MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Jia Li, Fei-Fei Li.
Introduction
Deep Learning with Weak Supervision

**Problem**: big data often come with noisy or corrupted labels.
Deep Networks is able to memorize noisy labels

Zhang et al. (2017) showed that:

Deep networks are very good at overfitting (memorizing) the noisy labels.

We study how to train very deep networks on noisy data to improve the generalization performance.
We study how to train very deep networks on noisy data to improve the generalization performance.

Very deep CNN (ResNet-101 and Inception-resnet-v2)
- a few hundred layers
- tens of millions of model parameters
- takes days to train on 100+ GPUs
We study how to train very deep networks on **noisy data** to improve the generalization performance.

Training labels are noisy or corrupted:
- incorrect labels (false positives)
- missing labels (false negatives)
Related Work

- A robust loss to model “prediction consistency” (Reed et al. 2014)
- Add a layer to model noise labels (Goldberger et al. 2017)
- Regularization AIR (Azadi et al. 2016)
- Forgetting larger dropout rate (Kawaguchi et al. 2017)
- “Clean up” noisy labels using the clean labels (Veit. et al. 2017, Lee et al. 2017)
- Knowledge distillation (Li et al. 2017)
- Fidelity-weighting (Dehghani 2017)
- Robust Discriminative Neural Network (Vahdat 2017)
- … ..

Related Work

- A robust loss to model “prediction consistency” (Reed et al. 2014)
- Add a layer to model noise labels (Goldberger et al. 2017)
- Regularization AIR (Azadi et al. 2016)
- Forgetting larger dropout rate (Kawaguchi et al. 2017)
- “Clean up” noisy labels using the clean labels (Veit et al. 2017, Lee et al. 2017)
- Knowledge distillation (Li et al. 2017)

- Fidelity-weighting (Dehghani 2017)
- Robust Discriminative Neural Network (Vahdat 2017)

... ...

We focus on training very deep CNN from scratch.

Backgrounds on Curriculum Learning
Traditional Learning

- Clean label
- Corrupted label

Randomly shuffled examples.
Curriculum Learning

**Method:** Curriculum learning (learning examples with focus).
Introduce a *teacher* to determine the weight and timing to learn every example.

Background (Curriculum Learning)

\[
\min_{\mathbf{w} \in \mathbb{R}^d, \mathbf{v} \in [0,1]^{n \times m}} \mathbb{F}(\mathbf{w}, \mathbf{v}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_i^T \mathbf{L}(\mathbf{y}_i, g_s(\mathbf{x}_i, \mathbf{w})) + G(\mathbf{v}; \lambda) + \theta \| \mathbf{w} \|_2^2
\]
Background (Curriculum Learning)

\[
\min_{\mathbf{w} \in \mathbb{R}^{d}, \mathbf{v} \in [0,1]^{n \times m}} \mathcal{F}(\mathbf{w},\mathbf{v}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_{i}^{T} \mathbf{L}(\mathbf{y}_{i},g_{\mathbf{s}}(\mathbf{x}_{i},\mathbf{w})) + G(\mathbf{v}; \lambda) + \theta \| \mathbf{w} \|_{2}^{2}
\]

An example: self-paced (Kumar et al 2010)

\[
G'(\mathbf{v}) = -\|\mathbf{v}\|_{1} \quad \Rightarrow \quad \mathbf{v}_{i}^{*} = \begin{cases} 1 & \ell_{i} < \lambda \\ 0 & \ell_{i} \geq \lambda \end{cases}
\]

regularizer weighting scheme

Favor examples of smaller loss

\[
\ell_{i} = L(\mathbf{y}_{i},g_{\mathbf{s}}(\mathbf{x}_{i},\mathbf{w})))
\]

\[
\mathbf{v}_{i} \Rightarrow \mathbf{u}_{i}
\]

Existing studies define a curriculum as a function:

