Tri-net for Semi-Supervised Deep Learning*

Dong-Dong Chen, Wei Wang, Wei Gao, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{chendd, wangw, gaow, zhouzh}@lamda.nju.edu.cn

Abstract

Deep neural networks have witnessed great successes in various real applications, but it requires a large number of labeled data for training. In this paper, we propose tri-net, a deep neural network which is able to use massive unlabeled data to help learning with limited labeled data. We consider model initialization, diversity augmentation and pseudo-label editing simultaneously. In our work, we utilize output smearing to initialize modules, use fine-tuning on labeled data to augment diversity and eliminate unstable pseudo-labels to alleviate the influence of suspicious pseudo-labeled data. Experiments show that our method achieves the best performance in comparison with state-of-the-art semi-supervised deep learning methods. In particular, it achieves 8.30% error rate on CIFAR-10 by using only 4000 labeled examples.

1 Introduction

Deep neural networks (DNNs) have become a hot wave during the past few years, and great successes have been achieved in various real applications, such as image classification [Krizhevsky et al., 2012], object detection [Girshick et al., 2014], scene labeling [Shelhamer et al., 2017], etc. DNNs always learn a large number of parameters requiring a large amount of labeled data to alleviate overfitting. It is well-known that collecting tremendous high-quality labeled data is expensive, yet we could easily collect abundant unlabeled data in many real applications. Hence, it is desirable to use unlabeled data to improve the performance of DNNs when training with limited labeled data.

A natural idea is to combine semi-supervised learning [Chapelle et al., 2006; Zhu, 2007; Zhou and Li, 2010] with deep learning. The disagreement-based learning [Zhou and Li, 2010] plays an important role in semi-supervised learning, in which co-training [Blum and Mitchell, 1998] and tri-training [Zhou and Li, 2005b] are two representatives. The basic idea of disagreement-based semi-supervised learning is to train multiple learners for the task and exploit the disagreements during the learning process. The disagreement in co-training is based on different views, while tri-training uses bootstrap sampling to get diverse training sets. Co-training has been combined with deep model for the tasks which have two views [Cheng et al., 2016; Ardehaly and Culotta, 2017]. Nevertheless, in real applications, we always confront the task with one-view data, and tri-training can be utilized no matter whether there are one or more views.

In this paper, we propose tri-net which combines tri-training with deep model. We first learn three initial modules, and each module is then used to predict a pool of unlabeled data, where two modules label some unlabeled instances for another module. Later, three modules are refined by using the newly labeled examples. We consider three key techniques in tri-net, i.e., model initialization, diversity augmentation and pseudo-label editing, which can be summarized as follows: we use output smearing [Breiman, 2000] to help generate diverse and accurate initial modules; we fine-tune the modules in some specific rounds on labeled data to augment the diversity among them; we propose a data editing method named DES based on the intuition that stable pseudo-labels are more reliable. Experiments are conducted on three benchmark datasets, i.e., MNIST, SVHN and CIFAR-10, and the results demonstrate that our tri-net has good performance on all datasets. In particular, it achieves 8.45% error rate on CIFAR-10 by using only 4,000 labeled examples. With more sophisticated initialization methods, tri-net can get even better performance. For example, when we use the semi-supervised deep learning method II model [Laine and Aila, 2016] to initialize our tri-net, we can achieve 8.30% error rate on CIFAR-10 by using only 4,000 labeled examples.

The rest of this paper is organized as follows: we introduce related work in Section 2 and present our tri-net in Section 3. Experimental results are given in Section 4. Finally, we make a conclusion in Section 5.

