Robust Learning from Noisy Labels

Dr. Quanming Yao
Leading Researcher@4Paradigm Inc.
http://www.cse.ust.hk/~qyaoaa/

Joint work with Gang Niu, Bo Han, Masashi Sugiyama (from RIKEN), Weihua Hu (from Stanford) and Ivor Tsang (from UTS)
Agenda

• Application Background

• Related Works

• Co-teaching: Learning from noisy labels

• Experiments

• Summary
Recent success of deep models

**ImageNet**
- 14197122 images
- 21841 classes indexed

- Sparse coding
  - Top 1 err: 47.1%
  - Top 5 err: 28.2%

- Conv-net
  - Top 1 err: 37.5%
  - Top 5 err: 17.0%

- ResNet
  - Top 1 err: 37.5%
  - Top 5 err: 17.0%

*Big & High-quality data is the fuel*
Where does big data come from?

Crowd-sourcing

- An example worker on AMT

- Incorrect ones

- Correct ones

Web crawler

- take image as sample

- take words from caption as labels
Where does big data come from?

Crowd-sourcing
- Workers may not be reliable
- There can be spammers or attackers

Web crawler
- The context can be complex
- Caption may not be relevant

Big & High-quality data: difficult & expensive

- Data: what we usually have in hand is a big data with noisy labels
- Performance: noisy labels degrade the accuracy of deep neural networks by 20% to 40%
What is special about noisy labels?

If the classifier A has the ability to predict, then a sample with noisy labels should have larger loss than a sample with correct labels.

Using hinge loss as an example:
- Red points: zero loss
- Blue points: much larger than zero
What is special about deep networks?

Stochastic gradient descent (SGD) is a **must** for training deep networks.

*Image classification*

*Train/test accuracy v.s. steps*

[Zhang, et al. 2016]
What is special about deep networks?

Noisy labels

Standard CNN

Learning easy patterns first, then (totally) over-fit noisy training data.

Independent with network types and structures.
What is special about deep networks?

Fundamental properties

- SGD is a must for deep networks
- Deep networks have memorization effects

Facts

- Noisy labels has larger losses.

How can we robustly learn from noisy label utilizing above properties and fact?
Agenda

• Application Background

• Related Works
  • MentorNet and Decoupling
  • Co-teaching: Learning from noisy labels

• Experiments

• Summary
Deep networks are all based on
- Stochastic gradient descent
- Gradient is performed by mini-batch

Mentor-Net
- drop samples with large loss in each mini-batch, use small loss samples in each mini-batch to update parameters
- use one classifier to self-bootstrap
State-of-the-arts: Decoupling [E. Malach and S. Shalev-Shwartz, 2017]

Easy samples
- Can be quickly learnt and classified (memorization)
- Have small gradients, which slow down network training

Decoupling
- Focus on hard examples, which can be more informative
- Use samples in each mini-batch that two classifiers have different predictions to update network
State-of-the-arts

How can we robustly learn from noisy label utilizing (small loss, memorization and SGD)?

<table>
<thead>
<tr>
<th></th>
<th>Mentor-net</th>
<th>Decoupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small loss</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Memorization</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>SGD</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
Agenda

- Application Background
- Related Works
- Co-teaching: Learning from noisy labels
- Experiments
- Summary
Co-teaching: Core Idea

Why not exchange small loss in each mini-batch for two classifiers?
Co-teaching: Core Idea

Algorithm 1 Co-teaching Paradigm.

1: Input $w_f$ and $w_g$, learning rate $\eta$, fixed $\tau$, epoch $T_k$ and $T_{max}$, iteration $N_{max}$;
2: Shuffle training set $\mathcal{D}$; //noisy dataset
3: Draw mini-batch $\mathcal{D}_f$ from $\mathcal{D}$;
4: Sample $\mathcal{D}_f = \arg\min_{\mathcal{D}} \ell(f, \mathcal{D}, R(T))$; //sample $R(T)$% small-loss instances
5: Sample $\mathcal{D}_g = \arg\min_{\mathcal{D}} \ell(g, \mathcal{D}, R(T))$; //sample $R(T)$% small-loss instances
6: Update $w_f = w_f - \eta \nabla f(\mathcal{D}_g)$; //update $w_f$ by $\mathcal{D}_g$;
7: Update $w_g = w_g - \eta \nabla g(\mathcal{D}_f)$; //update $w_g$ by $\mathcal{D}_f$;
8: Update $R(T) = 1 - \min\left\{ \frac{T}{T_k} \tau, \tau \right\}$;
9: Output $w_f$ and $w_g$

- Change the procedures in SGD algorithm

exchange small loss samples
Co-teaching: Key questions

Q1. Why can sampling small-loss instances help find clean instances?