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Regularizer $G$</th>
<th>Weights $v^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self paced (Kumar et al., 2010)</td>
<td>$-|\mathbf{v}|_1$</td>
<td>$v_i^* = \mathbb{1}(\ell_i \leq \lambda)$</td>
</tr>
<tr>
<td>Linear weighting (Jiang et al., 2015)</td>
<td>$\frac{1}{2} \sum_{i=1}^{n} (v_i^2 - 2v_i)$</td>
<td>$v_i^* = \max(0, 1 - \frac{1}{X} \ell_i)$</td>
</tr>
<tr>
<td>Focal loss (Lin et al., 2017)</td>
<td>-</td>
<td>$v_i^* = [1 - \exp{-\ell_i}]^2$</td>
</tr>
<tr>
<td>Hard negative mining (Felzenszwalb et al., 2008)</td>
<td>-</td>
<td>$v_i^* = \mathbb{1}(\ell_i &gt; \lambda(1 - y_i))$</td>
</tr>
<tr>
<td>Prediction variance (Chang et al., 2017)</td>
<td>-</td>
<td>$v_i^* = \frac{1}{z} \sqrt{\text{Var}(\ell_i) + \frac{\text{Var}(\ell_i)}{</td>
</tr>
</tbody>
</table>

Weighting scheme: how to compute the weight
<table>
<thead>
<tr>
<th>Strategy</th>
<th>Regularizer $G$</th>
<th>Weights $v^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self paced (Kumar et al., 2010)</td>
<td>$\frac{1}{2} \sum_i v_i (v_i^2 - 2v_i)$</td>
<td>$v_i^* = \mathbb{1}(\ell_i \leq \lambda)$</td>
</tr>
<tr>
<td>Linear weighting (Jiang et al., 2015)</td>
<td></td>
<td>$v_i^* = \max(0, 1 - \frac{1}{\lambda} \ell_i)$</td>
</tr>
<tr>
<td>Focal loss (Lin et al., 2017)</td>
<td></td>
<td>$v_i^* = [1 - \exp(-\ell_i)]^2$</td>
</tr>
<tr>
<td>Hard negative mining (Felzenszwalb et al., 2008)</td>
<td></td>
<td>$v_i^* = \mathbb{1}(\ell_i &gt; \lambda(1 - y_i))$</td>
</tr>
<tr>
<td>Prediction variance (Chang et al., 2017)</td>
<td></td>
<td>$v_i^* = \frac{1}{Z} \sqrt{\text{Var}(\ell_i) + \frac{\text{Var}(\ell_i)}{</td>
</tr>
</tbody>
</table>

Learn a curriculum by a neural network from data?

**mini-batch** → **Network** → **weights**
Method
MentorNet

Each curriculum is implemented as a network (called MentorNet)
Learning the MentorNet

Two ways to learn a MentorNet

- **Pre-Defined**: approximate existing curriculums.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Regularizer $G$</th>
<th>Weights $v^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self paced (Kumar et al., 2010)</td>
<td>$-|v|_1$</td>
<td>$v_i^* = 1(\ell_i \leq \lambda)$</td>
</tr>
<tr>
<td>Linear weighting (Jiang et al., 2015)</td>
<td>$\frac{1}{2} \sum_{i=1}^n (v_i^2 - 2v_i)$</td>
<td>$v_i^* = \max(0, 1 - \frac{1}{2} \ell_i)$</td>
</tr>
<tr>
<td>Focal loss (Lin et al., 2017)</td>
<td>-</td>
<td>$v_i^* = [1 - \exp(-\ell_i)]^2$</td>
</tr>
<tr>
<td>Hard negative mining (Felzenszwalb et al., 2008)</td>
<td>-</td>
<td>$v_i^* = 1(\ell_i &gt; \lambda(1 - y_i))$</td>
</tr>
<tr>
<td>Prediction variance (Chang et al., 2017)</td>
<td>-</td>
<td>$v_i^* = \frac{1}{2} \sqrt{\text{Var}(\ell_i) + \frac{\text{Var}(\ell_i)}{|\ell_i|_1}}$</td>
</tr>
</tbody>
</table>

\[ \|v_i^* - g_m(z_i)\|_2^2 \]

optimal weight

MentorNet output

source: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis non erat sem learning the MentorNet.
Learning the MentorNet

Two ways to learn a MentorNet

- **Data-Driven**: learn new curriculums from a small set (~10%) with clean labels.

Learn the MentorNet of CIFAR-10 and use it on CIFAR-100.
Algorithm (Training MentorNet with StudentNet)

Existing curriculum/self-paced learning is optimized by alternating batch GD.

We propose a **mini-batch SGD** and show good property on **model convergence** under standard assumptions.

