Deep Semi-Supervised Learning

Conference Paper · August 2018

DOI: 10.1109/ICPR.2018.8546327

3 authors, including:

Zeyad Hailat
Wayne State University
10 PUBLICATIONS **88** CITATIONS

SEE PROFILE

Deep Semi-Supervised Learning

Zeyad Hailat Artem Komarichev Xue-Wen Chen
Department of Computer Science
Wayne State University
Detroit, MI 48201
{zmhailat, fn9241, xwchen}@wayne.edu

***Abstract*—Convolutional neural networks (CNNs) attain state-of-the-art performance on various classification tasks assuming a sufficiently large number of labeled training examples. Unfortunately, curating sufficiently large labeled training dataset requires human involvement, which is expensive and time consuming. Semi-supervised methods can alleviate this problem by utilizing a limited number of labeled data in conjunction with sufficiently large unlabeled data to construct a classification model. Self-training techniques are among the earliest semi-supervised methods proposed to enhance learning by utilizing unlabeled data. In this paper, we propose a deep semi-supervised learning (DSSL) self-training method that utilizes the strengths of both supervised and unsupervised learning within a single model. We measure the efficacy of the proposed method on semi-supervised visual object classification tasks using the datasets CIFAR-10, CIFAR-100, STL-10, MNIST, and SVHN. The experiments show that DSSL surpasses semi-supervised state-of-the-art methods for most of the aforementioned datasets.**

I. Introduction

Deep learning methods are among the best methods addressing various machine learning problems including object recognition, classification, and image segmentation [1]. A key to the success of supervised deep learning methods is the availability of a sufficiently large labeled training data [2]. Unfortunately, creating a sufficiently large labeled training data with enough examples for each class (e.g., ImageNet [3]) is not an easy task. This task is time consuming, expensive, susceptible to noise and mislabeled [4] examples, and requires experienced human effort. On the other hand, unlabeled data is easily available and inexpensive to collect. For example, a dataset of unlabeled images can be collected from online publicly available resources such as webpages and videos. Therefore, there has been a recognizable advance in exploiting the available large unlabeled data alongside with the limited labeled data to enhance the performance of deep learning methods.

Generally speaking, supervised deep learning methods (e.g., CNNs) learn class-specific sets of features whereas unsupervised deep learning methods (e.g., autoencoders (AEs)) learn general detailed features [5]. Given the relative strengths of each approach, methods to combine both approaches have been forthcoming. Recently, several approaches were developed to combine supervised and unsupervised learning methods into a single model. [6] proposed the use of unsupervised learning as a pre-training step for a supervised model. In other approaches [7]–[9], supervised and unsupervised learning methods were combined into a single model and trained simultaneously.

Self-training [10] is a semi-supervised learning (SSL) method that utilizes the access to large number of unlabeled examples and limited number of labeled data. Self-training has been proven to be promising in many machine learning applications [11]. However, it showed a few shortcomings: e.g., (1) its performance may fluctuate due to the fact that some unlabeled instances will remain unlabeled during the training phase; (2) its accuracy may be deteriorated because erroneous predictions will lead to adding mislabeled instances; this is especially true when the size of labeled data is small. Furthermore, the performance of self-training methods also depends on other factors such as the characteristic of data and confidence measures for adding instances during training: an erroneous confidence measure often leads to mislabeled instances being added to the labeled set.

The primary contribution of this paper is a semi-supervised learning method using the self-training ideas. The architecture of the proposed method combines the strengths of both supervised and unsupervised neural networks into a single semi-supervised deep learning method which we refer as DSSL. DSSL exploits the availability of large-scale unlabeled data in conjunction with limited labeled data to learn both generic features (unsupervised) and discriminative features (supervised) in a self-training fashion. To the best of our knowledge, the proposed method trained in the proposed procedure is among the first of its kind. The novelty of the proposed method is a generalization of self-training, but differs standard self-training in three-fold: (1) The network architecture branches into two parallel tracks after the input goes through a series of deep convolutional blocks. One track is trained for the classification and the other is trained in an autoencoder fashion. The gradient from each track is aggregated and back-propagated to the initial deep convolutional unit. (2) The training algorithm uses a self-training to assign labels to all of the unlabeled samples and employs them later in the next epoch in the supervised training of the model. (3) The proposed method utilizes every example, both labeled and unlabeled, with both supervised and unsupervised methods.

