Large-Margin Softmax Loss for Convolutional Neural Networks

Weiyang Liu[1†] WYLIU@PKU.EDU.CN
Yandong Wen[2†] WEN.YANDONG@MAIL.SCUT.EDU.CN
Zhiding Yu[3⋆] YZHIDING@ANDREW.CMU.EDU
Meng Yang[4⋆] YANG.MENG@SZU.EDU.CN
[1]School of ECE, Peking University [2]School of EIE, South China University of Technology
[3]Dept. of ECE, Carnegie Mellon University [4]College of CS & SE, Shenzhen University

Abstract

Cross-entropy loss together with softmax is arguably one of the most common used supervision components in convolutional neural networks (CNNs). Despite its simplicity, popularity and excellent performance, the component does not explicitly encourage discriminative learning of features. In this paper, we propose a generalized large-margin softmax (L-Softmax) loss which explicitly encourages intra-class compactness and inter-class separability between learned features. Moreover, L-Softmax not only can adjust the desired margin but also can avoid overfitting. We also show that the L-Softmax loss can be optimized by typical stochastic gradient descent. Extensive experiments on four benchmark datasets demonstrate that the deeply-learned features with L-softmax loss become more discriminative, hence significantly boosting the performance on a variety of visual classification and verification tasks.

1. Introduction

Over the past several years, convolutional neural networks (CNNs) have significantly boosted the state-of-the-art performance in many visual classification tasks such as object recognition, (Krizhevsky et al., 2012; Sermanet et al., 2014; He et al., 2015b;a), face verification (Taigman et al., 2014; Sun et al., 2014; 2015) and hand-written digit recognition (Wan et al., 2013). The layered learning architecture, together with convolution and pooling which carefully extract features from local to global, renders the strong visual representation ability of CNNs as well as their current significant positions in large-scale visual recognition tasks.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

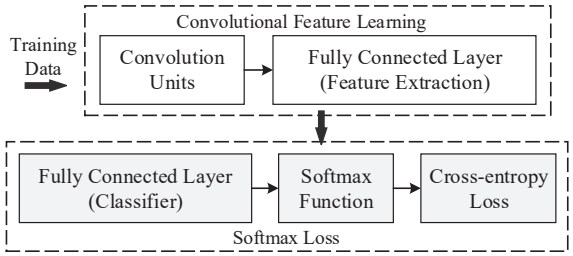

Figure 1. Standard CNNs can be viewed as convolutional feature learning machines that are supervised by the softmax loss.

Facing the increasingly more complex data, CNNs have continuously been improved with deeper structures (Simonyan & Zisserman, 2014; Szegedy et al., 2015), smaller strides (Simonyan & Zisserman, 2014) and new non-linear activations (Goodfellow et al., 2013; Nair & Hinton, 2010; He et al., 2015b). While benefiting from the strong learning ability, CNNs also have to face the crucial issue of overfilling. Considerable effort such as large-scale training data (Russakovsky et al., 2014), dropout (Krizhevsky et al., 2012), data augmentation (Krizhevsky et al., 2012; Szegedy et al., 2015), regularization (Hinton et al., 2012; Srivastava et al., 2014; Wan et al., 2013; Goodfellow et al., 2013) and stochastic pooling (Zeiler & Fergus, 2013) has been put to address the issue.

A recent trend towards learning with even stronger features is to reinforce CNNs with more discriminative information. Intuitively, the learned features are good if their intra-class compactness and inter-class separability are simultaneously maximized. While this may not be easy due to the inherent large intra-class variations in many tasks, the strong representation ability of CNNs make it possible to learn invariant features towards this direction. Inspired by such idea, the contrastive loss (Hadsell et al., 2006) and triplet loss (Schroff et al., 2015) were proposed to enforce extra intra-class compactness and inter-class separability. A con-

[†]Authors contributed equally. [⋆]Authors for corresponding.

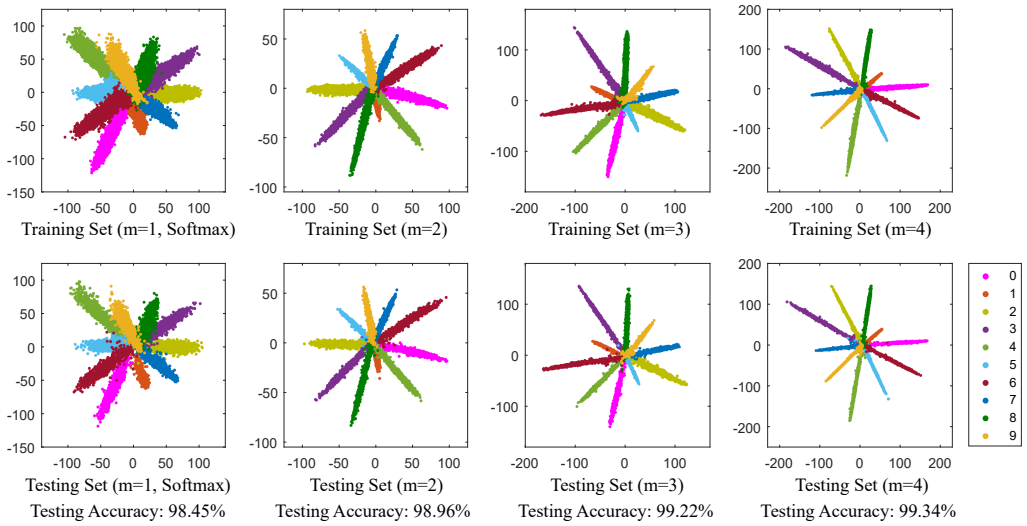

Figure 2. CNN-leanrned features visualization (Softmax Loss (m=1) vs. L-Softmax loss (m=2,3,4)) in MNIST dataset. Specifically, we set the feature (input of the L-Softmax loss) dimension as 2, and then plot them by class. We omit the constant term in the fully connected layer, since it just complicates our analysis and nearly does not affect the performance. Note that, the reason why the testing accuracy is not as good as in Fig.2 is that we only use 2D features to classify the digits here.