**Theorem 1.** Let the objective $F(w, v)$ defined in Eq. (1) be differentiable, $L(\cdot)$ be Lipschitz continuous in $w$ and $\nabla_v G(\cdot)$ be Lipschitz continuous in $v$. Let $w^t, v^t$ be iterates from Algorithm 1 and $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$. Then, $\lim_{t \to \infty} \mathbb{E}[\|\nabla_w F(w^t, v^t)\|_2^2] = 0$.

**Algorithm 1** SPADE for minimizing Eq. (1)

**Input**: Dataset $D$, a predefined $G$ or a learned $g_m(\cdot; \Theta)$

**Output**: The model parameter $w$ of StudentNet.

1. Initialize $w^0, v^0, t = 0$
2. **while Not Converged do**
3. Fetch a mini-batch $\Xi_t$ uniformly at random
4. For every $(x_i, y_i)$ in $\Xi_t$ compute $\phi(x_i, y_i, w^t)$
5. **if update curriculum then**
6. $\Theta \leftarrow \Theta^*$, where $\Theta^*$ is learned in Sec. 3.1
7. **end**
8. **if $G$ is used then**
9. $v^t_{\Xi} \leftarrow v^t_{\Xi} - \alpha_t \nabla_v F(w^{t-1}, v^{t-1})|_{\Xi_t}$
10. **end**
11. **else** $v^t_{\Xi} \leftarrow g_m(\phi(\Xi_t, w^{t-1}); \Theta)$
12. $w^t \leftarrow w^{t-1} - \alpha_t \nabla_w F(w^{t-1}, v^t)|_{\Xi_t}$
13. $t \leftarrow t + 1$
14. **end**
15. **return** $w^t$
Why does it work?

We can show that MentorNet plays a role similar to the robust M-estimator:

- Huber (Huber et al., 1964)
- The log-sum penalty (Candes et al., 2008).

The robust objective function is beneficial for noisy labels.

\[ \textbf{Proposition 1.} \text{ Suppose } (x, y) \text{ denotes a training sample and its corrupted label. For simplicity, let the MentorNet input } \phi(x, y, w) = \ell \text{ be the loss computed by the StudentNet model parameter } w. \text{ The MentorNet } g_m(\ell; \Theta) = v, \text{ where } v \text{ is the sample weight. If } g_m \text{ decreases with respect to } \ell, \text{ then there exists an underlying robust objective } F: } \]

\[ F(w) = \frac{1}{n} \sum_{i=1}^{n} \rho(\ell_i), \]

where \( \rho(\ell) = \int_{0}^{\ell_i} g_m(x; \Theta)dx. \) In the special cases, \( \rho(\ell) \) degenerates to the robust M-estimator: Huber (Huber et al., 1964) and the log-sum penalty (Candes et al., 2008).
Experiments
Setups

### Table 1. StudentNet and their accuracies on the clean training data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>#para</th>
<th>train acc</th>
<th>val acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>inception</td>
<td>1.7M</td>
<td>0.83</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>resnet101</td>
<td>84M</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>inception</td>
<td>1.7M</td>
<td>0.64</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>resnet101</td>
<td>84M</td>
<td>1.00</td>
<td>0.79</td>
</tr>
<tr>
<td>ImageNet</td>
<td>inception_resnet</td>
<td>59M</td>
<td>0.88</td>
<td>0.77</td>
</tr>
</tbody>
</table>
First, following (Zhang et al. 2017), we test on controlled corrupted labels:

- Change each label of an image dependently to a uniform random class with probability $p$.
- $p$ is called “noise fraction”.

Results on CIFAR

Baseline comparisons on **CIFAR-10 & CIFAR-100**
(under 20%, 40%, and 80% noise fractions)