Problem Formulation. The goal in a classical supervised learning framework is to learn a decision model $\mathcal{M}$ from n available training examples. The training examples are denoted by $\mathcal{D}_n$, where n is the total number of training examples in $\mathcal{D}$. We denote a training example (e.g., image) by $x_i \in \mathbb{R}^d$,

where $i \in \{1, \ldots, n\}$. Also, we denote the class label of x_i by $y_i \in \{1, \ldots, C\}$, where C is the total number of different classes in $\mathcal{D}_n$. In general, the training dataset is represented by $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^{n}$.

In SSL settings, the decision model is learned when the labels of a limited subset of $\mathcal{D}_n$ are available. The labeled subset of l examples is denoted by $\mathcal{D}_l$, where $\mathcal{D}_l \subset \mathcal{D}_n$. The labels of the remaining u examples are not available. The unlabeled data is denoted by $\mathcal{D}_u$, where $\mathcal{D}_u \subset \mathcal{D}_n$ and $u = n - l$. Consequently, the training data in a SSL framework is denoted by $\mathcal{D}_n = \{\mathcal{D}_l \cup \mathcal{D}_u\}$.

In particular, we employ self-training method, which is one of the common classes of SSL [11]. Self-training methods are iterative algorithms, where a model $\mathcal{M}$ is learned from the available labeled subset $\mathcal{D}_l$ first. Then, the learned model $\mathcal{M}$ is used to predict labels for the unlabeled subset $\mathcal{D}_u$. The predicted labels with confidence scores more than a predefined confidence threshold p are retained and used in future steps. After that, the model $\mathcal{M}$ is retrained on the combined labeled and predicted data. This procedure is repeated until meeting a stop condition.

II. Related work

The applications of self-training method include various machine learning problems such as natural language processing [12] and object detection [13]. One common problem of the self-training methods is that they are prone to error and poor prediction which can delude the model. The common solution to the problem caused by poor prediction is to set a predefined confidence threshold for using the labels with the confident score more than the threshold in each iteration. DSSL is a self-training semi-supervised method that considers standard prediction method. It predicts and uses the labels for all unlabeled data without any thresholds. The DSSL architecture and training procedure overcome the self-training problem of poor prediction. In the remaining part of this section, we focus on semi-supervised deep learning methods closely related to our proposed method.

Sajjadi et al. [14] proposed a transform/stability semi-supervised loss function that relies heavily on a learning model generating different outputs every time it is evaluated. The variations in output stem from different sources of randomization within the model such as dropout and augmentation. In the approach of Sajjadi et al. [14], each minibatch passes n times through heavy augmentation and model evaluation during training. The loss term is calculated as the sum of all pairwise mean squared distances between n outputs. In another study, they also utilized a mutual-exclusivity supervised loss term [15]. The computational cost of evaluating the transform/stability loss function during training increases linearly as a function of the number of evaluations n. Laine et al. [16] presented a temporal self-ensembling semi-supervised method which they refer to as Π-model, which is a special case of the transform/stability semi-supervised loss function of Sajjadi et al. [14] with $n = 2$. DSSL differs from these three methods by employing supervised and unsupervised loss functions in a single model and evaluating the model on each training example only once for each epoch. This helps DSSL achieve superior computational efficiency as the number of evaluations is constant for each epoch.

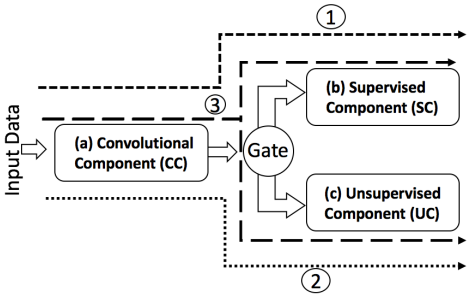

Fig. 1. The DSSL architecture outline.