sequent problem, however, is that the number of training pairs and triplets can theoretically go up to $\mathcal{O}(N^2)$ where N is the total number of training samples. Considering that CNNs often handle large-scale training sets, a subset of training samples need to be carefully selected for these losses. The softmax function is widely adopted by many CNNs (Krizhevsky et al., 2012; He et al., 2015a;b) due to its simplicity and probabilistic interpretation. Together with the cross-entropy loss, they form arguably one of the most commonly used components in CNN architectures. In this paper, **we define the softmax loss as the combination of a cross-entropy loss, a softmax function and the last fully connected layer** (see Fig. 1). Under such definition, many prevailing CNN models can be viewed as the combination of a convolutional feature learning component and a softmax loss component, as shown in Fig. 1. Despite its popularity, current softmax loss does not explicitly encourage intra-class compactness and inter-class-separability. Our key intuition is that the separability between sample and parameter can be factorized into amplitude ones and angular ones with cosine similarity: $\boldsymbol{W}_c\boldsymbol{x} = \|\boldsymbol{W}_c\|_2\|\boldsymbol{x}\|_2\cos(\theta_c)$, where c is the class index, and the corresponding parameters $\boldsymbol{W}_c$ of the last fully connected layer can be regarded as the linear classifier of class c. Under softmax loss, the label prediction decision rule is largely determined by the angular similarity to each class since softmax loss uses cosine distance as classification score. The purpose of this paper, therefore, is to generalize the softmax loss to a more general large-margin softmax (L-Softmax) loss in terms of angular similarity, leading to potentially larger angular separability between learned features. This is done by incorporating a preset constant m multiplying with the angle between sample and the classifier of ground truth class. m determines the strength of getting closer to the ground truth class, producing an angular margin. One shall see, the conventional softmax loss becomes a special case of the L-Softmax loss under our proposed framework. Our idea is verified by Fig. 2 where the learned features by L-Softmax become much more compact and well separated.

The L-Softmax loss is a flexible learning objective with adjustable inter-class angular margin constraint. It presents a learning task of adjustable difficulty where the difficulty gradually increases as the required margin becomes larger. The L-Softmax loss has several desirable advantages. First, it encourages angular decision margin between classes, generating more discriminative features. Its geometric interpretation is very clear and intuitive, as elaborated in Section 3.2. Second, it partially avoids overfitting by defining a more difficult learning target, casting a different viewpoint to the overfitting problem. Third, L-Softmax benefits not only classification problems, but also verification problems where ideally learned features should have the minimum inter-class distance being greater than the maximum intra-class distance. In this case, learning well separated features can significantly improve the performance.

Our experiments validate that L-Softmax can effectively boost the performance in both classification and verification tasks. More intuitively, the visualizations of the learned features in Fig. 2 and Fig. 5 show great discriminativeness of the L-Softmax loss. As a straightforward generalization of softmax loss, L-Softmax loss not only inherits

all merits from softmax loss but also learns features with large angular margin between different classes. Besides that, the L-Softmax loss is also well motivated with clear geometric interpretation as elaborated in Section 3.3.

2. Related Work and Preliminaries

Current widely used data loss functions in CNNs include Euclidean loss, (square) hinge loss, information gain loss, contrastive loss, triplet loss, Softmax loss, etc. To enhance the intra-class compactness and inter-class separability, (Sun et al., 2014) trains the CNN with the combination of softmax loss and contrastive loss. The contrastive loss inputs the CNNs with pairs of training samples. If the input pair belongs to the same class, the contrastive loss will require their features are as similar as possible. Otherwise, the contrastive loss will require their distance larger than a margin. (Schroff et al., 2015) uses the triplet loss to encourage a distance constraint similar to the contrastive loss. Differently, the triplet loss requires 3 (or a multiple of 3) training samples as input at a time. The triplet loss minimizes the distance between an anchor sample and a positive sample (of the same identity), and maximizes the distance between the anchor sample and a negative sample (of different identity). Both triplet loss and contrastive loss require a carefully designed pair selection procedure. Both (Sun et al., 2014) and (Schroff et al., 2015) suggest that enforcing such a distance constraint that encourages intra-class compactness and inter-class separability can greatly boost the feature discriminativeness, which motivates us to employ a margin constraint in the original softmax loss.

Unlike any previous work, our work cast a novel view on generalizing the original softmax loss. We define the i-th input feature $\boldsymbol{x}_i$ with the label y_i. Then the original softmax loss can be written as

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

where f_j denotes the j-th element ($j \in [1, K]$, K is the number of classes) of the vector of class scores $\boldsymbol{f}$, and N is the number of training data. In the softmax loss, $\boldsymbol{f}$ is usually the activations of a fully connected layer $\boldsymbol{W}$, so f_{y_i} can be written as $f_{y_i} = \boldsymbol{W}_{y_i}^T\boldsymbol{x}_i$ in which $\boldsymbol{W}_{y_i}$ is the y_i-th column of $\boldsymbol{W}$. Note that, we omit the constant b in $f_j, \forall j$ here to simplify analysis, but our L-Softmax loss can still be easily modified to work with b (In fact, the performance is nearly of no difference, so we do not make it complicated here.). Because f_j is the inner product between $\boldsymbol{W}_j$ and $\boldsymbol{x}_i$, it can be also formulated as $f_j = \|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)$ where θ_i ($0 \le \theta_j \le \pi$) is the angle between the vector $\boldsymbol{W}_j$ and $\boldsymbol{x}_i$. Thus the loss becomes

