Deep Face Recognition with Center Invariant Loss

Yue Wu[†], Hongfu Liu[†], Jun Li[†], Yun Fu[†‡]

[†]Department of Electrical & Computer Engineering, Northeastern University, Boston, MA, USA
[‡]College of Computer & Information Science, Northeastern University, Boston, MA, USA
{yuewu,yunfu}@ece.neu.edu,liu.hongf@husky.neu.edu,junl.mldl@gmail.com

ABSTRACT

Convolutional Neural Networks (CNNs) have been widely used for face recognition and got extraordinary performance with large number of available face images of different people. However, it is hard to get uniform distributed data for all people. In most face datasets, a large proportion of people have few face images. Only a small number of people appear frequently with more face images. These people with more face images have higher impact on the feature learning than others. The imbalanced distribution leads to the difficulty to train a CNN model for feature representation that is general for each person, instead of mainly for the people with large number of face images. To address this challenge, we proposed a center invariant loss which aligns the center of each person to enforce the learned features to have a general representation for all people. The center invariant loss penalizes the difference between each center of classes. With center invariant loss, we can train a robust CNN that treats each class equally regardless the number of class samples. Extensive experiments demonstrate the effectiveness of the proposed approach. We achieve state-of-the-art results on LFW and YTF datasets.

KEYWORDS

Face Recognition; Convolutional Neural Network; Center Aligned Loss

1 INTRODUCTION

Deep Face Recognition recently has witnessed great success with the development of Convolutional Neural Networks (CNNs). CNNs lead to many great achievements on a series of vision tasks such as object recognition [8, 13, 15, 16, 20, 24, 41], object segmentation [33], image retrieval [12, 18], video analysis [1, 38] and many other applications [3, 4, 11, 21, 31]. As for face recognition, CNNs like DeepFace [29], DeepID series [26–28], FaceNet [25], and many other approaches [17, 22, 34, 37, 39] have been proven to be effective.

To train a robust deep model, abundant training data and well-designed training strategies [13] are necessary. Many approaches focus on collecting and labeling more images to better train CNNs.

ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA
 ISBN 978-1-4503-5416-5/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3126686.3126693

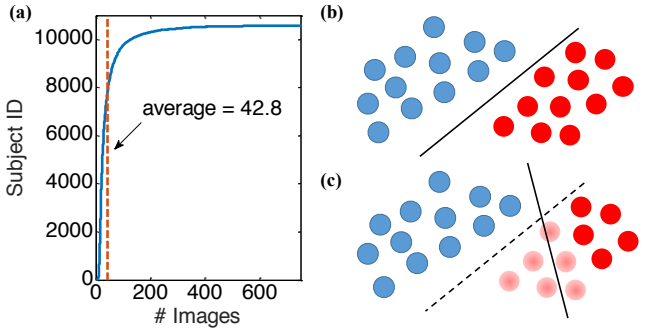

Figure 1: (a) Long-tail distribution of per-subject image numbers in the CASIA-WebFace dataset [40], (b) the classifier learned with balanced data, (c) the classifier learned with imbalanced data.

In [40], CASIA-WebFace was released as the largest publicly available face datatset with 0.5 million images of 10,575 identities at that time, and gained 97.73% accuracy on Labeled Faces in the Wild (LFW) benchmark [9]. Later, MS-Celeb-1M [6] was published with 8.9 million faces of 100 thousand identities. However, in these datasets, long-tail distribution of data [22, 43] are observed. We show the per-subject image number of the CASIA-WebFace dataset in Figure 1(a). We can see that only limited number of classes appear frequently. With imbalanced data for training, the learned classifier in Figure 1(c) has a bias on these classes with more samples compared with the classifier learned with balanced data in Figure 1(b). It is hard to learn features that have a good representation for all classes with the imbalanced distributed data.