Rasmus et al. [7] introduced the Γ-model, a subset of the ladder networks [17] that employed an encoder-decoder network architecture to tackle the semi-supervised learning problem. In the Γ-model, all ladder connections excluding the highest one are discarded. The highest connection is then used to construct two parallel, identical branches. One branch takes the original training input whereas the second branch accepts a noisy copy of the original training input. In this method, the unsupervised loss term is calculated as the squared difference between the pre-activation output of both noisy and clean branches. DSSL differs from the Γ-model by using one branch that splits into two supervised and unsupervised branches. In DSSL, a clean input passes through the network only once. The unsupervised loss term is the mean squared difference between the original training input and the corresponding output of the unsupervised branch. Furthermore, DSSL classifies the unlabeled data every other epoch and uses the predicted labels to train the next epoch using both the supervised and unsupervised loss functions. This approach differs from the one proposed by Rasmus et al. [7], which uses only the unlabeled data with the unsupervised loss function.

Tarvainen et al. [18] proposed a technique to improve the semi-supervised learning methods that averages the weights of deep neural networks after each iteration as opposed to each epoch [16] to better predict labels. Here, the consistency loss function was used to estimate the distance between what they called a "student" model and a weight-averaged model called a "teacher" model.

III. Method

DSSL is constructed from three components (Fig. 1). (a) A *Convolutional Component* (CC) consisting of sequence of deep convolutional blocks. (b) A *Supervised Component* (SC) comprising an average pooling layer, a linear layer, and a supervised loss function. We employ a supervised loss function that combines the negative log likelihood of softmax. The supervised loss function is described as:

$$\text{loss}_{\text{CE}}(z_{sc}, y) = -\sum_{j} log\left(\frac{e^{z_{sc_j}[y_j]}}{\sum_i e^{z_{sc_j}[i]}}\right), \qquad (1)$$

where j is the example number in a minibatch b, z_{sc_j} is the output of the SC, which is a vector with a score for each class, y_j is the index of the target class of the example j. (c) An *Unsupervised Component* (UC) comprising a sequence of deconvolutional blocks followed by a sparsity layer, a deconvolutional layer, a `Tanh` layer, and finally an unsupervised loss function. The unsupervised loss function measures the mean squared error between the original input and the output of the UC, which is defined as:

$$\text{loss}_{\text{MSE}}(z_{uc}, b) = \sum_{k} ||z_{uc_k} - b_k||_2^2, \tag{2}$$

where b is an input minibatch to the model, b_k is the k^{th} example (e.g., image) in b, and z_{uc} is the output of the UC.

We construct the CC and UC components from residual neural networks (ResNets), where each component compiles a sequence of residual blocks. A residual block is constructed from two parallel branches or connections. In general, we represent a residual block by:

$$z_r = \mathcal{F}(\mathsf{W}_{\text{shortcut}}, z_{r-1}) + \mathcal{G}(\mathsf{W}_{\text{resid\_branch}}, z_{r-1}), \tag{3}$$

where z_r is the output of a residual block r that is used as an input to the next component, z_{r-1} is the input to the residual block r, $\mathcal{F}$ is the shortcut branch function, $\mathsf{W}_{\text{shortcut}}$ are the shortcut branch parameters (weights), $\mathcal{G}$ is the residual branch function, and $\mathsf{W}_{\text{resid\_branch}}$ are the residual branch parameters (weights).

DSSL includes two different types of residual blocks. A *convolutional residual block (CRB)* uses a set of convolutional layers and a *deconvolutional residual block (DRB)* uses deconvolutional layers. The CC is constructed from a sequence of CRB whereas the UC constructed from a series of DRB. We name the DSSL models following the naming scheme introduced by Zagoruyko et al. [23]. The naming scheme shows the size of the model represented by the number of convolutional layers and the widening factor. For example, DSSL 22-4 represents a model with 22 convolutional layers and a width of 4. The supplementary materials [1] show detailed implementation of the evaluated DSSL models.