2 Related Work

Many methods have been proposed to tackle semi-supervised learning, we only introduce the most related ones. For more information of semi-supervised learning, see [Chapelle et al., 2006; Zhu, 2007; Zhou and Li, 2010]. Disagreement-based semi-supervised learning started from the seminal paper of Blum and Mitchell [1998] on co-

*This work was supported by the NSFC (61751306, 61673202, 61503179), the Jiangsu Science Foundation (BK20150586) and the Fundamental Research Funds for the Central Universities.
training. Co-training first learns two classifiers from two views and then lets them label unlabeled data for each other to improve performance. However, in most real applications the data sets have only one view rather than two. Some methods employed different learning algorithms or different parameter configurations to learn two different classifiers [Goldman and Zhou, 2000; Zhou and Li, 2005a]. Although these methods do not rely on the existence of two views, they require special learning algorithms to construct classifiers. Zhou and Li [2005b] proposed tri-training, which utilizes bootstrap sampling to get three different training sets and generates three classifiers from these three training sets respectively. Tri-training requires neither the existence of multiple views nor special learning algorithms, thus it can be applied to more real applications. For these algorithms, there have been some theoretical studies to explain why unlabeled data can improve the learning performance [Blum and Mitchell, 1998; Balcan et al., 2004; Wang and Zhou, 2010; Balcan and Blum, 2010].

With the fast development of deep learning, disagreement-based semi-supervised learning has been combined with deep model for some applications. Cheng et al. [2016] developed a semi-supervised multimodal deep learning framework based on co-training to deal with the RGB-D object-recognition task. They utilized each view (i.e., RGB and depth) to learn a DNN and the two DNNs labeled unlabeled data to augment the training set. Ardehaly and Culotta [2017] combined co-training with deep model to address the demographic classification task. They generated two DNNs from two views (i.e., image and text) respectively and let them provide pseudo-labels for each other. Nevertheless, many tasks have only one view in real applications. It is more desirable to develop the disagreement-based deep models for one-view data.

There are many other methods in semi-supervised deep learning. Some of them were based on generative models. These methods paid efforts to learn the input distribution $p(x)$. Variational auto-encoder (VAE) combined variational methods with DNNs to help estimate $p(x)$ [Kingma et al., 2014; Maaloe et al., 2016] while generative adversarial networks (GANs) aimed to leverage a generator to detect the low-density boundaries [Salimans et al., 2016; Dai et al., 2017]. In contrast to the generative nature, our tri-net is a discriminative model and does not need to estimate $p(x)$. Some combined graph-based methods with deep neural networks [Weston et al., 2012; Luo et al., 2017]. They enforced smoothness of the predictions with respect to the graph structure while we do not need to construct the graph. Some were perturbation-based discriminative methods. They utilized local variations of the input to regularize the output to be smooth [Bachman et al., 2014; Rasmus et al., 2015; Laine and Aila, 2016; Sajjadi et al., 2016]. VAT [Miyato et al., 2017] and VAdD [Park et al., 2018] introduced adversarial training [Goodfellow et al., 2014] into these methods while temporal ensembling [Laine and Aila, 2016] and mean teacher [Tarvainen and Valpola, 2017] introduced ensemble learning [Zhou, 2012] into them. Compared with these state-of-the-art methods, our method can achieve better performance.

3 Our Approach

3.1 Overview

In semi-supervised learning, we have a small labeled data set $L = \{ (x_l, y_l) | l = 1, 2, \ldots, L \}$ with $L$ labeled examples and a large-scale unlabeled data set $U = \{ (x_u) | u = 1, 2, \ldots, U \}$ with $U$ unlabeled instances. Suppose the data have $C$ classes and $y_i = (y_{i1}, y_{i2}, \ldots, y_{ic})$, where $y_{ic} = 1$ if the example belongs to the $c$-th class otherwise $y_{ic} = 0$, for $c = 1, 2, \ldots, C$. Our goal is to learn a model from the training set $L \cup U$ to classify unseen instances. In this paper, we propose tri-net by combining tri-training with deep neural network. Our tri-net has three phases which are described as follows.

**Initialization.** The first step in tri-net is to generate three accurate and diverse modules. Instead of training three networks separately, tri-net is one DNN which is composed of a shared module $M_S$ and three different modules $M_1$, $M_2$ and $M_3$. Here, $M_1$, $M_2$ and $M_3$ classify the shared features generated by $M_S$. This network structure is inspired by Saito et al. [2017] and is efficient for implementation. In order to get three accurate and diverse modules, we use output smearing (Section 3.2) to generate three different labeled data sets, i.e., $L_{os}^1$, $L_{os}^2$ and $L_{os}^3$. We train $M_S$, $M_1$, $M_2$ and $M_3$ simultaneously on the three data sets. Specifically, $M_S$ and $M_v$ are trained on $L_{os}^v$ ($v = 1, 2, 3$).

