Large Margin Deep Neural Networks: 
Theory and Algorithms

Shizhao Sun  
College of Computer and Control Engineering  
Nankai University  
Tianjin, 300071, P.R.China  
sunshizhao@mail.nankai.edu.cn

Wei Chen  
Microsoft Research  
Beijing, 100080, P.R.China  
wche@microsoft.com

Liwei Wang  
School of Electronics Engineering  
and Computer Science  
Peking University  
Beijing, 100871, P.R.China  
wanglw@cis.pku.edu.cn

Tie-Yan Liu  
Microsoft Research  
Beijing, 100080, P.R.China  
tie-yan.liu@microsoft.com

Abstract

Deep neural networks (DNN) have achieved huge practical success in recent years. However, its theoretical properties (in particular generalization ability) are not yet very clear, since existing error bounds for neural networks cannot be directly used to explain the statistical behaviors of practically adopted DNN models (which are multi-class in their nature and may contain convolutional layers). To tackle the challenge, we derive a new margin bound for DNN in this paper, in which the expected 0-1 error of a DNN model is upper bounded by its empirical margin error plus a Rademacher Average based capacity term. This new bound is very general and is consistent with the empirical behaviors of DNN models observed in our experiments. According to the new bound, minimizing the empirical margin error can effectively improve the test performance of DNN. We therefore propose large margin DNN algorithms, which impose margin penalty terms to the cross entropy loss of DNN, so as to reduce the margin error during the training process. Experimental results show that the proposed algorithms can achieve significantly smaller empirical margin errors, as well as better test performances than the standard DNN algorithm.

1 Introduction

Deep neural networks (DNN) have achieved great practical success in many machine learning tasks, such as speech recognition, image classification, and natural language processing [7, 10, 11, 18, 25].

To understand why DNN works well and to further improve its performance, extensive researches have been done regarding its theoretical properties, in particular, the generalization ability. For example, in [3, 9, 15], error bounds for neural networks were derived based on Vapnik-Chervonenkis (VC) dimension. In [2, 16], a margin bound was given to fully connected neural networks in the setting of binary classification. While these works shed some lights on the theoretical properties of

1There are also some other works that study the approximation ability of neural networks. For example, in [13, 22] it is shown that neural networks with at least one hidden layer are universal approximators of any continuous function, and in [5, 6, 21], it is demonstrated that deeper neural networks can compute more complex functions than shallower neural networks.
DNN, they are far from sufficient in helping us deeply understand and improve DNN models, due
to the following reasons. First, the number of parameters in many practical DNN models could be
very large, sometimes even larger than the size of training data. This makes the VC dimension based
generalization bound too loose to use. Second, practically DNN is usually used to perform multi-
class classifications, such as the tasks of ImageNet and CIFAR-10, however, most existing bounds
for neural networks are regarding binary classification only. Third, in many real tasks, convolutional
neural networks (CNN) \cite{8} are widely used and proven to be very effective \cite{7, 18, 24}, however,
most existing bounds are derived for fully connected neural networks only.

To tackle the aforementioned challenge, in this paper, we derive a new margin bound for DNN. In
this bound, the expected 0-1 error is upper bounded by the empirical margin error at arbitrary margin
coefficient plus a Rademacher Average (RA) based capacity term. Different from previous bounds,
our proposed margin bound works for both the binary and multi-class settings, and can cover both
fully connected and convolutional neural networks. We have conducted experiments to validate the
reasonability of the bound and found that the bound is consistent with the real behaviors of DNN
models in practical learning tasks.

Our margin bound suggests that the test performance of DNN can be improved by minimizing the
empirical margin error at the optimal margin coefficient. The minimization of commonly used
loss function (i.e., the cross entropy loss) in DNN, however, cannot achieve this goal since it is
the surrogate of the 0-1 error (equivalent to a trivial margin error at zero coefficient), but not the
non-trivial margin error at non-zero coefficient. Specifically, given a data sample, the cross entropy
loss focuses on the minimization of the model output for the true category, while the non-trivial
margin error is concerned with whether the gap between the model output for the true category
and the maximum output for wrong categories is larger than the given margin coefficient. With
this difference in mind, we propose adding a margin penalty term to the cross entropy loss in order
to enlarge the margins on the data samples during the training process. To be specific, given a
data sample, the margin penalty term punishes the small gap between the model output for the true
category and the output for any wrong category. We then minimize the penalized loss function
using back propagation. For ease of reference, we call such an algorithm large margin DNN. To the
best of our knowledge, this is the first work that explicitly minimizes the empirical margin errors
during DNN training.\footnote{One related work is \cite{20}, which combines the generative deep learning methods (e.g., RBM) with a margin-
max posterior. In contrast, our approach aims to enlarge the margin of discriminative deep learning methods
like DNN.}