DSSL implements a fork with a gate (Fig. 1) that joins CC with UC from one end and with SC from the other end. The gate is assigned one of three modes (i.e., *gate*$\in\{$*on-off*, *off-on*, *on-on*$\}$, *on-off* means SC is on and UC is off) to decide which active components to learn and contribute to the model by directing the output of the convolutional component w to the right components. The gate mode is updated only at the beginning of every epoch and does not change until the beginning of the next epoch.

A DSSL (Algorithm 1) accepts the input dataset $\mathcal{D}$. Moreover, the labels of $\mathcal{D}_l$ never change whereas the labels of $\mathcal{D}_u$ are initially assigned to random values. Training a DSSL model starts the first epoch ($t = 1$) with mode of the *gate=on-off* and $\mathcal{D} = \mathcal{D}_l$ (Fig. 1, path 1). In this epoch, the evaluated model consists of the sequence of CC, SC and the supervised loss function $loss_{\text{CE}}$ (Eq. 1). This step learns CC and SC weights with all available labeled training data $\mathcal{D}_l$. In the second epoch ($t = 2$), the mode of the gate is switched to *gate=off-on* and evaluated on $\mathcal{D} = \{\mathcal{D}_u \cup \mathcal{D}_l\}$ (Fig. 1, path 2). The evaluated model in this epoch includes the sequence of CC, UC and the unsupervised loss function $loss_{\text{MSE}}$ (Eq. 2). This epoch aims to learn UC weights and update CC weights using all available training data regardless of the labels. In the third epoch ($t = 3$), the mode of the gate is switched once more to *gate=on-off* and $\mathcal{D} = \mathcal{D}_l$. Then, the model is evaluated to produce a supervised model $\mathcal{M}_t$. The evaluated model is the sequence of CC, SC and the supervised loss function $loss_{\text{CE}}$ (Eq. 1). After that, DSSL employs self-training by using $\mathcal{M}_t$ to classify the unlabeled dataset $\mathcal{D}_u$. These labels are saved and used accordingly in the next epoch. Next, in the fourth epoch ($t = 4$), the mode of the gate is switched to *gate=on-on* and $\mathcal{D} = \{\mathcal{D}_u \cup \mathcal{D}_l\}$. It should be noted that this $\mathcal{D}$ now includes the new predicted labels from the previous epoch. Then, DSSL is evaluated on $\mathcal{D}$ (Fig. 1, path 3). In this epoch, the evaluated model starts with CC then branches to evaluate the SC and UC simultaneously, where the branches end with the loss functions $loss_{\text{CE}}$ (Eq. 1) and $loss_{\text{MSE}}$ (Eq. 2), respectively. Moreover, the supervised loss function uses the labels of both $\mathcal{D}_l$ and $\mathcal{D}_u$. The model training then alternates between the steps of epochs $t = 3$ and $t = 4$ with gate modes alternating between *on-off* and *on-on* until the maximum number of epochs is reached.

Algorithm 1 The DSSL Algorithm

Require: $\mathcal{D}$ = training, $\mathcal{D}_l$ = labeled, and $\mathcal{D}_u$ = unlabeled datasets.
Require: f_{cc}, f_{sc} and f_{uc} = convolutional, supervised and unsupervised components, respectively.
Require: $loss_{\text{CE}}$ and $loss_{\text{MSE}}$ = cross-entropy (supervised) and mean-squared error (unsupervised) loss functions.
Require: $gate$ = a split that decides which branches contribute to the model in every epoch.

