel** helps network training, but introducing too much noise harms.
- It is generally safe to add relatively small noise (e.g., $\alpha = 10\%$).



DisturbLabel prevents over-fitting

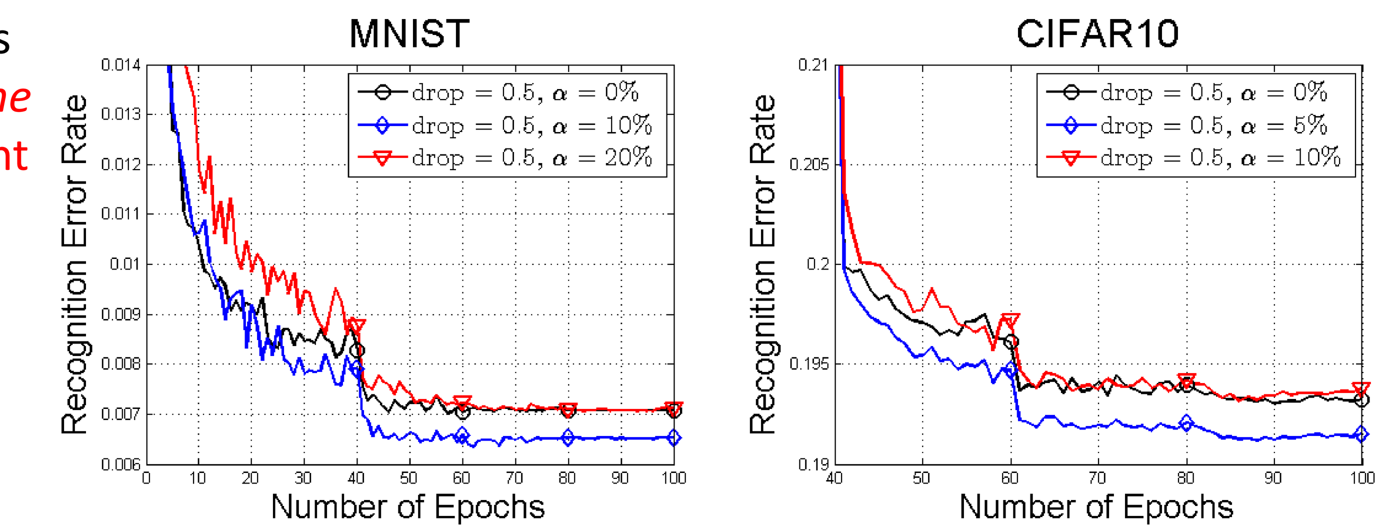
- Both **Dropout** and **DisturbLabel** slow down the network training process, but lead to better generalization (smaller testing error). Therefore, we can conclude that **DisturbLabel** prevents over-fitting, like **Dropout** does.
- The reason lies in model ensemble and data augmentation.



DisturbLabel can be considered as a *latent* way of *model ensemble*

Dropout: models trained with *same* data and *different* structures

DisturbLabel: models trained with *different* data and *same* structure



DisturbLabel can be considered as a *latent* way of *data augmentation*

Given a disturbed data $(\mathbf{x}_n, \tilde{\mathbf{y}}_n)$, its loss function value is $L(\mathbf{x}_n, \tilde{\mathbf{y}}_n)$. We can generate an augmented data $(\tilde{\mathbf{x}}_n, \mathbf{y}_n)$, so that $L(\tilde{\mathbf{x}}_n, \mathbf{y}_n) \approx L(\mathbf{x}_n, \tilde{\mathbf{y}}_n)$, to stylize the effect of $(\mathbf{x}_n, \tilde{\mathbf{y}}_n)$ with $(\tilde{\mathbf{x}}_n, \mathbf{y}_n)$.

- $\tilde{\mathbf{x}}_n$ is obtained by iterative backprop;
- See the right figure for some examples.

To further illustrate that **DisturbLabel** serves as latent data augmentation, we perform two experiments:

- Training with few data: only 1% or 10% data are preserved (see lowerleft);
- Training with imbalanced data: except for the first class, only only 1% or 10% data are preserved (see lowerright).

	MNIST	CIFAR10
1% data, w/o DisturbLabel	10.92	43.29
1% data, w/ DisturbLabel	6.38	37.83
10% data, w/o DisturbLabel	2.83	27.21
10% data, w/ DisturbLabel	1.89	24.37
100% data, w/o DisturbLabel	0.86	22.50
100% data, w/ DisturbLabel	0.66	20.26

	MNIST		CIFAR10	
	overall	*first* class	overall	*first* class
1% data, w/o DisturbLabel	9.31	0.28	42.01	11.48
1% data, w/ DisturbLabel	6.29	2.35	36.92	24.30
10% data, w/o DisturbLabel	2.78	0.47	26.50	13.09
10% data, w/ DisturbLabel	1.76	1.46	24.03	18.19
100% data, w/o DisturbLabel	0.86	0.89	22.50	22.41
100% data, w/ DisturbLabel	0.66	0.71	20.26	20.29

RESULTS

Results on some *small* datasets

	MNIST		SVHN	
	-DA	+DA	-DA	+DA
Zeiler [42]	0.45	–	2.80	–
Goodfellow [5]	0.47	–	2.47	–
Lin [21]	0.47	–	2.35	–
Wan [38]	0.52	0.21	–	1.94
Lee [21]	0.39	–	1.92	–
Liang [20]	0.31	–	1.77	–
LeNet, no regularization	0.86	0.48	3.93	3.48
LeNet, + Dropout	0.68	0.43	3.65	3.25
LeNet, + DisturbLabel	0.66	0.45	3.69	3.27
LeNet, + both	0.63	0.41	3.61	3.21
BigNet, no regularization	0.69	0.39	2.87	2.35
BigNet, + Dropout	0.36	0.29	2.23	2.08
BigNet, + DisturbLabel	0.38	0.32	2.28	2.21
BigNet, + both	0.33	0.28	2.19	2.02

	CIFAR10		CIFAR100	
	-DA	+DA	-DA	+DA
Zeiler [42]	15.13	–	42.51	–
Goodfellow [5]	11.68	9.38	38.57	–
Lin [21]	10.41	8.81	35.68	–
Wan [38]	–	9.32	–	–
Lee [21]	9.69	7.97	34.57	–
Liang [20]	8.69	7.09	31.75	–
LeNet, no regularization	22.50	15.76	56.72	43.31
LeNet, + Dropout	19.42	14.24	49.08	41.28
LeNet, + DisturbLabel	20.26	14.48	51.83	41.84
LeNet, + both	19.18	13.98	48.72	40.98
BigNet, no regularization	11.23	9.29	39.54	33.59
BigNet, + Dropout	9.69	7.08	33.30	27.05
BigNet, + DisturbLabel	9.82	7.93	34.81	28.39
BigNet, + both	9.45	6.98	32.99	26.63

Please refer to our paper for **ImageNet** results

REFERENCES

Key references are numbered as they appear in the paper.

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

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

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

ACKNOWLEDGEMENTS

This work was done when Lingxi Xie and Zhen Wei were interns at MSR. This work was supported by ARO grant W911NF-15-1-0290, Faculty Research Gift Awards by NEC Labs of America and Blippar, and NSFC 61322201, 61429201 and 61432019. We thank Prof. Alan Yuille and the anonymous reviewers for instructive discussions.