We have conducted experiments on two benchmark datasets (MNIST and CFAR-10) to test the
performance of the proposed large margin DNN. The experimental results show that large margin
DNN can achieve significantly better test performance than standard DNN. In addition, the models
trained using large-margin DNN have smaller margin error at most margin coefficients, and thus
their performance gains can be well explained by our derived margin bound.

The remaining part of this paper is organized as follows. In Section\textsuperscript{2} we give the notations used
throughout the paper. In Section\textsuperscript{3} we give the margin bound, its theoretical proof, and empirical
validation. In Section\textsuperscript{4} we propose two large margin DNN algorithms and conduct experiments to
test their performances. In Section\textsuperscript{5} we conclude the paper and discuss some future works.

\section{Preliminaries}

Given a multi-class classification problem, we denote $\mathcal{X} = \mathbb{R}^d$ as the input space, $\mathcal{Y} = \{1, \cdots, K\}$
as the output space, and $P$ as the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Here $d$ denotes the dimension of
the input space, and $K$ denotes the number of categories in the output space. We have a training set
$S = \{(x^1, y^1), \cdots, (x^m, y^m)\}$, which is i.i.d. sampled from $\mathcal{X} \times \mathcal{Y}$ according to distribution $P$.
The goal is to learn a prediction model $f \in \mathcal{F} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ from the training set, which produces
an output vector $(f(x, k); k \in \mathcal{Y})$ for each instance $x \in \mathcal{X}$ indicating its likelihood of belonging to
category $k$. Then the final classification is determined by $\arg \max_{k \in \mathcal{Y}} f(x,k)$. The classification
accuracy of prediction model $f$ is measured by its expected 0-1 error, i.e.,

\begin{equation}
err_P(f) = \Pr_{(x,y) \sim P} \| \arg \max_{k \in \mathcal{Y}} f(x,k) \neq y \| = \Pr_{(x,y) \sim P} [\rho(f(x,y)) < 0],
\end{equation}

\textsuperscript{2}One related work is \cite{20}, which combines the generative deep learning methods (e.g., RBM) with a margin-
max posterior. In contrast, our approach aims to enlarge the margin of discriminative deep learning methods
like DNN.
where $I_{[\cdot]}$ is the indicator function and $\rho(f; x, y) = f(x, y) - \max_{k \neq y} f(x, k)$ is the margin of model $f$ at sample $(x, y)$.

We call the 0-1 error on the training set training error and that on the test set test error. Since the expected 0-1 error cannot be obtained due to the unknown distribution $P$, one usually uses the test error as its proxy when examining the classification accuracy.

Now, we consider using neural networks to fulfill the multi-class classification task. Suppose there are $L$ layers in a neural network, including $L - 1$ hidden layers and an output layer. There are $n_l$ nodes in layer $l$. The number of nodes in the output layer is fixed by the classification problem, i.e., $n_L = K$. There are weights associated with the edges between nodes in adjacent layers of the neural network. As a common trick to avoid over fitting, people usually impose a penalty constraint $A$ on the sum of the weights, which is usually implemented by means of weight decay in real training.

Mathematically, we denote the function space of multi-layer neural networks with depth $L$, and weight penalty constraint $A$ as $\mathcal{F}_A$, i.e.,

\begin{equation}
\mathcal{F}_A = \left\{ (x, k) \rightarrow \sum_{i=1}^{n_{l-1}} w_i f_i(x); f_i \in \mathcal{F}_A^{l-1}, \sum_{i=1}^{n_{l-1}} |w_i| \leq A, w_i \in \mathbb{R} \right\};
\end{equation}