• When labels are correct, small-loss instances are more likely to be ones with correct labels.

• However, the above requires that the classifier is reliable enough. The “memorization” effect of deep networks can exactly help us address this problem.
Co-teaching: Core Idea

Algorithm 1 Co-teaching Paradigm.
1: Input $w_f$ and $w_g$, learning rate $\eta$, fixed $\tau$, epoch $T_k$ and $T_{\text{max}}$, iteration $N_{\text{max}}$;
for $T = 1, 2, \ldots, T_{\text{max}}$ do
  2: Shuffle training set $\mathcal{D}$; //noisy dataset
  for $N = 1, \ldots, N_{\text{max}}$ do
    3: Draw mini-batch $\mathcal{D}$ from $\mathcal{D}$;
    4: Sample $\mathcal{D}_f = \arg\min_{\mathcal{D}} \ell(f, D, R(T))$; //sample $R(T)\%$ small-loss instances
    5: Sample $\mathcal{D}_g = \arg\min_{\mathcal{D}} \ell(g, D, R(T))$; //sample $R(T)\%$ small-loss instances
    6: Update $w_f = w_f - \eta \nabla f(\mathcal{D}_f)$; //update $w_f$ by $\mathcal{D}_f$
    7: Update $w_g = w_g - \eta \nabla g(\mathcal{D}_g)$; //update $w_g$ by $\mathcal{D}_g$
  end
8: Update $R(T) = 1 - \min\left\{\frac{T}{T_k}, \tau, \tau\right\}$; How many samples to be kept
end
9: Output $w_f$ and $w_g$

- $R(T)$: adapts from memorization effect
- At start, network can learn, thus we keep more samples; then, when start overfitting, we drop more samples
Co-teaching: Core Idea

Algorithm 1 Co-teaching Paradigm.

1: **Input** $w_f$ and $w_g$, learning rate $\eta$, fixed $\tau$, epoch $T_k$ and $T_{\text{max}}$, iteration $N_{\text{max}}$;

for $T = 1, 2, \ldots, T_{\text{max}}$ do

2: **Shuffle** training set $\mathcal{D}$;

for $N = 1, \ldots, N_{\text{max}}$ do

3: **Draw** mini-batch $\tilde{\mathcal{D}}$ from $\mathcal{D}$;

4: **Sample** $\mathcal{D}_f = \arg \min_{\mathcal{D}} \ell(f, \mathcal{D}, R(T))$; //sample $R(T)\%$ small-loss instances

5: **Sample** $\mathcal{D}_g = \arg \min_{\mathcal{D}} \ell(g, \mathcal{D}, R(T))$; //sample $R(T)\%$ small-loss instances

6: **Update** $w_f = w_f - \eta \nabla f(\mathcal{D}_g)$; //update $w_f$ by $\mathcal{D}_g$;

7: **Update** $w_g = w_g - \eta \nabla g(\mathcal{D}_f)$; //update $w_g$ by $\mathcal{D}_f$;

8: **Update** $R(T) = 1 - \min \left\{ \frac{T}{T_k}, \tau \right\}$;

end

end

9: **Output** $w_f$ and $w_g$

- $\tau$: noisy level (assumption to be known), i.e., how many labels are not correct in the training data
Co-teaching: Key questions

Q2. Why do we need two networks and cross-update the parameters?

• Different classifiers generate different decision boundaries and then have different abilities to learn.

• We expect that they can have different abilities to filter out the label noise.
Agenda

- Application Background
- Related Works
- Co-teaching: Learning from noisy labels
- Experiments
- Summary
Experiments: setup

- Transition matrices of different noise types (using 5 classes as an example)
- *Pair* is much harder than *symmetry*
Experiments: setup

<table>
<thead>
<tr>
<th>CNN on MNIST</th>
<th>CNN on CIFAR-10</th>
<th>CNN on CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>28×28 Gray Image</td>
<td>32×32 RGB Image</td>
<td>32×32 RGB Image</td>
</tr>
<tr>
<td>3×3 conv, 128 LReLU</td>
<td>3×3 conv, 128 LReLU</td>
<td>3×3 conv, 128 LReLU</td>
</tr>
<tr>
<td>3×3 conv, 128 LReLU</td>
<td>3×3 conv, 128 LReLU</td>
<td>3×3 conv, 128 LReLU</td>
</tr>
<tr>
<td>2×2 max-pool, stride 2, dropout, $p = 0.25$</td>
<td>2×2 max-pool, stride 2, dropout, $p = 0.25$</td>
<td>2×2 max-pool, stride 2, dropout, $p = 0.25$</td>
</tr>
<tr>
<td>3×3 conv, 256 LReLU</td>
<td>3×3 conv, 256 LReLU</td>
<td>3×3 conv, 256 LReLU</td>
</tr>
<tr>
<td>3×3 conv, 256 LReLU</td>
<td>3×3 conv, 256 LReLU</td>
<td>3×3 conv, 256 LReLU</td>
</tr>
<tr>
<td>avg-pool</td>
<td>dense 128→10</td>
<td>dense 128→10</td>
</tr>
</tbody>
</table>