for *epoch_no* $\in [1, max\_no\_epochs]$ **do**
  if *epoch_no is Odd* **then**
    $gate \leftarrow$ *on-off*
    $\mathcal{D} \leftarrow \mathcal{D}_l$
  else if *epoch_no* $== 2$ **then**
    $gate \leftarrow$ *off-on*
    $\mathcal{D} \leftarrow \{\mathcal{D}_l \cup \mathcal{D}_u\}$
  else ▷ The *epoch_no* is even AND > 2
    $gate \leftarrow$ *on-on*
    Classify $\mathcal{D}_u$ *and then update* $\mathcal{D}_u$ *labels*
    $\mathcal{D} \leftarrow \{\mathcal{D}_l \cup \mathcal{D}_u\}$
  end if
  for each minibatch $b \in \mathcal{D}$ **do**
    $w_b \leftarrow f_{cc}(aug(b))$
    if $gate ==$ *on-on* OR $gate ==$ *on-off* **then**
      $z_{\text{sc}} \leftarrow f_{\text{sc}}(w_b)$
      $loss_{\text{CE}}(z_{\text{sc}}, y) \leftarrow -\sum_j log\left(\frac{e^{z_{\text{sc}_j}[y_j]}}{\sum_i e^{z_{\text{sc}_j}[i]}}\right)$
    end if
    if $gate ==$ *on-on* OR $gate ==$ *off-on* **then**
      $z_{uc} \leftarrow f_{\text{uc}}(w_b)$
      $loss_{\text{MSE}}(z_{\text{uc}}, b) \leftarrow \sum_k ||z_{\text{uc}_k} - b_k||_2^2$
    end if
    Update weights using SGD with momentum
  end for
end for

[1] https://www.dropbox.com/sh/xzjoqj3u7pei12z/AAA31ioYApUJ9TyfOgI99yGza?dl=0

TABLE I
THE (MEAN $\pm$ STD.) OF THE CLASSIFICATION ERROR RATES ON CIFAR-10 AND CIFAR-100. WE EVALUATE DSSL 28-10 ONCE.

Method	CIFAR-10			CIFAR-100	
	4,000 labels	8,000 labels	All labels	10,000 labels	All labels
Ensemble GAN [24]	15.59 ± 0.47	14.87 ± 0.89			
TE [16]	12.16 ± 0.31		5.60 ± 0.10	38.65 ± 0.51	26.30 ± 0.15
WAC [18]	12.31 ± 0.28		5.56 ± 0.03		
Sajjadi et al. [14]	11.29 ± 0.24				21.43 ± 0.16
DSSL 22-4	**10.83 $\pm$ 0.4**	**8.73 $\pm$ 0.24**	4.56 ± 0.08	**33.08 $\pm$ 0.23**	22.4 ± 0.24
DSSL 28-10	**9.77**			**30.26**	**18.33**

IV. EXPERIMENTS

In this section, we show the efficiency of DSSL on the benchmark datasets CIFAR-10 [19], CIFAR-100 [19], STL-10 [20], SVHN [21], and MNIST [22]. We train all models in this paper from scratch and do not fine-tune any one of them. Unless otherwise specified, all of our experiments adopt the protocols used by [23]. For every dataset, we report the *mean* and *standard deviation* (mean $\pm$ std.) of the classification error rates. For each dataset, we generate multiple random partitions consisting of different ratios of labeled/unlabeled data. These error rates are calculated by taking the average and standard deviation of error rates of the random partitions. The state-of-the-art results are provided in the tables using a boldface font. Additionally, the preprocessing and setting details for all experiments are available in the supplementary materials.

*1) **CIFAR-10**:* The CIFAR-10 dataset consists of $60,000$ RGB images of size 32×32 pixels. The dataset is divided evenly into 10 different classes. Furthermore, it is available in two predefined parts that we adopt hereafter in our experiments. These parts consist of $50,000$ and $10,000$ examples used for training and testing, respectively. We create five labeled datasets with $4,000$ labeled examples each. We construct each dataset by randomly selecting 400 examples per class from the CIFAR-10 training dataset (i.e., D_l). We retain their actual labels and then treat the rest of the training data as unlabeled (i.e., D_u). We evaluate DSSL 22-4 once on each dataset. Moreover, we repeat the previous steps to construct and evaluate five other labeled datasets on DSSL 22-4, each consisting of $8,000$ labeled examples. Finally, we evaluate DSSL as a supervised method where we use all of the available training dataset as both labeled and unlabeled data and evaluate DSSL 22-4 five times (Table I). We also use one $4,000$ labeled dataset from CIFAR-10 to evaluate a large DSSL 28-10 model for one time. It achieves an error rate of 9.77% and is within 1.06% of the overall performance improvement on the medium DSSL 22-4. However, training larger models require more computational time and memory space.