for $l = 1, \ldots, L - 1$,

\begin{equation}
\mathcal{F}_A^l = \left\{ x \rightarrow \varphi \left( \phi(f_1(x)), \ldots, \phi(f_{p_l}(x)) \right); f_1, \ldots, f_{p_l} \in \mathcal{F}_A \right\},
\end{equation}

\begin{equation}
\mathcal{F}_A^n = \left\{ x \rightarrow \sum_{i=1}^{n_{l-1}} w_i f_i(x); f_i \in \mathcal{F}_A^{l-1}, \sum_{i=1}^{n_{l-1}} |w_i| \leq A, w_i \in \mathbb{R} \right\};
\end{equation}

and,

\begin{equation}
\mathcal{F}_A^0 = \left\{ x \rightarrow x; i \in \{1, \ldots, d\} \right\};
\end{equation}

where $w_i$ denotes the weight in the neural network, and the functions $\varphi$ and $\phi$ will be explained in the following.

The output of the neural networks is a normalized vector produced by the softmax operation. Please note that the above formulation can cover both fully connected layers and convolutional layers:

(1) If the $l$-th layer is a convolutional layer, the outputs of the $(l - 1)$-th layer are mapped to the $l$-th layer by means of local convolutional filters, activation, and then pooling. That is, in Eqn (3), lots of weights $w_i$ equal 0, and $n_l$ is determined by $n_{l-1}$ as well as the number and domain size of the convolutional filters. In Eqn (3), $p_l$ equals the size of the pooling region in the $l$-th layer, and function $\varphi : \mathbb{R}^p \rightarrow \mathbb{R}$ is called the pooling function. Widely-used pooling functions include the max-pooling $\max(t_1, \ldots, t_p)$ and the average-pooling $(t_1 + \cdots + t_p)/p$. $\phi$ is an increasing function and usually called the activation function. Widely-used activation functions include the standard sigmoid function $\phi(t) = \frac{1}{1 + e^{-t}}$, the tanh function $\phi(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$, and the rectifier function $\phi(t) = \max(0, t)$. Please note that all these activation functions are 1-Lipschitz.

(2) If the $l$-th layer is a fully connected layer, the outputs of the $(l - 1)$-th layer are mapped to the $l$-th layer by global linear combination and subsequently activation. That is, in Eqn (3) $p_l = 1$ and $\varphi(x) = x$.

Given a convolutional layer and a fully connected layer, if the numbers of nodes in their previous layer are the same, the average number of weights (and thus the sum of the weights) associated with each node in a convolutional layer will be much smaller than that for a fully connected layer. As a result, to achieve the same strength of weight decay, the penalty constraint $A$ for the convolutional layers should be much smaller than that for the fully connected layers.

Because distribution $P$ is unknown and the 0-1 error is non-continuous, a common way of learning the weights in the neural network is to minimize the empirical (surrogate) loss function. A widely used loss function is the cross entropy loss, which is defined as follows,

\begin{equation}
C(f; x, y) = - \sum_{k=1}^{K} z_k \ln f(x, k),
\end{equation}

where $z_k = 1$ if $k = y$, and $z_k = 0$ otherwise.

Back-propagation algorithm is usually employed to minimize the loss functions, in which the weights are updated by means of stochastic gradient descent (SGD).
3 Margin Bound for Multi-class Deep Neural Networks

In this section, we present our margin bound for multi-class DNN, followed by its proof and empirical validation.

3.1 Margin Bound

Our bound is based on the empirical margin bound and the Rademacher Average (RA) of DNN. First of all, we give definitions to the multi-class empirical margin error and RA respectively.

Definition 1. Suppose \( f \in F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \) is a multi-class prediction model. For \( \forall \gamma \in (0, 1) \), the empirical margin error of \( f \) at margin coefficient \( \gamma \) is defined as follows:

\[
err_\gamma(f) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[f(x_i, y_i) < \gamma].
\]

Definition 2. \([4]\) Suppose \( F : \mathcal{X} \rightarrow \mathbb{R} \) is a model space with a single dimensional output. The Rademacher average (RA) of \( F \) is defined as follows:

\[
R_m(F) = E_{\varepsilon, \sigma} \left[ \sup_{f \in F} \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(x_i) \right],
\]

where \( X = \{x_1, \cdots, x_m\} \sim P^m_x \), and \( \sigma = \{\sigma_1, \cdots, \sigma_m\} \) are i.i.d. sampled with \( P(\sigma_i = 1) = 1/2, P(\sigma_i = -1) = 1/2 \).