**Training.** In the training process, some unlabeled data will be labeled and added into the labeled training sets. In order not to change the distribution of labeled training sets, we assume that the unlabeled data are selected from a pool of $U$. We use $N$ to denote the size of the pool. This strategy is widely used in semi-supervised learning [Blum and Mitchell, 1998; Zhou and Li, 2005a; Saito et al., 2017]. With three modules, if two modules agree on the prediction of the unlabeled instance from the pool and the prediction is confident...
Algorithm 1 Tri-net

Input:
Labeled set \( L \) and unlabeled set \( U \)

Labeling: the methods of labeling when the predictions of two classifiers are confident and agree with each other

DES: the methods of pseudo-label editing

\( \sigma_0 \): the initial threshold parameter for filtrating the unconfident pseudo-labels

\( \sigma_{os} \): the value to decrease \( \sigma \) if output smearing is used in this learning round

Output:
Tri-net: the model composed of \( M_{S}, M_{1}, M_{2} \) and \( M_{3} \)

1: Initialization:
2: Generate \( \{L_{os}^{1}, L_{os}^{2}, L_{os}^{3}\} \) by using output smearing on \( L \)
3: Train \( M_{S}, M_{1}, M_{2}, M_{3} \) with mini-batch from training set \( L_{os}^{-1} \), \( L_{os}^{2}, L_{os}^{3} \)
4: \( flag_{os} = 1; \sigma = \sigma_0 \)
5: Training:
6: for \( t = 1 \rightarrow T \) do
7: \( N_t = \text{min}(1000 \times 2^t, U) \)
8: if \( N_t = U \) then
9: \( \text{if mod}(t, 4) = 0 \) then
10: Train \( M_{S}, M_{1}, M_{2}, M_{3} \) with mini-batch from training set \( L_{os}^{1}, L_{os}^{2}, L_{os}^{3} \)
11: \( flag_{os} = 1; \sigma = \sigma - 0.05 \)
12: continue
13: if \( flag_{os} = 1 \) then
14: \( flag_{os} = 0; \sigma_{t} = \sigma - \sigma_{os} \)
15: else
16: \( \sigma_{t} = \sigma \)
17: for \( v = 1 \rightarrow 3 \) do
18: \( \mathcal{P}C_v \leftarrow \emptyset \)
19: \( \mathcal{P}C_v \leftarrow \text{Labeling}(M_{S}, M_{1}, M_{2}, U, N_t, \sigma_{t})(j, h \neq v) \)
20: \( \mathcal{P}C_v \leftarrow \text{DES}(M_{S}, \mathcal{P}C_v, M_{j}, M_{h}) \)
21: \( \hat{L}_v \leftarrow L \cup \mathcal{P}C_v \)
22: if \( v = 1 \) then
23: Train \( M_{S}, M_{v} \) with mini-batch from training set \( \hat{L}_v \)
24: else
25: Train \( M_{v} \) with mini-batch from training set \( \hat{L}_v \)
26: return \( M_{S}, M_{1}, M_{2} \) and \( M_{3} \)

3.2 Output Smearing

Output smearing was proposed by Breiman [2000]. It constructs diverse training sets by injecting random noise into true labels and generates modules from the diverse training sets respectively. Injecting noise into true labels can also regularize the modules by smoothing the labels [Szegedy et al., 2016]. We apply this technique to initialize our modules \( M_{1}, M_{2} \) and \( M_{3} \). For an example \( \{x, y\} \) \((l = 1, 2, \ldots, L)\), where \( y_i = (y_{1i}, y_{2i}, \ldots, y_{ci}) \), \( y_{ci} = 1 \) if the example belongs to the \( c \)-th class otherwise \( y_{ci} = 0 \). In output smearing, we add noise into every component of \( y_i \).

\[
\hat{y}_{ci} = y_{ci} + \text{ReLU}(z_{ci} \times \text{std})
\]