To solve this problem, Masi *et al.* [22] utilized domain specific methods for face image synthesis to generate more images. They inflated the size of CASIA-WebFace dataset to several times its size to avoid the long-tail effect. However, the face synthesizing is highly relied on the accurate facial landmark detection, which may introduce poor synthesizing data. And large amount synthesizing samples take longer time to finish the training procedure. In [43], Zhang *et al.* proposed range loss that utilized k greatest range's harmonic mean values in one classes to measure the intra-class variations and add a penalty term to reduce these variations. Models trained on a balanced dataset achieved state-of-the-art performance. However, to ensure there are enough samples for one class to calculate the range loss, the mini-batch need to be carefully constructed, which introduce more labors. Hariharan *et al.* [7] introduced the concept of low-shot visual learning problem and a Squared Gradient Magnitude loss to penalize the difference between classifiers

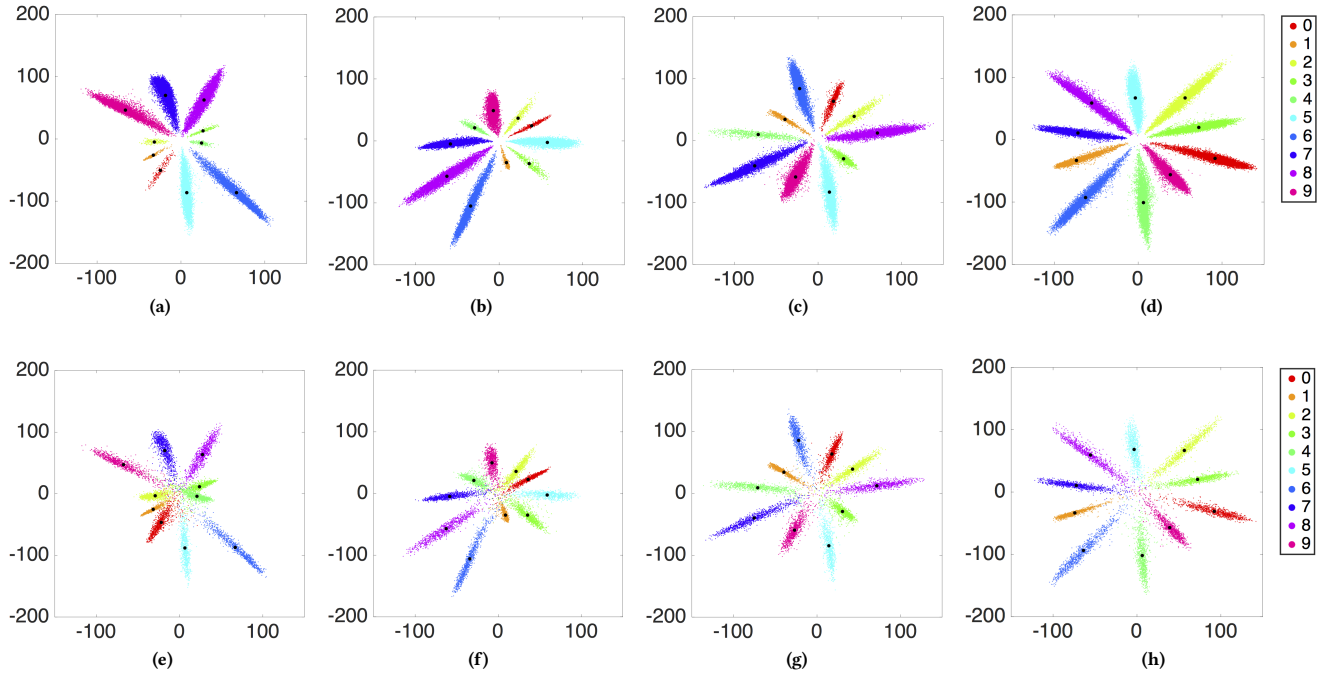

Figure 2: Distributions of deeply learned features in different number of training samples and training/test sets. Five of ten classes have (a) (e) 200 samples, (b) (f) 500 samples, (c) (g) 1000 samples and (d) (h) full samples. (a) (b) (c) (d) are for training sets and (e) (f) (g) (h) are for test sets. Centers of each class are denoted as black dots.

learned on large and small classes. They also hallucinated additional samples by transferring modes of variation from base classes to improve the low-shot learning performance. However, in the problem of feature learning given long-tail distribution data, both small classes and large classes are used to train the model.