Furthermore, we study the effects of the self-training (the classification of unlabeled data) on DSSL. For 4,000 labels from CIFAR-10, without self-training, the error rates of DSSL 22-4 is 18%; the DSSL with self-training on the same data achieves an error rate of 11%. As a baseline of WRN 22-4 evaluated on the same dataset, we obtain an error rate of 21% [2]. Therefore, the combination of DSSL architecture and self-training has a joint impact on the performance over the DSSL. Notice that in the proposed method, all unlabeled examples will be predicted with labels and used for self-training in each epoch, which is different from standard self-training methods. The unsupervised branch helps prevent from overfitting the data and reduce the influence of erroneous predictions.

Additionally, we evaluate the effect of dropout and augmentation on DSSL 22-4 on one $4,000$ labeled CIFAR-10 dataset. We run every combination of enabling / disabling dropout and augmentation as an experiment with a dropout of 0.3 (Table 2. supplementary materials). When we disable both dropout and augmentation, DSSL 22-4 achieves an error rate of 24.49%. The error rate decreases to 16.96%, when we enable dropout and disable augmentation. When we disable dropout while keeping augmentation the error rate decreases even further to 11.4%. Finally, DSSL 22-4 achieves the best error rate of 10.83%, when we enable both dropout and augmentation. From these results, we see that the augmentation has more effect than dropout on DSSL, but it benefits from having them both enabled.

*2) **CIFAR-100**:* Similar to CIFAR-10, CIFAR-100 is a collection of $60,000$ images with size 32×32 pixels. However, CIFAR-100 includes 100 classes compared to the 10 classes found in CIFAR-10. This creates the challenge of having a large number of classes with a smaller number of labeled examples per class. In our experiments, we adopt the two publicly available parts of CIFAR-100, namely the $50,000$ and $10,000$ examples for training and testing, respectively. We construct and evaluate DSSL 22-4 for five labeled training datasets. Each dataset is constructed by randomly selecting 100 examples per class (i.e., 20%) while keeping their actual labels. We then assign the rest of the training data to a random set of labels. Finally, we evaluate DSSL 22-4 on all available training examples as labeled and unlabeled data. We also evaluate the large DSSL 28-10 once on labeled training dataset consisting of $10,000$ training examples. The remaining training examples were treated as unlabeled. Table I shows a significant performance improvement when we use a larger model in both cases.

[2]We evaluate WRN 22-4 CNN using the training dataset consisting of $4,000$ labeled examples once.

TABLE II
THE (MEAN $\pm$ STD.) OF THE CLASSIFICATION ERROR RATES ON STL-10.

Method	1,000 labels	All labels
Huang et al. [25]	23.2 $\pm$ 0.3	
CC-GAN [26]	22.21 $\pm$ 0.8	
SCI [27]		18.66 $\pm$ 0.1
DSSL 22-4	**20.1 $\pm$ 0.98**	**10.69 $\pm$ 0.09**

*3) **STL-10**:* The STL-10 dataset includes 10 different classes with 500 and 800 colorful images per class for training and testing, respectively. STL-10 also includes $100,000$ unlabeled images that were selected from labeled examples on ImageNet. These images were extracted from a similar but broader distribution to the training labeled images. All images have the same size of 96×96 pixels. STL-10 is similar in spirit to CIFAR-10. However, STL-10 is more challenging because all STL-10 examples have higher resolution. Furthermore, STL-10 also includes a smaller number of labeled images per class compared to CIFAR-10. Finally, the unlabeled data comes from a wider distribution than the labeled data distribution.

We evaluated DSSL 22-4 on the 10 folds provided with STL10 dataset. Each fold includes 100 labeled training examples for each class. Given the increased image size, DSSL needs more training examples to avoid overfitting. Therefore, we replicate the labeled examples of each fold 10 times. We then randomly select $50,000$ examples from the unlabeled data for D_u. We do not use all available unlabeled data due to resource and time limitations. As a second experiment, we use all available $5,000$ training labeled examples as a D_l. D_u is as defined in the previous experiment. We report the performance of the aforementioned model on the $8,000$ labeled testing examples (Table II).