(2)

where \( z_{ci} \) is sampled independently from the standard normal distribution, \( \text{std} \) is the standard deviation, ReLU is a function

\[
\text{ReLU}(a) = \begin{cases} 
    a, & a > 0 \\
    0, & a \leq 0.
\end{cases}
\]

Here, we use ReLU function to ensure \( \hat{y}_{ci} \) non-negative and normalize \( \hat{y}_{ci} \) according to Eq. 4.

\[
\hat{y}_{c} = (\hat{y}_{1c}, \hat{y}_{2c}, \ldots, \hat{y}_{Cc})/\sum_{c=1}^{C} \hat{y}_{c}.
\]

(4)

With output smearing, we construct three diverse training sets \( L_{os}^{1}, L_{os}^{2} \) and \( L_{os}^{3} \) from the initial labeled data set \( L \), where \( L_{os}^{i} = \{(x, y_i) | 1 \leq l \leq L \} \) \((v = 1, 2, 3)\) is constructed by output smearing and \( y_i \) is calculated according to Eq. 4. Then we initialize tri-net with \( L_{os}^{1}, L_{os}^{2} \) and \( L_{os}^{3} \) by minimizing Loss shown in Eq. 5.

\[
\text{Loss} = \frac{1}{L} \sum_{l=1}^{L} \left\{ L_{y}(M_{1}(M_{S}(x_l)), \hat{y}_{l}^{1}) + L_{y}(M_{2}(M_{S}(x_l)), \hat{y}_{l}^{2}) + L_{y}(M_{3}(M_{S}(x_l)), \hat{y}_{l}^{3}) \right\}
\]

(5)

Here, \( L_{y} \) denotes the standard softmax cross-entropy loss function, \( M_{S} \) denotes the shared module, \( M_{1}, M_{2} \) and \( M_{3} \) denote the three modules in tri-net, \( M_{v}(M_{S}(x)) \) denotes the output of \( M_{v} \) on \( x \) where \( M_{v} \) classifies the features generated by \( M_{S} \) on \( x \) \((v = 1, 2, 3)\).

3.3 Diversity Augmentation

Diversity among three modules in tri-net plays an important role in the training process. When three modules label unlabeled data to augment the training sets of one another, they become more and more similar. In order to maintain the diversity, we fine-tune three modules \( M_{1}, M_{2} \) and \( M_{3} \) on the diverse training sets \( L_{os}^{1}, L_{os}^{2} \) and \( L_{os}^{3} \) in some specific rounds. In the experiments, the fine-tuning is executed every 3 rounds, which will be described in Section 4.
3.4 Pseudo-Label Editing

The pseudo-labels of the newly labeled examples may be incorrect, and these incorrect pseudo-labels will degenerate the performance. Data editing which can deal with the suspicious pseudo-labels is important and there have been some data-editing methods in semi-supervised learning [Zhang and Zhou, 2011]. However, these existing methods are usually based on graph and are difficult to be used in DNNs due to the high dimension. Here, we propose a new data-editing method for DNNs with dropout [Srivastava et al., 2014]. Generally, dropout works in two modes: at training mode, the connections of the network are different in every forward pass; at test mode, the connections are fixed. This means that the prediction for dropout working in training mode may change. For each \((x_i, \bar{y}_i)\), \(\bar{y}_i\) is the pseudo-label predicted by the modules working in test mode. We use dropout working in train mode to measure the stability of the pseudo-labeled data, i.e., we use the modules to predict the label of \(x_i\) for \(K\) times in training mode and record the frequency \(k\) that the prediction is different from \(\bar{y}_i\). If \(k > \frac{K}{2}\), we regard the pseudo-label \(\bar{y}_i\) of \(x_i\) as an unstable pseudo-label. For these unstable pseudo-labels, we will eliminate them. We set \(K = 9\) in all experiments.

4 Experiments

4.1 Setup

Datasets. We run experiments on three widely used benchmark datasets, i.e., MNIST, SVHN, and CIFAR-10. We randomly sample 100, 1,000, and 4,000 labeled examples from MNIST, SVHN and CIFAR-10 as the initial labeled data set \(\mathcal{L}\) respectively and use the standard data split for testing as that in previous work.