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i})}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right) \tag{2}$$

3. Large-Margin Softmax Loss

3.1. Intuition

We give a simple example to describe our intuition. Consider the binary classification and we have a sample $\boldsymbol{x}$ from class 1. The original softmax is to force $\boldsymbol{W}_1^T\boldsymbol{x} > \boldsymbol{W}_2^T\boldsymbol{x}$ (i.e. $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$) in order to classify $\boldsymbol{x}$ correctly. However, we want to make the classification more rigorous in order to produce a decision margin. So we instead require $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$ ($0 \le \theta_1 \le \frac{\pi}{m}$) where m is a positive integer. Because the following inequality holds:

$$\begin{aligned}\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) &\ge \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1)\\ &> \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2).\end{aligned} \tag{3}$$

Therefore, $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$ has to hold. So the new classification criteria is a stronger requirement to correctly classify $\boldsymbol{x}$, producing a more rigorous decision boundary for class 1.

3.2. Definition

Following the notation in the preliminaries, the L-Softmax loss is defined as

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}}{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right) \tag{4}$$

in which we generally require

$$\psi(\theta) = \begin{cases}\cos(m\theta), & 0 \le \theta \le \frac{\pi}{m}\\ \mathcal{D}(\theta), & \frac{\pi}{m} < \theta \le \pi\end{cases} \tag{5}$$

where m is a integer that is closely related to the classification margin. With larger m, the classification margin becomes larger and the learning objective also becomes harder. Meanwhile, $\mathcal{D}(\theta)$ is required to be a monotonically decreasing function and $\mathcal{D}(\frac{\pi}{m})$ should equal $\cos(\frac{\pi}{m})$.

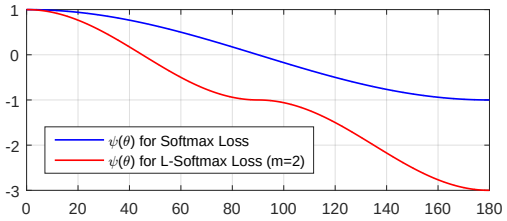

Figure 3. $\psi(\theta)$ for softmax loss and L-Softmax loss.

To simplify the forward and backward propagation, we construct a specific $\psi(\theta_i)$ in this paper:

$$\psi(\theta) = (-1)^k\cos(m\theta) - 2k, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}] \tag{6}$$

where $k \in [0, m-1]$ and k is an integer. Combining Eq. (1), Eq. (4) and Eq. (6), we have the L-Softmax loss that

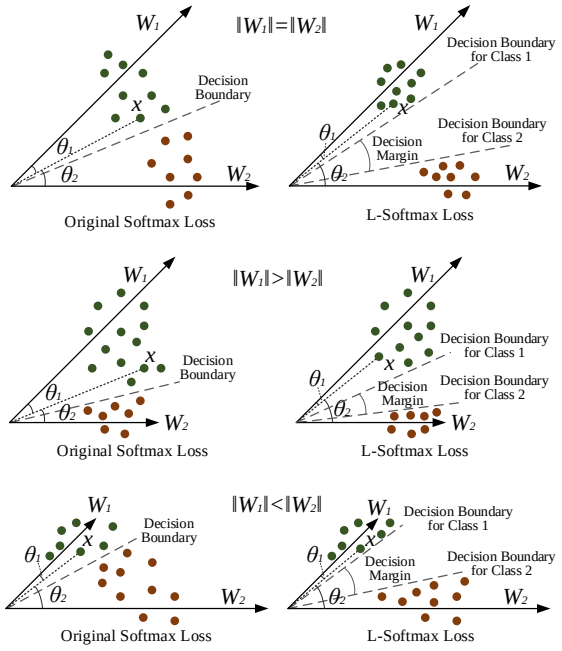

Figure 4. Examples of Geometric Interpretation.

is used throughout the paper. For forward and backward propagation, we need to replace $\cos(\theta_j)$ with $\frac{\boldsymbol{W}_j^T \boldsymbol{x}_i}{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|}$, and replace $\cos(m\theta_{y_i})$ with

$$\begin{aligned}\cos(m\theta_{y_i}) = C_m^0 \cos^m(\theta_{y_i}) - C_m^2 \cos^{m-2}(\theta_{y_i})(1-\cos^2(\theta_{y_i})) \\ + C_m^4 \cos^{m-4}(\theta_{y_i})(1-\cos^2(\theta_{y_i}))^2 + \cdots \\ (-1)^n C_m^{2n} \cos^{m-2n}(\theta_{y_i})(1-\cos^2(\theta_{y_i}))^n + \cdots\end{aligned} \tag{7}$$

where n is an integer and $2n \leq m$. After getting rid of θ, we could perform derivation with respect to $\boldsymbol{x}$ and $\boldsymbol{W}$. It is also trivial to perform derivation with mini-batch input.

3.3. Geometric Interpretation

We aim to encourage aa angle margin between classes via the L-Softmax loss. To simplify the geometric interpretation, we analyze the binary classification case where there are only $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$.