- CNN models used on MNIST, CIFAR-10, and CIFAR-100. The slopes of all LReLU functions in the networks are set to 0.01.
- These are not state-of-the-art models, but testbed for noisy labels [S. Laine and T. Aila, 2017].
Experiments: setup

Further correct loss

<table>
<thead>
<tr>
<th></th>
<th>Bootstrap</th>
<th>S-model</th>
<th>F-correction</th>
<th>Decoupling</th>
<th>MentorNet</th>
<th>Co-teaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>large class</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>heavy noise</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>flexibility</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>no pre-train</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Experiments: MNIST

Average test accuracy on MNIST over the last ten epochs

<table>
<thead>
<tr>
<th>Flipping-Rate</th>
<th>Normal</th>
<th>Bootstrap</th>
<th>S-model</th>
<th>F-correction</th>
<th>Decoupling</th>
<th>MentorNet</th>
<th>Co-teaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair-45%</td>
<td>56.52%</td>
<td>57.23%</td>
<td>56.88%</td>
<td>0.24%</td>
<td>58.03%</td>
<td>80.88%</td>
<td>87.63%</td>
</tr>
<tr>
<td></td>
<td>±0.55%</td>
<td>±0.73%</td>
<td>±0.32%</td>
<td>±0.03%</td>
<td>±0.07%</td>
<td>±4.45%</td>
<td>±0.21%</td>
</tr>
<tr>
<td>Symmetry-50%</td>
<td>66.05%</td>
<td>67.55%</td>
<td>62.29%</td>
<td>79.61%</td>
<td>81.15%</td>
<td>90.05%</td>
<td>91.32%</td>
</tr>
<tr>
<td></td>
<td>±0.61%</td>
<td>±0.53%</td>
<td>±0.46%</td>
<td>±1.96%</td>
<td>±0.03%</td>
<td>±0.30%</td>
<td>±0.06%</td>
</tr>
<tr>
<td>Symmetry-20%</td>
<td>94.05%</td>
<td>94.40%</td>
<td>98.31%</td>
<td><strong>98.80%</strong></td>
<td>95.70%</td>
<td>96.70%</td>
<td>97.25%</td>
</tr>
<tr>
<td></td>
<td>±0.16%</td>
<td>±0.26%</td>
<td>±0.11%</td>
<td>±0.12%</td>
<td>±0.02%</td>
<td>±0.22%</td>
<td>±0.03%</td>
</tr>
</tbody>
</table>
Experiments: MNIST

Test accuracy vs number of epochs on MNIST dataset
Experiments: MNIST

Label precision vs number of epochs on MNIST dataset.
Experiments: CIFAR10

Average test accuracy on CIFAR10 over the last ten epochs

<table>
<thead>
<tr>
<th>Flipping Rate</th>
<th>Standard</th>
<th>Bootstrap</th>
<th>S-model</th>
<th>F-correction</th>
<th>Decoupling</th>
<th>MentorNet</th>
<th>Co-teaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair-45%</td>
<td>49.50%</td>
<td>50.05%</td>
<td>48.21%</td>
<td>6.61%</td>
<td>48.80%</td>
<td>58.14%</td>
<td><strong>72.62%</strong></td>
</tr>
<tr>
<td></td>
<td>±0.42%</td>
<td>±0.30%</td>
<td>±0.55%</td>
<td>±1.12%</td>
<td>±0.04%</td>
<td>±0.38%</td>
<td>±0.15%</td>
</tr>
<tr>
<td>Symmetry-50%</td>
<td>48.87%</td>
<td>50.66%</td>
<td>46.15%</td>
<td>59.83%</td>
<td>51.49%</td>
<td>71.10%</td>
<td><strong>74.02%</strong></td>
</tr>
<tr>
<td></td>
<td>±0.52%</td>
<td>±0.56%</td>
<td>±0.76%</td>
<td>±0.17%</td>
<td>±0.08%</td>
<td>±0.48%</td>
<td>±0.04%</td>
</tr>
<tr>
<td>Symmetry-20%</td>
<td>76.25%</td>
<td>77.01%</td>
<td>76.84%</td>
<td><strong>84.55%</strong></td>
<td>80.44%</td>
<td>80.76%</td>
<td>82.32%</td>
</tr>
<tr>
<td></td>
<td>±0.28%</td>
<td>±0.29%</td>
<td>±0.66%</td>
<td>±0.16%</td>
<td>±0.05%</td>
<td>±0.36%</td>
<td>±0.07%</td>
</tr>
</tbody>
</table>
Experiments: CIFAR10