With the above definitions, we can obtain the following theorem, which gives an upper bound to the expected 0-1 error of the DNN model.

Theorem 1. Suppose input space \( \mathcal{X} = [-M, M]^d \), in the neural networks, function \( \phi \) is \( L_\phi \)-Lipschitz, function \( \phi \) is max-pooling or average-pooling and \( p_l \leq p \). For \( \forall \delta > 0 \), with probability at least \( 1 - \delta \), we have, \( \forall f \in F^L \),

\[
er_{p}(f) \leq \inf_{\gamma \in (0, 1)} \left\{ err_\gamma(f) + cM \frac{SK(2K - 1)}{\gamma} \sqrt{\frac{\log \log(2\gamma^{-1})}{m}} + \sqrt{\frac{\log(2\gamma^{-1})}{2m}} \right\},
\]

where \( c \) is a constant.

Proof. In order to prove the theorem, we leverage the following margin bound for general multi-class prediction models (see Theorem 11 in [16]),

\[
er_{p}(f) \leq \inf_{\gamma \in (0, 1)} \left\{ err_\gamma(f) + cM \frac{SK(2K - 1)}{\gamma} R_m(F^L) + \sqrt{\frac{\log \log(2\gamma^{-1})}{m}} + \sqrt{\frac{\log(2\gamma^{-1})}{2m}} \right\},
\]

where \( \gamma \) is at one of the extreme points of the \( 1 \) ball, which implies:

\[
R_m(F^L) \leq A E_{\varepsilon, \sigma} \left[ \sup_{f \in F^L} \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(x_i) \right] = AR_m(F^L). \tag{10}
\]

For function class \( F^L \), if the \((L - 1)\)-th layer is a fully connected layer, it is clear that \( R_m(F^L) \leq R_m(\phi \circ F^L) \) holds. If the \((L - 1)\)-th layer is a convolutional layer with max-pooling or average-pooling, we have,

\[
R_m(F^L_{\text{conv}}) = E_{\varepsilon, \sigma} \left[ \sup_{f_1, \cdots, f_{L-1} \in F^L_{\text{conv}}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i \phi(f_1(x_i), \cdots, \phi(f_{L-1}(x_i))) \right] \leq E_{\varepsilon, \sigma} \left[ \sup_{f_1, \cdots, f_{L-1} \in F^L_{\text{conv}}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i \phi(f_1(x_i)) \right] \leq E_{\varepsilon, \sigma} \left[ \sup_{f_1, \cdots, f_{L-1} \in F^L_{\text{conv}}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i \phi(f_1(x_i)) \right] = p_{L-1} R_m(\phi \circ F^L_{\text{conv}}). \tag{13}
\]
The inequality (12) holds due to the fact that most widely used activation functions $\phi$ (e.g., standard sigmoid and rectifier) have non-negative outputs. Therefore, for both fully connected layers and convolutional layers, $R_m(F_{L-1}^A) \leq p_{L-1} R_m(\phi \circ \bar{F}_{L-1}^A)$ uniformly holds. Further considering the Lipschitz property of $\phi$, we have,

$$R_m(F_{L-1}^A) \leq 2p_{L-1} L \phi R_m(\bar{F}_{L-1}^A).$$

(14)

By iteratively using the maximization principle of inner product in (10) and the property of RA in (14), considering $p_l \leq p$, we can obtain the following inequality,

$$R_m(F_{L-1}^A) \leq (2p L \phi)^{L-1} R_m(\bar{F}_{L}^A).$$

(15)

According to [4], $R_m(\bar{F}_{L}^A)$ can be bounded by:

$$R_m(\bar{F}_{L}^A) \leq cAM \sqrt{\frac{\ln d}{m}},$$

(16)

where $c$ is a constant.

Combining (15) and (16), we can obtain the following upper bound on the RA of DNN,

$$R_m(F_{L}^A) \leq cM \sqrt{\frac{\ln d}{m}} (p L \phi)^L.$$

(17)

By substituting this RA bound into Inequality (9), we can prove the theorem.