*4) **MNIST**:* The MNIST dataset is a famous classification benchmark dataset of handwritten digits. It collects $70,000$ grayscale labeled images of size 28×28 pixels. We adopt the suggested parts of $60,000$ and $10,000$ images for training and testing, respectively. We create five different labeled datasets from MNIST training dataset. For each dataset, we randomly select 5% labeled examples from the training dataset while using the rest as unlabeled. We follow the same steps mentioned before to construct five other datasets consisting of 10% labeled examples per class. We evaluate the performance of DSSL 22-4 on the datasets after we remove the `Tanh` layer from the UC. Finally, we evaluate DSSL 22-4 five times on all available training dataset as labeled and unlabeled (Table III).

*5) **SVHN**:* The Street View House Numbers (SVHN) is a digit classification dataset that compiles $73,257$ primary training labeled images, $531,131$ extra labeled training images, and $26,032$ testing images. All images are colorful and are of size 32×32 pixels. The large variations in the images makes the SVHN dataset harder than MNIST to classify. Similar to [14], we use the standard training images. We evaluate DSSL 22-4 after removing `Tanh` layer UC on various percentages of labeled/unlabeled data from SVHN. We construct five different datasets with 1% of labeled training examples. For each dataset, we randomly select 1% of the labeled examples and keep their labels, and then the rest of the training data is unlabeled. We repeat the previous steps with the proportions 5%, 10% and 20% of labeled examples, and create five datasets for each proportion.

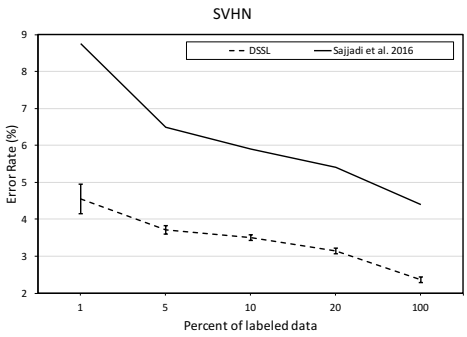

Fig. 2. DSSL 22-4 vs. Sajjadi [14] on various labeled / unlabeled training ratios from the SVHN dataset. DSSL 22-4 results show the (mean $\pm$ std.) of classification error rates.

DSSL suffers from an overfitting problem for the 1% datasets because the number of labeled examples is very small and insufficient for training. Therefore, we replicate every 1% dataset 10 times. The mean of error rates of the 1% labeled dataset with 10 replicas is 4.56% (Fig. 2)

V. DISCUSSION

The proposed DSSL method uses self-training, but alleviates the aforementioned self-training shortcomings by utilizing every training example (both labeled and unlabeled) during training phases. It adds every unlabeled example by predicting their labels without using any confidence thresholds. It also includes an unsupervised branch to fully exploit the information provided through both labeled and unlabeled data. The nature of DSSL training prevents it from falling into the problem of poor prediction. Moreover, the DSSL architecture that includes the supervised and unsupervised components is one more factor that helps avoid this problem.

CIFAR-10 results show that DSSL behaves similar to neural network methods where dropout and augmentation enhance the overall model performance. However, it does not rely on them to run. Despite the fact that we use fairly medium sized models (e.g., DSSL 22-4) due to time and resources limits,

TABLE III
THE (MEAN $\pm$ STD.) OF THE CLASSIFICATION ERROR RATES ON MNIST.
‡ STATE-OF-THE-ART SUPERVISED METHODS.

Method	5% labeled	10% labeled	All labels
Conv-CatGAN [28]			0.48
Sajjadi et al. [14]			0.27 $\pm$ 0.02
DropConnect [29]‡			0.21
MCDNN [30]‡			0.23
Our DSSL 22-4	0.41$\pm$0.06	0.39$\pm$ 0.02	0.24 $\pm$ 0.01

we are able to set a new state-of-the-art record for most of the semi-supervised tasks demonstrated above. Moreover, the performance of DSSL can be improved even further by using larger models such as DSSL 40-10.