Test accuracy vs number of epochs on CIFAR10 dataset
Experiments: CIFAR10

Label precision vs number of epochs on CIFAR10 dataset.
Experiments: $R(T)$ and $\tau$

Impact of memorization

$$R(T) = 1 - \min \left\{ \frac{T^c}{T_k}, \tau, \tau \right\}$$

Choices

- $c \in \{0.5, 1.0, 2\}$
- $T_k \in \{5, 10, 15\}$

Algorithm 1: Co-teaching Paradigm.

1: Input $w_f$ and $w_g$, learning rate $\eta$, fixed $\tau$, epoch $T$
2: for $T = 1, 2, \ldots, T_{\text{max}}$ do
3: Shuffle training set $\mathcal{D}$;
4: for $N = 1, \ldots, N_{\text{max}}$ do
5: Draw mini-batch $\mathcal{D}$ from $\mathcal{D}$;
6: Sample $\mathcal{D}_f = \arg \min_{\mathcal{D}} \ell(f, \mathcal{D}, R(T))$;
7: Sample $\mathcal{D}_g = \arg \min_{\mathcal{D}} \ell(g, \mathcal{D}, R(T))$;
8: Update $w_f = w_f - \eta \nabla f(\mathcal{D}_g)$;
9: Update $w_g = w_g - \eta \nabla g(\mathcal{D}_f)$;
10: Update $R(T) = 1 - \min \left\{ \frac{T^c}{T_k}, \tau, \tau \right\}$;
end
11: Output $w_f$ and $w_g$

Impact of memorization

$R(T)$: how fast drop
Experiments: $R(T)$ and $\tau$

<table>
<thead>
<tr>
<th></th>
<th>$c = 0.5$</th>
<th>$c = 1$</th>
<th>$c = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair-45%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_k = 5$</td>
<td>75.56%±0.33%</td>
<td>87.59%±0.26%</td>
<td>87.54%±0.23%</td>
</tr>
<tr>
<td>$T_k = 10$</td>
<td><strong>88.43%±0.25%</strong></td>
<td>87.56%±0.12%</td>
<td>87.93%±0.21%</td>
</tr>
<tr>
<td>$T_k = 15$</td>
<td><strong>88.37%±0.09%</strong></td>
<td>87.29%±0.15%</td>
<td><strong>88.09%±0.17%</strong></td>
</tr>
<tr>
<td>Symmetry-50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_k = 5$</td>
<td>91.75%±0.13%</td>
<td>91.75%±0.12%</td>
<td><strong>92.20%±0.14%</strong></td>
</tr>
<tr>
<td>$T_k = 10$</td>
<td>91.70%±0.21%</td>
<td>91.55%±0.08%</td>
<td>91.27%±0.13%</td>
</tr>
<tr>
<td>$T_k = 15$</td>
<td>91.74%±0.14%</td>
<td>91.20%±0.11%</td>
<td>91.38%±0.08%</td>
</tr>
<tr>
<td>Symmetry-20%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_k = 5$</td>
<td>97.05%±0.06%</td>
<td>97.10%±0.06%</td>
<td>97.41%±0.08%</td>
</tr>
<tr>
<td>$T_k = 10$</td>
<td>97.33%±0.05%</td>
<td>96.97%±0.07%</td>
<td><strong>97.48%±0.08%</strong></td>
</tr>
<tr>
<td>$T_k = 15$</td>
<td>97.41%±0.06%</td>
<td>97.25%±0.09%</td>
<td><strong>97.51%±0.05%</strong></td>
</tr>
</tbody>
</table>

- $R(T)$ and $\tau$ can influence the performance
- However, their sensitive is not high, and they can be easily set
- In previous experiments, we set $c = 1$ and $T_k = 10$
Agenda

• Application Background
• Related Works
• Co-teaching: Learning from noisy labels
• Experiments

• Summary
Future works

• Why and how co-teaching works? What are theoretical justifications?
• Can we extend from 2 classifiers to K (> 2) classifiers?
Summary

• Co-teaching is an effective deep learning training approach which can robustly learn from noisy labels.
• Co-teaching is built on SGD and small loss, and is independent of network structures.

Paper.
• Co-teaching: Robust training deep neural networks with extremely noisy labels. NeurIPS. 2018

Code.
• https://github.com/bhanML/Co-teaching

Thanks & QS