First, we consider the $\|\boldsymbol{W}_1\| = \|\boldsymbol{W}_2\|$ scenario as shown in Fig. 4. With $\|\boldsymbol{W}_1\| = \|\boldsymbol{W}_2\|$, the classification result depends entirely on the angles between $\boldsymbol{x}$ and $\boldsymbol{W}_1(\boldsymbol{W}_2)$. In the training stage, the original softmax loss requires $\theta_1 < \theta_2$ to classify the sample $\boldsymbol{x}$ as class 1, while the L-Softmax loss requires $m\theta_1 < \theta_2$ to make the same decision. We can see the L-Softmax loss is more rigor about the classification criteria, which leads to a classification margin between class 1 and class 2. If we assume both softmax loss and L-Softmax loss are optimized to the same value and all training features can be perfectly classified, then the angle margin between class 1 and class 2 is given by $\frac{m-1}{m+1}\theta_{1,2}$ where $\theta_{1,2}$ is the angle between classifier vector $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. The L-Softmax loss also makes the decision boundaries for class 1 and class 2 different as shown in Fig 4, while originally the decision boundaries are the same. From another viewpoint, we let $\theta_1' = m\theta_1$ and assume that both the original softmax loss and the L-Softmax loss can be optimized to the same value. Then we can know θ_1' in the original softmax loss is $m-1$ times larger than θ_1 in the L-Softmax loss. As a result, the angle between the learned feature and $\boldsymbol{W}_1$ will become smaller. For every class, the same conclusion holds. In essence, the L-Softmax loss narrows the feasible angle[1] for every class and produces an angle margin between these classes.

For both the $\|\boldsymbol{W}_1\| > \|\boldsymbol{W}_2\|$ and $\|\boldsymbol{W}_1\| < \|\boldsymbol{W}_2\|$ scenarios, the geometric interpretation is a bit more complicated. Because the length of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ is different, the feasible angles of class 1 and class 2 are also different (see the decision boundary of original softmax loss in Fig. 4). Normally, the larger $\boldsymbol{W}_j$ is, the larger the feasible angle of its corresponding class is. As a result, the L-Softmax loss also produces different feasible angles for different classes. Similar to the analysis of the $\|\boldsymbol{W}_1\| = \|\boldsymbol{W}_2\|$ scenario, the proposed loss will also generate a decision margin between class 1 and class 2.

3.4. Discussion

The L-Softmax loss utilizes a simple modification over the original softmax loss, achieving a classification angle margin between classes. By assigning different values for m, we define a flexible learning task with adjustable difficulty for CNNs. The L-Softmax loss is endowed with some nice properties such as

- The L-Softmax loss has a clear geometric interpretation. m controls the margin among classes. With bigger m (under the same training loss), the ideal margin between classes becomes larger and the learning difficulty is also increased. With $m = 1$, the L-Softmax loss becomes identical to the original softmax loss.
- The L-Softmax loss defines a relatively difficult learning objective with adjustable margin (difficulty). A difficult learning objective can effectively avoid overfitting and take full advantage of the strong learning ability from deep and wide architectures.
- The L-Softmax loss can be easily used as a drop-in replacement for standard loss, as well as used in tandem with other performance-boosting approaches and

[1] Feasible angle of the i-th class refers to the possible angle between $\boldsymbol{x}$ and $\boldsymbol{W}_i$ that is learned by CNNs.

modules, including learning activation functions, data augmentation, pooling functions or other modified network architectures.

4. Optimization

It is easy to compute the forward and backward propagation for the L-Softmax loss, so it is also trivial to optimize the L-Softmax loss using typical stochastic gradient descent. For L_i, the only difference between the original softmax loss and the L-Softmax loss lies in f_{y_i}. Thus we only need to compute f_{y_i} in forward and backward propagation while $f_j, j \neq y_i$ is the same as the original softmax loss. Putting in Eq. (6) and Eq. (7), f_{y_i} is written as

$$\begin{aligned} f_{y_i} =& (-1)^k \cdot \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\| \cos(m\theta_i) - 2k \cdot \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\| \\ =& (-1)^k \cdot \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\| \Big(C_m^0 \big(\frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|}\big)^m - \\ & C_m^2 \big(\frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|}\big)^{m-2} \big(1 - \big(\frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|}\big)^2\big) + \cdots \Big) \\ & - 2k \cdot \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\| \end{aligned} \tag{8}$$

where $\frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}\|} \in [\cos(\frac{k\pi}{m}), \cos(\frac{(k+1)\pi}{m})]$ and k is an integer that belongs to $[0, m-1]$. For the backward propagation, we use the chain rule to compute the partial derivative: $\frac{\partial L_i}{\partial \boldsymbol{x}_i} = \sum_j \frac{\partial L_i}{\partial f_j} \frac{\partial f_j}{\partial \boldsymbol{x}_i}$ and $\frac{\partial L_i}{\partial \boldsymbol{W}_{y_i}} = \sum_j \frac{\partial L_i}{\partial f_j} \frac{\partial f_j}{\partial \boldsymbol{W}_{y_i}}$. Because $\frac{\partial L_i}{\partial f_j}$ and $\frac{\partial f_j}{\partial \boldsymbol{x}_i}, \frac{\partial f_j}{\partial \boldsymbol{W}_{y_i}}, \forall j \neq y_i$ are the same for both original softmax loss and L-Softmax loss, we leave it out for simplicity. $\frac{\partial f_{y_i}}{\partial \boldsymbol{x}_i}$ and $\frac{\partial f_{y_i}}{\partial \boldsymbol{W}_{y_i}}$ can be computed via