<table>
<thead>
<tr>
<th>Method</th>
<th>Resnet-101 CIFAR-100</th>
<th>StudentNet CIFAR-10</th>
<th>Inception CIFAR-100</th>
<th>StudentNet CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2 0.4 0.8</td>
<td>0.2 0.4 0.8</td>
<td>0.2 0.4 0.8</td>
<td>0.2 0.4 0.8</td>
</tr>
<tr>
<td>FullModel</td>
<td>0.60 0.45 0.08</td>
<td>0.82 0.69 0.18</td>
<td>0.43 0.38 0.15</td>
<td>0.76 0.72 0.42</td>
</tr>
<tr>
<td>Forgetting</td>
<td>0.61 0.44 0.16</td>
<td>0.78 0.63 0.35</td>
<td>0.42 0.37 0.17</td>
<td>0.76 0.71 0.44</td>
</tr>
<tr>
<td>Self-paced</td>
<td>0.70 0.55 0.13</td>
<td>0.89 0.85 0.28</td>
<td>0.44 0.38 0.14</td>
<td>0.80 0.74 0.33</td>
</tr>
<tr>
<td>Focal Loss</td>
<td>0.59 0.44 0.09</td>
<td>0.79 0.65 0.28</td>
<td>0.43 0.38 0.15</td>
<td>0.77 0.74 0.40</td>
</tr>
<tr>
<td>Reed Soft</td>
<td>0.62 0.46 0.08</td>
<td>0.81 0.63 0.18</td>
<td>0.42 0.39 0.12</td>
<td>0.78 0.73 0.39</td>
</tr>
<tr>
<td>MentorNet PD</td>
<td>0.72 0.56 0.14</td>
<td>0.91 0.77 0.33</td>
<td>0.44 0.39 0.16</td>
<td>0.79 0.74 0.44</td>
</tr>
<tr>
<td>MentorNet DD</td>
<td><strong>0.73 0.68 0.35</strong></td>
<td><strong>0.92 0.89 0.49</strong></td>
<td><strong>0.46 0.41 0.20</strong></td>
<td><strong>0.79 0.76 0.49</strong></td>
</tr>
</tbody>
</table>

Significant improvement over baselines.
**Data-Driven** performs better than **Pre-Defined MentorNet.**
Results on ImageNet

**ImageNet** under 40% noise fraction

<table>
<thead>
<tr>
<th>Method</th>
<th>P@1</th>
<th>P@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoReg</td>
<td>0.538</td>
<td>0.770</td>
</tr>
<tr>
<td>NoReg+WeDecay</td>
<td>0.560</td>
<td>0.809</td>
</tr>
<tr>
<td>NoReg+Dropout</td>
<td>0.575</td>
<td>0.807</td>
</tr>
<tr>
<td>NoReg+DataAug</td>
<td>0.522</td>
<td>0.772</td>
</tr>
<tr>
<td>NoReg+MentorNet</td>
<td>0.590</td>
<td>0.814</td>
</tr>
<tr>
<td>FullModel</td>
<td>0.612</td>
<td>0.844</td>
</tr>
<tr>
<td>Forgetting(FullModel)</td>
<td>0.628</td>
<td>0.845</td>
</tr>
<tr>
<td>MentorNet(FullModel)</td>
<td>0.651</td>
<td>0.859</td>
</tr>
</tbody>
</table>

**NoReg**: Vanilla model with no regularization

**FullModel**: Added weight decay, dropout, and data augmentation.
Results on WebVision

2.4 million images of noisy labels crawled by Flickr/Google Search.
1,000 classes defined in ImageNet ILSVRC 2012.
Real-world noisy labels.

Results on WebVision

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>ILSVRC12</th>
<th>WebVision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire</td>
<td>Li et al. (2017a)</td>
<td>0.476 (0.704)</td>
<td>0.570 (0.779)</td>
</tr>
<tr>
<td>Entire</td>
<td>Forgetting</td>
<td>0.590 (0.808)</td>
<td>0.666 (0.856)</td>
</tr>
<tr>
<td>Entire</td>
<td>Lee et al. (2017)*</td>
<td>0.602 (0.811)</td>
<td>0.685 (0.865)</td>
</tr>
<tr>
<td>Entire</td>
<td>MentorNet</td>
<td>0.625 (0.830)</td>
<td>0.708 (0.880)</td>
</tr>
<tr>
<td>Entire</td>
<td>MentorNet*</td>
<td><strong>0.642 (0.848)</strong></td>
<td><strong>0.726 (0.889)</strong></td>
</tr>
</tbody>
</table>

The best-published result on the WebVision benchmark!

Substantiate that MentorNet is beneficial for training very deep networks on noisy data.

Conclusions
Take-home messages

We discussed a novel way to train very deep networks on noisy data.

- The generalization performance can be improved by learning another network (called MentorNet) to supervise the training of the base network.
- Proposed an algorithm to perform curriculum learning and self-paced learning for deep networks via mini-batch SGD.

Code will be available soon on Google Github.