As for Theorem[11] we have the following discussions:

(1) When the number of training examples approaches infinity (i.e., $m \to \infty$), the expected 0-1 error will be bounded solely by the empirical margin error, since other terms in the bound vanish at the rate of $m^{-1/2}$. Considering that the empirical margin error is an increasing function of $\gamma$, the theorem tells us that the expected 0-1 error will be bounded by the empirical margin error at $\gamma = 0$ (i.e., the training error). In other words, if we have sufficiently many training examples, the performance of the learned DNN model on the unseen test data will be no worse than that on the training data.

(2) When the training set is finite, several factors will influence the value of the margin bound. When the number of training data (i.e., $m$) increases, or the dimension of the input space (i.e., $d$) and the size of the output space (i.e., $K$) decrease, the margin bound will become smaller. On the other hand, when the weight penalty constraint (i.e., $A$) increases, the RA term will increase and the empirical margin error will decrease since the hypothesis space becomes larger. When the depth of DNN (i.e., $L$) increases, the RA term will increase, however, it is unclear how the empirical margin error will change (since the space of deeper and thinner nets may not contain the space of shallower and wider nets). To understand the influence of network depth, we have conducted some empirical study whose results will be shown in the next subsection.

(3) As compared to previous margin bound for neural networks ([2, 16]), our margin bound can not only cover binary classification and fully connected DNN, but also multi-class classification and convolutional neural networks (CNN). For CNN, due to the sparse local connections, the sum of weights for each node (and thus $A$) will be much smaller than that of a fully-connected DNN. As a result, the generalization of CNN is usually better than that of a fully-connected DNN.

3.2 Empirical Validation

In this section, we conduct experiments to study how the weight penalty constraint $A$ and the network depth $L$ influence the test performance of DNN, and to validate the reasonability of our margin bound.

We conduct experiments on two datasets, MNIST [19] and CIFAR-10 [17]. The MNIST dataset (for handwritten digit classification) consists of $28 \times 28$ black and white images, each containing a digit 0 to 9. There are 60k training examples and 10k test examples in this dataset. The CIFAR-10 dataset (for object recognition) consists of $32 \times 32$ RGB images, each containing an object, e.g., cat, dog, or ship. There are 50k training examples and 10k test examples in this dataset. For each
Second, we investigate the influence of the network depth on the test performance of the DNN models observed in our experiments (see Figure 1(c) and 1(d)). Further considering that deeper networks tend to have larger capacity, the margin bound in Theorem 1 tells us that when the network depth increases, the margin coefficients (see Figure 2(a) and 2(b)).

We change the degree of weight constraint A (through changing the weight decay coefficients α in the algorithm implementation), and report the empirical margin error and test error in Figure 1. As we can see, on both datasets, by decreasing the strength of weight decay (i.e., smaller α), which corresponds to increasing A, the empirical margin error decreases (see Figure 1(a) and 1(b)). By jointly considering this observation and the fact that the RA term increases with A (see the discussions on Theorem 1), we can come to the conclusion that when the weight constraint A increases, the expected 0-1 error will first decrease and then increase. This theoretical result is consistent with the test performance of the DNN models observed in our experiments (see Figure 1(c) and 1(d)).

Second, we investigate the influence of the network depth L. For this purpose, we train fully-connected DNN models with different depths and restricted total number of weights. Following the practices in [8, 23], we set the numbers of weights of the DNN model for MNIST and CIFAR-10 as 0.64M and 5M respectively. For simplicity and also following many previous works [1, 8, 12, 23], we assume that each hidden layer has the same number of nodes in the experiment. The experimental results are shown in Figure 2. From the figures, we can observe that no matter on which dataset, deeper networks have smaller empirical margin errors than shallower networks for most of the margin coefficients (see Figure 2(a) and 2(b)). Further considering that deeper networks tend to have larger capacity, the margin bound in Theorem 1 tells us that when the network depth increases, the expected 0-1 error will first decrease and then increase. This theoretical result is also consistent with the test performances of the DNN models observed in our experiments (see Figure 2(c) and 2(d)). This indicates the reasonability and practical value of our derived margin bound.

\[ S = \frac{1}{2} \left( 1 + \frac{1}{\gamma} \right) \]

3Specifically, for MNIST, the DNN model has one hidden layer with 800 hidden units; for CIFAR-10, the DNN model has two hidden layers with 1190 hidden units in each hidden layer.