$$\begin{aligned} \frac{\partial f_{y_i}}{\partial \boldsymbol{x}_i} =& (-1)^k \cdot \Big(C_m^0 \frac{m(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-1} \boldsymbol{W}_{y_i}}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-1}} - \\ & C_m^0 \frac{(m-1)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^m \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|^{m-1}\|\boldsymbol{x}_i\|^{m+1}} - C_m^2 \frac{(m-2)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-3} \boldsymbol{W}_{y_i}}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-3}} \\ & + C_m^2 \frac{(m-3)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-2} \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|^{m-3}\|\boldsymbol{x}_i\|^{m-1}} + C_m^2 \frac{m(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-1} \boldsymbol{W}_{y_i}}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-1}} \\ & - C_m^2 \frac{(m-1)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^m \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|^{m-1}\|\boldsymbol{x}_i\|^{m+1}} + \cdots \Big) - 2k \cdot \frac{\|\boldsymbol{W}_{y_i}\| \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}, \end{aligned} \tag{9}$$

$$\begin{aligned} \frac{\partial f_{y_i}}{\partial \boldsymbol{W}_{y_i}} =& (-1)^k \cdot \Big(C_m^0 \frac{m(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-1} \boldsymbol{x}_i}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-1}} - \\ & C_m^0 \frac{(m-1)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^m \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|^{m+1}\|\boldsymbol{x}_i\|^{m-1}} - C_m^2 \frac{(m-2)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-3} \boldsymbol{x}_i}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-3}} \\ & + C_m^2 \frac{(m-3)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-2} \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|^{m-1}\|\boldsymbol{x}_i\|^{m-3}} + C_m^2 \frac{m(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^{m-1} \boldsymbol{x}_i}{(\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|)^{m-1}} \\ & - C_m^2 \frac{(m-1)(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^m \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|^{m+1}\|\boldsymbol{x}_i\|^{m-1}} + \cdots \Big) - 2k \cdot \frac{\|\boldsymbol{x}_i\| \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|}. \end{aligned} \tag{10}$$

In implementation, k can be efficiently computed by constructing a look-up table for $\frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|}$ (i.e. $\cos(\theta_{y_i})$). To be specific, we give an example of the forward and backward propagation when $m = 2$. Thus f_i is written as

$$f_i = (-1)^k \frac{2(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^2}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|} - \big(2k + (-1)^k\big) \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\| \tag{11}$$

where, $k = \begin{cases} 0, & \frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|} \leq \cos(\frac{\pi}{2}) \\ 1, & \frac{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|} > \cos(\frac{\pi}{2}) \end{cases}$.

In backward propagation, $\frac{\partial f_{y_i}}{\partial \boldsymbol{x}_i}$, $\frac{\partial f_{y_i}}{\partial \boldsymbol{W}_{y_i}}$ can be computed with

$$\begin{aligned} \frac{\partial f_{y_i}}{\partial \boldsymbol{x}_i} =& (-1)^k \Big(\frac{4\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|} - \frac{2(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^2 \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|^3} \Big) \\ & - \big(2k + (-1)^k\big) \frac{\|\boldsymbol{W}_{y_i}\| \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}, \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial f_{y_i}}{\partial \boldsymbol{W}_{y_i}} =& (-1)^k \Big(\frac{4\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i \boldsymbol{x}_i}{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|} - \frac{2(\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i)^2 \boldsymbol{W}_{y_i}}{\|\boldsymbol{x}_i\|\|\boldsymbol{W}_{y_i}\|^3} \Big) \\ & - \big(2k + (-1)^k\big) \frac{\|\boldsymbol{x}_i\| \boldsymbol{W}_{y_i}}{\|\boldsymbol{W}_{y_i}\|}. \end{aligned} \tag{13}$$

While $m \geq 3$, we can still use Eq. (8), Eq. (9) and Eq. (10) to compute the formula for forward and backward propagation.

5. Experiments and Results

5.1. Experimental Settings

We evaluate the generalized softmax loss in two typical vision applications: visual classification and face verification. In visual classification, we use three standard benchmark datasets: MNIST (LeCun et al., 1998), CIFAR10 (Krizhevsky, 2009), and CIFAR100 (Krizhevsky, 2009). In face verification, we evaluate our method on the widely used LFW dataset (Huang et al., 2007). We only use a single model in all baseline CNNs to compare our performance. For convenience, we use L-Softmax to denote the L-Softmax loss. Both Softmax and L-Softmax in the experiments use the same CNN shown in Table 1.

General Settings: We follow the design philosophy of VGG-net (Simonyan & Zisserman, 2014) in two aspects: (1) for convolution layers, the kernel size is 3×3 and 1 padding (if not specified) to keep the feature map unchanged. (2) for pooling layers, if the feature map size is halved, the number of filters is doubled in order to preserve the time complexity per layer. Our CNN architectures are described in Table 1. In convolution layers, the stride is set to 1 if not specified. We implement the CNNs using the Caffe library (Jia et al., 2014) with our modifications. For all experiments, we adopt the PReLU (He et al., 2015b) as the activation functions, and the batch size is 256. We use a weight decay of 0.0005 and momentum of 0.9. The weight initialization in (He et al., 2015b)

Layer	MNIST (for Fig. 2)	MNIST	CIFAR10/CIFAR10+	CIFAR100	LFW
Conv0.x	N/A	[3×3, 64]×1	[3×3, 64]×1	[3×3, 96]×1	[3×3, 64]×1, Stride 2
Conv1.x	[5×5, 32]×2, Padding 2	[3×3, 64]×3	[3×3, 64]×4	[3×3, 96]×4	[3×3, 64]×4
Pool1	2×2 Max, Stride 2				
Conv2.x	[5×5, 64]×2, Padding 2	[3×3, 64]×3	[3×3, 96]×4	[3×3, 192]×4	[3×3, 256]×4
Pool2	2×2 Max, Stride 2				
Conv3.x	[5×5, 128]×2, Padding 2	[3×3, 64]×3	[3×3, 128]×4	[3×3, 384]×4	[3×3, 256]×4
Pool3	2×2 Max, Stride 2				
Conv4.x	N/A	N/A	N/A	N/A	[3×3, 256]×4
Fully Connected	2	256	256	512	512

Table 1. Our CNN architectures for different benchmark datasets. Conv1.x, Conv2.x and Conv3.x denote convolution units that may contain multiple convolution layers. E.g., [3×3, 64]×4 denotes 4 cascaded convolution layers with 64 filters of size 3×3.