Network Architectures. The network architecture of tri-net for CIFAR-10 is shown in Figure 2, which is derived from the popular architecture [Laine and Aila, 2016] used in semi-supervised deep learning. In order to get more diversity among three modules, we use different convolution kernel sizes, different network structures (with/without residual block) and different depths for \(M_1\), \(M_2\) and \(M_3\). The network architectures for MNIST and SVHN are similar to that in Figure 2 but in a smaller size.

Parameters. In order to prevent the network from overfitting, we gradually increase the pool size \(N = 1000 \times 2^t\) up to the size of unlabeled data \(U\) [Saito et al., 2017], where \(t\) denotes the learning round. The maximal learning round \(T\) is set to be 30 in all experiments. We gradually decrease the confidence threshold \(\sigma\) after \(N = U\) to make more unlabeled data to be labeled (line 11, Algorithm 1). In the training process, we respectively fine-tune three modules \(M_1\), \(M_2\) and \(M_3\) on the diverse training sets \(L_{os}^1\), \(L_{os}^2\) and \(L_{os}^3\) every 3 rounds after \(N = U\) to maintain the diversity (line 10, Algorithm 1). Since \(L_{os}^1\), \(L_{os}^2\) and \(L_{os}^3\) are injected into random noise, the confidence threshold \(\sigma\) is decreased by \(\sigma_{os}\) (line 14, Algorithm 1). We set \(\sigma_0 = 0.999\) and \(\sigma_{os} = 0.01\) in MNIST; \(\sigma_0 = 0.95\) and \(\sigma_{os} = 0.25\) in SVHN and CIFAR-10. We use dropout \((p = 0.5)\) after each max-pooling layer, use Leaky-ReLU \((\alpha = 0.1)\) as activate function except the FC layer, and use soft-max for FC layer. We also use Batch Normalization [Ioffe and Szegedy, 2015] for all layers except the FC layer. We use SGD with a mini-batch size of 16. The learning rate starts from 0.1 in initialization (from 0.02 in training) and is divided by 10 when the error plateaus. In initialization, three modules \(M_1\), \(M_2\) and \(M_3\) are trained for up to 300 epochs in SVHN and CIFAR-10 (100 in MNIST). In training, three modules \(M_1\), \(M_2\) and \(M_3\) are trained for up to 90 epochs in SVHN and CIFAR-10 (60 in MNIST). We set \(std = 0.05\) in SVHN and CIFAR-10 (0.001 in MNIST). We use a weight decay of 0.0001 and a momentum of 0.9. Following the setting in Laine and Aila [2016], we use ZCA, random crop and horizon flipping for CIFAR-10, zero-mean normalization and random crop for SVHN.

4.2 Results

We compare our tri-net with state-of-the-art methods shown in Table 1. Recently, Abbasnejad et al. [2017] exploited a pre-trained model in their infinite Variational Autoencoder (infinite VAE) method, however, the state-of-the-art methods did not use the pre-trained model. To make a fair comparison, we do not exploit the pre-trained model as that in state-of-the-art methods. The results in Table 1 indicate that tri-net has good performance. It achieves the error rate of 0.53% on
MNIST with 100 labeled examples and 8.45% error rate on CIFAR-10 with 4000 labeled examples, which are much better than state-of-the-art methods. Since tri-net exploits three modules while the state-of-the-art methods exploit one or two modules, the time cost of tri-net is more than that of these methods.

There is an initialization in tri-net, with more sophisticated initialization methods, tri-net could have better performance. 