4Specifically, for MNIST, the DNN models with depth 2, 3, 4, 5 and 6 respectively have 800, 494, 399, 346 and 311 units in each hidden layer when the total number of weights is 0.64M; for CIFAR-10, the DNN models with depth 2, 3, 4, 5 and 6 respectively have 1650, 1190, 1000, 886, 806 units in each hidden layer when the total number of weights is 5M.
4 Large Margin Deep Neural Networks

Inspired by Theorem 1, we propose refining the existing DNN algorithms by explicitly enlarging the margin (and thus reducing the margin error) during the training process. For ease of reference, we call the new algorithms large margin DNN.

As we know, a widely used loss function for multi-class DNN is the cross entropy loss. However, as mentioned in the introduction, the minimization of the cross entropy loss could not minimize the non-trivial margin error (with non-zero margin coefficient). Specifically, given a data sample, the cross entropy loss focuses on maximizing the model output for the true category, while the non-trivial margin error is concerned with whether the gap between the model output for the true category and the maximum output for the wrong categories is larger than the margin coefficient $\gamma$. In this sense, the non-trivial margin error can guide the training process in a finer granularity that even if the model has already made a correct prediction in terms of the 0-1 error, its parameters can still be optimized by minimizing the non-trivial margin error.

One straightforward way of refining the DNN algorithms is to change its loss function to be the non-trivial margin error. However, as indicated by Theorem 1, what matters is the margin bound at the optimal margin coefficient $\gamma^+$, which is unfortunately unknown in advance. Therefore, we choose to take a different approach. Specifically, we propose to add a weighted margin penalty term to the cross entropy loss, in order to enlarge the non-trivial margin (or equivalently reduce the margin error) during the training process. By tuning the weight of the penalty term, we can manipulate the non-trivial margin error at different margin coefficient.

4.1 Algorithm Description

We propose adding two kinds of margin penalty terms to the original cross entropy loss. The first penalty term is related to margin, and the second one is related to an upper bound of the margin. Specifically, the corresponding new loss functions are defined as follows (for ease of reference, we call them $C_1$ and $C_2$ respectively): for model $f$, sample $x, y$,

$$C_1(f; x, y) = C(f; x, y) + \lambda \left(1 - \rho(f; x, y)\right)^2,$$

$$C_2(f; x, y) = C(f; x, y) + \frac{\lambda}{K-1} \sum_{k \neq y} \left(1 - (f(x, y) - f(x,k))\right)^2.$$

We call the algorithms that minimize the above new loss functions large margin DNN algorithms (LMDNN). For ease of reference, we denote the large margin DNN minimizing $C_1$ and $C_2$ as LMDNN-$C_1$ and LMDNN-$C_2$ respectively, and standard DNN algorithms minimizing $C$ as DNN. To train the LMDNN, we also employ the back propagation method.

4.2 Experimental Results

We compare the performances of LMDNNs with DNN on both MNIST and CIFAR-10 datasets. We use the well-tuned neural network structures as given in the Caffe tutorial (i.e., LeNet for MNIST and AlexNet for CIFAR-10), and adopt the same training and fine-tune process for all the algorithms under investigation.

As for data pre-processing, we scale the pixel values in MNIST to $[0, 1]$, and subtract the per-pixel mean computed over the training set from each image in CIFAR-10. On both datasets, we do not use data augmentation for simplicity.

For the training process, the weights are initialized randomly and updated by mini-batch SGD. We use the model in the last iteration as our final model. For MNIST, we set the batch size as 64, the momentum as 0.9, and the weight decay coefficient as 0.0005. Each neural network is trained for 10k iterations and the learning rate in each iteration $T$ decreases by multiplying the initial learning rate with a factor of $(1 + 0.0001T)^{-0.75}$. For CIFAR-10, we set the batch size as 100, the momentum as 0.9, and the weight decay coefficient as 0.004. Each neural network is trained for 70k iterations. The learning rate is set to be $10^{-3}$ for the first 60k iterations, $10^{-4}$ for the next 5k iterations, and

\[\text{http://caffe.berkeleyvision.org/gathered/examples/mnist.html}\]
\[\text{http://caffe.berkeleyvision.org/gathered/examples/cifar10.html}\]
10^{-5} for the other 5k iterations. Each model is trained for 10 times (with different initializations), and we report the mean and standard deviation of the test error over the 10 learned models.