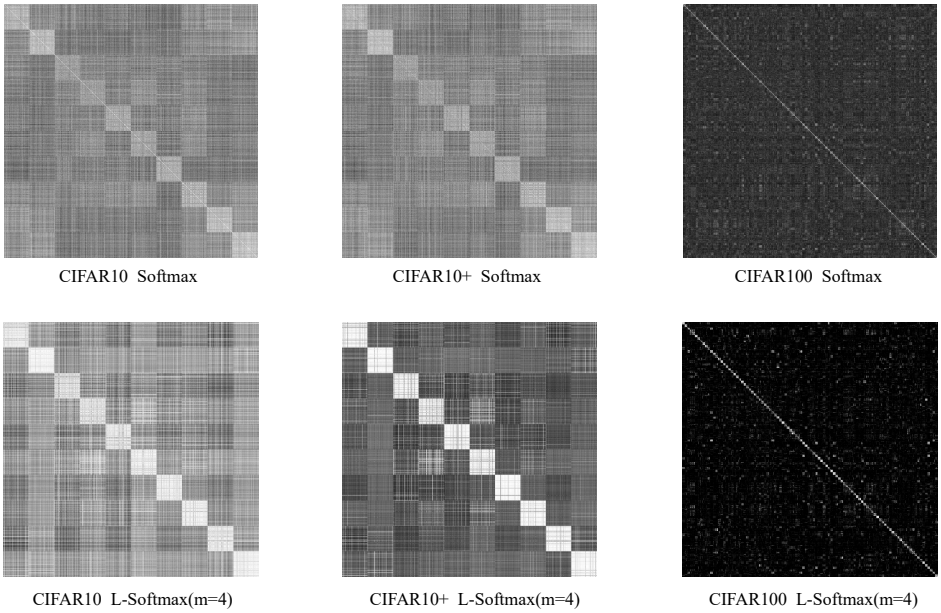

Figure 5. Confusion matrix on CIFAR10, CIFAR10+ and CIFAR100.

and batch normalization (Ioffe & Szegedy, 2015) are used in our networks but without dropout. Note that we only perform the mean substraction preprocessing for training and testing data. For optimization, normally the stochastic gradient descent will work well. However, when training data has too many subjects (such as CASIA-WebFace dataset), the convergence of L-Softmax will be more difficult than softmax loss. For those cases that L-Softmax has difficulty converging, we use a learning strategy by letting $f_{y_i} = \frac{\lambda\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i})+\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}{1+\lambda}$ and start the gradient descent with a very large λ (it is similar to optimize the original softmax). Then we gradually reduce λ during iteration. Ideally λ can be gradually reduced to zero, but in practice, a small value will usually suffice.

MNIST, CIFAR10, CIFAR100: We start with a learning rate of 0.1, divide it by 10 at 12k and 15k iterations, and eventually terminate training at 18k iterations, which is determined on a 45k/5k train/val split.

Face Verification: The learning rate is set to 0.1, 0.01, 0.001 and is switched when the training loss plateaus. The total number of epochs is about is about 30 for our models.

Testing: we use the softmax to classify the testing samples in MNIST, CIFAR10 and CIFAR100 dataset. In LFW dataset, we use the simple cosine distance and the nearest neighbor rule for face verification.

5.2. Visual Classification

MNIST: Our network architecture is shown in Table 1. Table 2 shows the previous best results and those for our proposed L-Softmax loss. From the results, the L-Softmax loss not only outperforms the original softmax loss using the same network but also achieves the state-of-the-art performance compared to the other deep CNN architectures. In Fig. 2, we also visualize the learned features by the L-Softmax loss and compare them to the original softmax

Method	Error Rate
CNN (Jarrett et al., 2009)	0.53
DropConnect (Wan et al., 2013)	0.57
FitNet (Romero et al., 2015)	0.51
NiN (Lin et al., 2014)	0.47
Maxout (Goodfellow et al., 2013)	0.45
DSN (Lee et al., 2015)	0.39
R-CNN (Liang & Hu, 2015)	**0.31**
GenPool (Lee et al., 2016)	**0.31**
Hinge Loss	0.47
Softmax	0.40
L-Softmax (m=2)	0.32
L-Softmax (m=3)	**0.31**
L-Softmax (m=4)	**0.31**

Table 2. Recognition error rate (%) on MNIST dataset.