1

2

3

MNIST, SVHN, and CIFAR-10. * indicates that the method does not use data augmentation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST (L = 100)</th>
<th>SVHN (L = 1000)</th>
<th>CIFAR-10 (L = 4000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ladder network [Rasmus et al., 2015]</td>
<td>0.89 ± 0.50</td>
<td>-</td>
<td>20.40 ± 0.47*</td>
</tr>
<tr>
<td>GoodSemiBadGan [Dai et al., 2017]</td>
<td>0.795 ± 0.098</td>
<td>4.25 ± 0.03*</td>
<td>14.41 ± 0.03*</td>
</tr>
<tr>
<td>II model [Laine and Aila, 2016]</td>
<td>-</td>
<td>4.82 ± 0.17</td>
<td>12.36 ± 0.31</td>
</tr>
<tr>
<td>Temporal ensembling [Laine and Aila, 2016]</td>
<td>-</td>
<td>4.42 ± 0.16</td>
<td>12.16 ± 0.24</td>
</tr>
<tr>
<td>Mean teacher [Tarvainen and Valpola, 2017]</td>
<td>-</td>
<td>3.95 ± 0.19</td>
<td>12.31 ± 0.28</td>
</tr>
<tr>
<td>VAT + EntMin [Miyato et al., 2017]</td>
<td>-</td>
<td>3.86</td>
<td>10.55</td>
</tr>
<tr>
<td>II + SNTG [Luo et al., 2017]</td>
<td>0.66 ± 0.07</td>
<td>3.82 ± 0.25</td>
<td>11.00 ± 0.13</td>
</tr>
<tr>
<td>VAdd(KL)+VAT [Park et al., 2018]</td>
<td>-</td>
<td>3.55 ± 0.05</td>
<td>9.22 ± 0.10</td>
</tr>
<tr>
<td>Tri-net</td>
<td>0.53 ± 0.10</td>
<td>3.71 ± 0.14</td>
<td>8.45 ± 0.22</td>
</tr>
<tr>
<td>Tri-net + II model</td>
<td>0.52 ± 0.05</td>
<td>3.45 ± 0.10</td>
<td>8.30 ± 0.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>datasets</th>
<th>MNIST</th>
<th>SVHN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>err</td>
<td>agr</td>
<td>err</td>
</tr>
<tr>
<td>without output smearing</td>
<td>8.55 ± 0.00</td>
<td>85.69 ± 0.50</td>
<td>12.47 ± 0.12</td>
</tr>
<tr>
<td>with output smearing</td>
<td>7.85 ± 0.48</td>
<td>86.52 ± 0.55</td>
<td>12.20 ± 0.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>datasets</th>
<th>MNIST</th>
<th>SVHN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>with the same structure</td>
<td>0.60</td>
<td>3.95</td>
<td>9.05</td>
</tr>
<tr>
<td>with different structures</td>
<td>0.53</td>
<td>3.71</td>
<td>8.45</td>
</tr>
</tbody>
</table>

Different network structures are used to get three diverse modules $M_1$, $M_2$, and $M_3$, we conduct the experiments with the same network structure for three modules $M_1$, $M_2$, and $M_3$ as a comparison. The results shown in Table 3 indicate that different structures bring better performance. The parameter $\sigma_{os}$ controls the confidence threshold when output smearing is used in the training process. We conduct the experiments with different $\sigma_{os} \in [0.01, 0.25]$, and the results
shown in Table 4 indicate that tri-net is not very sensitive to the parameter $\sigma_{os}$.

<table>
<thead>
<tr>
<th>$\sigma_{os}$</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.53</td>
<td>0.55</td>
<td>0.58</td>
<td>0.60</td>
</tr>
<tr>
<td>SVHN</td>
<td>4.23</td>
<td>4.09</td>
<td>3.81</td>
<td>3.71</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>9.38</td>
<td>9.10</td>
<td>8.65</td>
<td>8.45</td>
</tr>
</tbody>
</table>

Table 4: Error rates (%) of tri-net with different $\sigma_{os}$.

5 Conclusion

In this paper, we propose tri-net for semi-supervised deep learning, in which we generate three modules to exploit unlabelled data by considering model initialization, diversity augmentation and pseudo-label editing simultaneously. Experiments on several benchmarks demonstrate that our method is superior to state-of-the-art semi-supervised deep learning methods. In particular, it can achieve the error rate of 8.30% on CIFAR-10 by using only 4000 labeled examples. Extending tri-net with more modules could exploit the power of ensemble in labeling the unlabeled data confidently. In this situation, one important issue is to maintain the diversity among these modules, which will be an interesting research direction in semi-supervised deep learning.

References