Table 1 shows the mean test performance of DNN, and the best mean test performances of LMDNNs by tuning margin penalty coefficient $\lambda$. We can observe that, on both MNIST and CIFAR-10, LMDNNs achieve significant performance gains. In particular, LMDNN-C1 can reduce the test error from 0.899% to 0.734% on MNIST and from 18.399% to 17.598% on CIFAR-10; LMDNN-C2 can reduce the test error from 0.899% to 0.736% on MNIST and from 18.399% to 17.728% on CIFAR-10.

To further understand the effect of adding the margin penalty terms, we plot the empirical margin errors of both DNN and LMDNNs in Figure 3. We can see that by introducing the margin penalty terms, LMDNNs indeed achieve smaller empirical margin errors (i.e., larger margin) than DNN. Furthermore, the model with smaller empirical margin errors really has better test performances. For example, LMDNN-C1 corresponds to both smaller empirical margin error and better test performance than LMDNN-C2. This is consistent with Theorem 1, and in return indicates the reasonability of the theorem.

We also report the mean test error of LMDNNs with different margin penalty coefficient $\lambda$ (see Figure 4). In the figure, we use dashed line to represent the mean test error of DNN (corresponding to $\lambda = 0$). From the figure, we can see that on both MNIST and CIFAR-10, (1) there is a range of $\lambda$ where LMDNNs outperform the baseline (i.e., DNN); (2) although the best test performance of LMDNN-C2 is not as good as that of LMDNN-C1, the former has a broader range of $\lambda$ that can outperform the baseline in terms of test error. This indicates the value of using LMDNN-C2: it eases the tuning of hyperparameter $\lambda$ by avoiding the max operator in the loss function; (3) with increasing $\lambda$, the test error of LMDNNs will first decrease, and then increase. As aforementioned, by tuning the penalty coefficient $\lambda$, we can manipulate the margin error at different margin coefficients. This result verifies our theorem: there is an optimal margin coefficient and we should target at minimizing its corresponding empirical margin error.

5 Conclusion and Future Work

In this work, we have derived a novel margin bound for DNN which is very general and can cover the multi-class setting and convolutional neural networks. Based on the theory, we have proposed two large margin DNN algorithms, which achieve significant performance gains over the standard DNN algorithm. In the future, we plan to study how other factors influence the test performance of DNN, such as node allocations across layers, types of connections, regularization tricks, etc. We will also work on the design of effective algorithms that can further boost the performance of DNN.
References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In Advances in Neural Information Processing Systems, pages 2654–2662, 2014.

[2] P. L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Transactions on Information Theory, 44(2):525–536, 1998.

[3] P. L. Bartlett, V. Maiorov, and R. Meir. Almost linear vc-dimension bounds for piecewise polynomial networks. Neural computation, 10(8):2159–2173, 1998.

[4] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. The Journal of Machine Learning Research, 3:463–482, 2003.

[5] Y. Bengio. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1):1–127, 2009.

[6] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. IEEE Transactions on Neural Networks, 2014.

[7] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition, pages 3642–3649, 2012.

[8] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume, volume 15, pages 315–323, 2011.

[9] P. W. Goldberg and M. R. Jerrum. Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. Machine Learning, 18(2-3):131–148, 1995.

[10] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6):82–97, 2012.

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[13] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

[15] M. Karpinski and A. Macintyre. Polynomial bounds for vc dimension of sigmoidal neural networks. In Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, pages 200–208. ACM, 1995.

[16] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. Annals of Statistics, pages 1–50, 2002.

[17] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1097–1105, 2012.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[20] C. Li, J. Zhu, T. Shi, and B. Zhang. Max-margin deep generative models. arXiv preprint arXiv:1504.06787, 2015.

[21] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In Advances in Neural Information Processing Systems, pages 2924–2932, 2014.

[22] F. Scarselli and A. C. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. Neural networks, 11(1):15–37, 1998.

[23] P. Y. Simard, D. Steinraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In 12th International Conference on Document Analysis and Recognition, volume 2, pages 958–958, 2003.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.

[25] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In Neural Networks: Tricks of the Trade, pages 639–655. Springer, 2012.