Method	CIFAR10	CIFAR10+
DropConnect (Wan et al., 2013)	9.41	9.32
FitNet (Romero et al., 2015)	N/A	8.39
NiN + LA units (Lin et al., 2014)	10.47	8.81
Maxout (Goodfellow et al., 2013)	11.68	9.38
DSN (Lee et al., 2015)	9.69	7.97
All-CNN (Springenberg et al., 2015)	9.08	7.25
R-CNN (Liang & Hu, 2015)	8.69	7.09
ResNet (He et al., 2015a)	N/A	6.43
GenPool (Lee et al., 2016)	7.62	6.05
Hinge Loss	9.91	6.96
Softmax	9.05	6.50
L-Softmax (m=2)	7.73	6.01
L-Softmax (m=3)	7.66	5.94
L-Softmax (m=4)	**7.58**	**5.92**

Table 3. Recognition error rate (%) on CIFAR10 dataset. CIFAR10 denotes the performance without data augmentation, while CIFAR10+ is with data augmentation.

loss. Fig. 2 validates the effectiveness of the large margin constraint within L-Softmax loss. With larger m, we indeed obtain a larger angular decision margin.

CIFAR10: We use two commonly used comparison protocols in CIFAR10 dataset. We first compare our L-Softmax loss under no data augmentation setup. For the data augmentation experiment, we follow the standard data augmentation in (Lee et al., 2015) for training: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. In testing, we only evaluate the single view of the original 32×32 image. The results are shown in Table 3. One can observe that our L-Softmax loss greatly boosts the accuracy, achieving 1%-2% improvement over the original softmax loss and the other state-of-the-art CNNs.

CIFAR100: We also evaluate the generalize softmax loss on the CIFAR100 dataset. The CNN architecture refers to Table 1. One can notice that the L-Softmax loss outperform the CNN with softmax loss and all the other competitive methods. The L-Softmax loss improves more than 2.5%

Method	Error Rate
FitNet (Romero et al., 2015)	35.04
NiN (Lin et al., 2014)	35.68
Maxout (Goodfellow et al., 2013)	38.57
DSN (Lee et al., 2015)	34.57
dasNet (Stollenga et al., 2014)	33.78
All-CNN (Springenberg et al., 2015)	33.71
R-CNN (Liang & Hu, 2015)	31.75
GenPool (Lee et al., 2016)	32.37
Hinge Loss	32.90
Softmax	32.74
L-Softmax (m=2)	29.95
L-Softmax (m=3)	29.87
L-Softmax (m=4)	**29.53**

Table 4. Recognition error rate (%) on CIFAR100 dataset.

Method	Outside Data	Accuracy
FaceNet (Schroff et al., 2015)	200M*	**99.65**
Deep FR (Parkhi et al., 2015)	2.6M	98.95
DeepID2+ (Sun et al., 2015)	300K*	98.70
(Yi et al., 2014)	WebFace	97.73
(Ding & Tao, 2015)	WebFace	98.43
Softmax	WebFace	96.53
Softmax + Contrastive	WebFace	97.31
L-Softmax (m=2)	WebFace	97.81
L-Softmax (m=3)	WebFace	98.27
L-Softmax (m=4)	WebFace	**98.71**

Table 5. Verification performance (%) on LFW dataset. * denotes the outside data is private (not publicly available).

accuracy over the CNN and more than 1% over the current state-of-the-art CNN.

Confusion Matrix Visualization: We also give the confusion matrix comparison between the softmax baseline and the L-Softmax loss (m=4) in Fig. 5. Specifically we normalize the learned features and then calculate the cosine distance between these features. From Fig. 5, one can see that the intra-class compactness is greatly enhanced while the inter-class separability is also enlarged.

Error Rate vs. Iteration: Fig. 6 illustrates the relation between the error rate and the iteration number with different m in the L-Softmax loss. We use the same CNN (same as the CIFAR10 network) to optimize the L-Softmax loss with $m = 1, 2, 3, 4$, and then plot their training and testing error rate. One can observe that the original softmax suffers from severe overfitting problem (training loss is very low but testing loss is higher), while the L-Softmax loss can greatly avoid such problem. Fig. 7 shows the relation between the error rate and the iteration number with different number of filters in the L-Softmax loss (m=4). We use four different CNN architecture to optimize the L-Softmax loss with $m = 4$, and then plot their training and testing error rate. These four CNN architectures have the same structure and only differ in the number of filters (e.g. 32/32/64/128

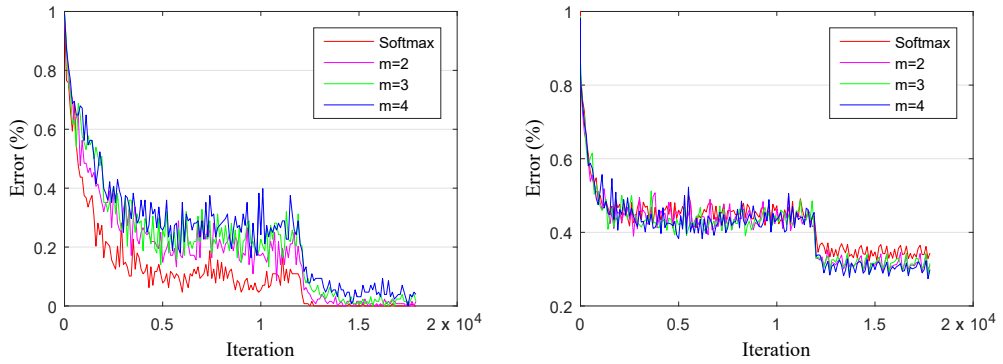

Figure 6. Error vs. iteration with different value of m on CIFAR100. The left shows training error and the right shows testing error.