DSSL adds a new set of parameters compared to the original corresponding WRN models. For example, the DSSL models 22-4 and 28-10 add 19% and 14.8% of the overall parameters to the original WRN 22-4 and 28-10, respectively. The DSSL 22-4 and 28-10 models use $5.3M$ and $42.8M$ parameters, respectively. The overall time complexity of a DSSL model is similar to the corresponding ResNet model complexity except that the former runs a classifier every other epoch to classify the unlabeled data. For example, we train DSSL 22-4 on CIFAR-10 with $4K$ labeled examples for a total of 320 epochs. We classify the $46K$ unlabeled images 159 times. The average time of classifying all of the $46K$ images with a minibatch size of 100 is 30 seconds. Moreover, the time and number of basic operations that DSSL takes to train an epoch is not the same for all epochs. DSSL turns off the unsupervised branch in half of the epochs and evaluates only the labeled data. For example, evaluating DSSL on the $4K$ CIFAR-10 labeled dataset is very fast. In the second half of the epochs, only then DSSL runs on all available training dataset.

VI. Conclusion

In this paper, we presented DSSL, a semi-supervised classification method that utilizes both supervised and unsupervised neural networks. DSSL uses a limited number of labeled training examples in conjunction with sufficiently large unlabeled examples to create a classification model. The combination of DSSL architecture and self-training has a joint impact on the performance over the DSSL. We measured the performance of DSSL method on five benchmark datasets with various labeled / unlabeled ratios of training examples and then compared our results with state-of-the-art methods. The experiments show that DSSL sets a new state-of-the-art record for CIFAR-10 with 4K and 8K labeled training examples, CIFAR-100 with 10K labeled training examples and STL-10 with 1K labeled training examples with mean of error rates of 9.77%, 8.73%, 33.08% and 20.1%, respectively. Furthermore, DSSL attains the state-of-the-art performance on the SVHN dataset using various ratios of labeled / unlabeled training examples. On the MNIST dataset, DSSL is competitive with other state-of-the-art methods.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[2] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *IEEE 16th ICDM*. IEEE, 2016, pp. 967–972.

[5] M. S. Aslan, Z. Hailat, T. K. Alafif, and X.-W. Chen, "Multi-channel multi-model feature learning for face recognition," *Pattern Recognition Letters*, vol. 85, pp. 79–83, 2017.

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[7] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *NIPS*, 2015, pp. 3546–3554.

[8] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proceedings of the 25th ICML*. ACM, 2008, pp. 792–799.

[9] I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio, "Multi-prediction deep boltzmann machines," in *NIPS*, 2013, pp. 548–556.

[10] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the 9th International CIKM*. ACM, 2000, pp. 86–93.

[11] X. Zhu, "Semi-supervised learning literature survey," vol. 10. Computer Sciences, University of Wisconsin-Madison, 2005, p. 10.

[12] D. McClosky, E. Charniak, and M. Johnson, "Reranking and self-training for parser adaptation," in *COLING ACL 2006*, 2006, p. 337.

[13] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *2005 7th IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1*.

[14] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *NIPS*, 2016, pp. 1163–1171.

[15] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Mutual exclusivity loss for semi-supervised deep learning," in *2016 IEEE International Conference on ICIP*. IEEE, 2016, pp. 1908–1912.

[16] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[17] H. Valpola, "From neural PCA to deep unsupervised learning," *Advances in Independent Component Analysis and Learning Machines*, pp. 143–171, 2015.

[18] A. Tarvainen and H. Valpola, "Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images." Technical report, University of Toronto, 2009.

[20] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the 14th AISTATS*, 2011, pp. 215–223.

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2, 2011, p. 5.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016, pp. 2226–2234.

[25] C. Huang, C. Change Loy, and X. Tang, "Unsupervised learning of discriminative attributes and visual representations," in *Proceedings of the IEEE Conference on CVPR*, 2016, pp. 5175–5184.

[26] E. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks," *arXiv preprint arXiv:1611.06430*, 2016.

[27] E. Hoffer, I. Hubara, and N. Ailon, "Deep unsupervised learning through spatial contrasting," *arXiv preprint arXiv:1610.00243*, 2016.

[28] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

[29] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th ICML-13*, 2013, pp. 1058–1066.

[30] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE CVPR*. IEEE, 2012, pp. 3642–3649.