Please come to our Poster: #113 (Hall B)
MentorNet is a general and flexible framework:

- Each curriculum is implemented as a network.
  - Approximate existing curriculums
  - Discover new curriculums from data
- Plug in/out a MentorNet to change the curriculum
Results

Our method overcomes the “memorization” problem of deep networks.
Algorithm (Training MentorNet with StudentNet)

Training MentorNet and StudentNet via mini-batch SGD.

Algorithm 1 SPADE for minimizing Eq. (1)

Input: Dataset $D$, a predefined $G$ or a learned $g_m(\cdot; \Theta)$
Output: The model parameter $w$ of StudentNet.

Initialize $w^0$, $v^0$, $t = 0$

while Not Converged do
  Fetch a mini-batch $\Xi_t$ uniformly at random
  For every $(x_i, y_i)$ in $\Xi_t$ compute $\phi(x_i, y_i, w^t)$
  if update curriculum then
    $\Theta \leftarrow \Theta^*$, where $\Theta^*$ is learned in Sec. 3.1
  end
  if $G$ is used then
    $v^{t-1}_\Xi \leftarrow v^{t-1}_\Xi - \alpha_t \nabla v^\Xi_{\Xi_t}(w^{t-1}, v^{t-1})|_{\Xi_t}$
  end
  else
    $v^{t}_\Xi \leftarrow g_m(\phi(\Xi_t, w^{t-1}); \Theta)$;
    $w^t \leftarrow w^t - \alpha_t \nabla w^\Xi_{\Xi_t}(w^{t-1}, v^t)|_{\Xi_t}$
  end
  $t \leftarrow t + 1$
end

return $w^t$
Algorithm (Training MentorNet with StudentNet)

Training MentorNet and StudentNet via mini-batch SGD.

---

**Algorithm 1** SPADE for minimizing Eq. (1)

**Input**: Dataset $D$, a predefined $G$ or a learned $g_m(\cdot; \Theta)$

**Output**: The model parameter $w$ of StudentNet.

1. Initialize $w^0, v^0, t = 0$
2. while Not Converged do
3.   Fetch a mini-batch $\Xi_t$ uniformly at random
4.   For every $(x_i, y_i) \in \Xi_t$ compute $\phi(x_i, y_i, w^t)$
5.   if update curriculum then
6.     $\Theta \leftarrow \Theta^*$, where $\Theta^*$ is learned in Sec. 3.1
7.   end
8.   if $G$ is used then
9.     $v^t_{\Xi} \leftarrow v^t_{\Xi} - \alpha_t \nabla_{v^t} F(w^{t-1}, v^{t-1})_{\Xi_t}$
10. end
11. else $v^t_{\Xi} \leftarrow a_m(\phi(\Xi_t, w^{t-1}; \Theta); v^t)$
12. $w^t \leftarrow w^{t-1} - \alpha_t \nabla_{w^t} F(w^{t-1}, v^t)_{\Xi_t}$
13. end
14. $t \leftarrow t + 1$
15. return $w^t$
Algorithm (Training MentorNet with StudentNet)

Algorithm 1 SPADE for minimizing Eq. (1)

Input : Dataset $\mathcal{D}$, a predefined $G$ or a learned $g_m(\cdot; \Theta)$
Output : The model parameter $w$ of StudentNet.

1. Initialize $w^0$, $v^0$, $t = 0$
2. while Not Converged do
3. Fetch a mini-batch $\Xi_t$ uniformly at random
4. For every $(x_i, y_i)$ in $\Xi_t$ compute $\phi(x_i, y_i, w^t)$
5. if update curriculum then
6. $\Theta \leftarrow \Theta^*$, where $\Theta^*$ is learned in Sec. 3.1
7. end
8. if $G$ is used then
9. $v^t_{\Xi} \leftarrow v^t_{\Xi} - \alpha_t \nabla_v \mathbb{F}(w^{t-1}, v^{t-1})|_{\Xi_t}$
10. end
11. else $v^t_{\Xi} \leftarrow g_m(\phi(\Xi_t, w^{t-1}); \Theta)$;
12. $w^t \leftarrow w^{t-1} - \alpha_t \nabla_w \mathbb{F}(w^{t-1}, v^t)|_{\Xi_t}$
13. end
14. $t \leftarrow t + 1$
15. return $w^t$