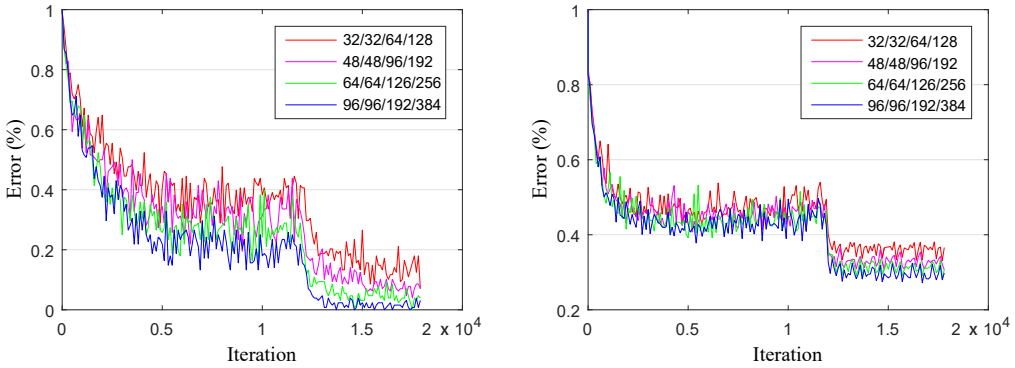

Figure 7. Error vs. iteration (m=4) with different number of filters on CIFAR100. The left (right) presents training (testing) error.

denotes that there are 32, 32, 64 and 128 filters in every convolution layer of Conv0.x, Conv1.x Conv2.x and Conv3.x, respectively). On both the training set and testing set, the L-Softmax loss with larger number of filters performs better than those with smaller number of filters, indicating L-Softmax loss does not easily suffer from overfitting. The results also show that our L-Softmax loss can be optimized easily. Therefore, one can learn that the L-Softmax loss can make full use of the stronger learning ability of CNNs, since stronger learning ability leads to performance gain.

5.3. Face Verification

To further evaluate the learned features, we conduct an experiment on the famous LFW dataset (Huang et al., 2007). The dataset collects 13,233 face images from 5749 persons from uncontrolled conditions. Following the unrestricted with labeled outside data protocol (Huang et al., 2007), we train on the publicly available CASIA-WebFace (Yi et al., 2014) outside dataset (490k labeled face images belonging to over 10,000 individuals) and test on the 6,000 face pairs on LFW. People overlapping between the outside training data and the LFW testing data are excluded. As preprocessing, we use IntraFace (Asthana et al., 2014) to align the face images and then crop them based on 5 points. Then we train a single network for feature extraction, so we only compare the single model performance of current state-of-the-art CNNs. Finally PCA is used to form a compact feature vector. The results are given in Table 5. The generalize softmax loss achieves the current best results while only trained with the CASIA-WebFace outside data, and is also comparable to the current state-of-the-art CNNs with private outside data. Experimental results well validate the conclusion that the L-Softmax loss encourages the intra-class compactness and inter-class separability.

6. Concluding Remarks

We proposed the Large-Margin Softmax loss for the convolutional neural networks. The large-margin softmax loss defines a flexible learning task with adjustable margin. We can set the parameter m to control the margin. With larger m, the decision margin between classes also becomes larger. More appealingly, the Large-Margin Softmax loss has very clear intuition and geometric interpretation. The extensive experimental results on several benchmark datasets show clear advantages over current state-of-the-art CNNs and all the compared baselines.

Acknowledgement

The authors would like to thank Prof. Le Song (Gatech) for constructive suggestions. This work is partially supported by the National Natural Science Foundation for Young Scientists of China (Grant no.61402289) and National Science Foundation of Guangdong Province (Grant no. 2014A030313558).

References

Asthana, Akshay, Zafeiriou, Stefanos, Cheng, Shiyang, and Pantic, Maja. Incremental face alignment in the wild. In *CVPR*, 2014.

Ding, Changxing and Tao, Dacheng. Robust face recognition via a multimodal deep face representation. *IEEE TMM*, 17(11): 2049–2058, 2015.

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Hadsell, Raia, Chopra, Sumit, and LeCun, Yann. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015b.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Huang, Gary B, Ramesh, Manu, Berg, Tamara, and Learned-Miller, Erik. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report, 2007.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jarrett, Kevin, Kavukcuoglu, Koray, Ranzato, Marc'Aurelio, and LeCun, Yann. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick, Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. In *AISTATS*, 2015.

Lee, Chen-Yu, Gallagher, Patrick W, and Tu, Zhuowen. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

Liang, Ming and Hu, Xiaolin. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. In *ICLR*, 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Parkhi, Omkar M, Vedaldi, Andrea, and Zisserman, Andrew. Deep face recognition. In *BMVC*, 2015.

Romero, Adriana, Ballas, Nicolas, Kahou, Samira Ebrahimi, Chassang, Antoine, Gatta, Carlo, and Bengio, Yoshua. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *IJCV*, pp. 1–42, 2014.

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

Sermanet, Pierre, Eigen, David, Zhang, Xiang, Mathieu, Michaël, Fergus, Rob, and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, 2014.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity. In *ICLR*, 2015.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

Stollenga, Marijn F, Masci, Jonathan, Gomez, Faustino, and Schmidhuber, Jürgen. Deep networks with internal selective attention through feedback connections. In *NIPS*, 2014.

Sun, Yi, Chen, Yuheng, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face representation by joint identification-verification. In *NIPS*, 2014.

Sun, Yi, Wang, Xiaogang, and Tang, Xiaoou. Deeply learned face representations are sparse, selective, and robust. In *CVPR*, 2015.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *CVPR*, 2015.

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, and Wolf, Lars. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Cun, Yann L, and Fergus, Rob. Regularization of neural networks using dropconnect. In *ICML*, 2013.

Yi, Dong, Lei, Zhen, Liao, Shengcai, and Li, Stan Z. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

Zeiler, Matthew D and Fergus, Rob. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.