Inspired by the idea of penalizing the difference between large and small classes in [5, 7], we propose a center invariant loss which aligns the center of each class to enforce the learned features to have a general representation for all classes. Different from [7] that penalty is added on the classifier weights, the center invariant loss adds constraints on the features. Center invariant loss penalizes the difference between the *L2* norm of each center of classes and the mean *L2* norm of all centers. With joint supervision of multiple losses (softmax loss, center loss [34] and center invariant loss), we can train a robust CNN that treat each class equally regardless the number of class samples. Our contributions are summarized as follows:

- We propose a new loss function, namely center invariant loss to enhance the generalization ability of deeply learned features by penalizing the difference between large classes and small classes.
- We verify that the center invariant loss could help deeply learned features to separate the feature space equally for all classes given extremely imbalanced training data and improve the classification performance.
- Our results show that one single model trained with CASIA-WebFace achieves state-of-the-art performance, 99.12% on LFW [9] and 93.88% on YTF [35] benchmarks.

2 CENTER INVARIANT LOSS

In this section, we first investigate the distributions of deeply learned features with softmax loss on a toy dataset with imbalanced data. Then we present the formulation of the proposed center invariant loss. Last we show how to optimize the center invariant loss with stochastic gradients descent.

2.1 Imbalanced Data Phenomenon

In this subsection, distributions of deeply learned features of a Convolutional Neural Network (CNN) on MNIST [14] are investigated. The CNN model we used is LeNets++ [34] that has 6 convolutional layers and one fully-connected layer. The output number of the fully-connected layer is 2, which means the deeply learned features have 2 dimensions. Thus, the features can be directly plotted on 2-D surface for visualization. The softmax loss function is formalized as:

$$L_s = -\log\left(\frac{e^{\mathbf{w}_y^T \mathbf{x}_i + b_y}}{\sum_{j=1}^{m} e^{\mathbf{w}_j^T \mathbf{x}_i + b_j}}\right), \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is the deeply leaned feature of the i-th sample with dimension d. This sample belongs to the y-th class. $\mathbf{w}_j \in \mathbb{R}^d$ is the j-th weights for the j-th class in the output fully connected layer. $b_j \in \mathbb{R}$ is the corresponding bias term. m is the number of classes.

The MNIST training set has 10 classes with total 60,000 samples. There are about 6,000 samples for each class. The training set is nearly balanced. After training LeNets++ with original MNIST training set, the resulting 2D deep features are plotted in Figure 2d and 2h. Figure 2d shows the distributions of the training set and Figure 2h shows the distributions on MNIST test set with 10,000 samples.

From the two figures, we can observe that the deeply learned features for each class are nearly equitably, unbiased distributed in the feature space. Centers of each class almost form a circle. Both the training set and the test set have the same phenomenon. Then we reduce the training samples in five of ten classes. 200, 500 and 1000 samples are randomly selected from all training samples in the five classes. The other five classes use all training samples they have. The same CNN model and training procedure are utilized. The only difference between these CNN models is the ***number*** of training samples of the five classes. Distributions of these deeply learned features in the training set are shown in Figure 2a, 2b and 2c for 200, 500 and 1000 samples, respectively. From these figures, we can observe that the feature space is dominated by the classes that have more samples. For example, in Figure 2a, although these features seem separable, the feature space is actually not well occupied by all classes. The classes with small number of samples take a small area compared with the other classes. Moreover, we can observe that the centers of five classes with small samples are closer to the origin of the feature space than the other five classes. Bad splitting of the feature space leads to a bad generalization ability of the deeply learned features, shown in Figure 2e, 2f and 2g for the test set.

To improve the generalization ability of the deeply learned features with imbalanced training data, we proposed center invariant loss of which the motivation is to align class centers for a better separation of the feature space.

2.2 Model

We introduce the center invariant loss in this subsection. In m classes, the k-th class ($1 \leq k \leq m$) has n_k samples. The deeply learned feature of the i-th sample is $\mathbf{x}_i$. The k-th class center of the deeply learned features is $\mathbf{c}_k$, which can be written as:

$$\mathbf{c}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_i. \tag{2}$$

Center invariant loss of the i-th sample with label y can be formalized as:

$$L_I = \frac{1}{4}(||\mathbf{c}_y||_2^2 - \frac{1}{m}\sum_{k=1}^{m}||\mathbf{c}_k||_2^2)^2. \tag{3}$$

This formulation penalizes differences between centers of each class. Centers distribute at different positions in the feature space. The center invariant loss aims to make these centers to have the same Euclidean norm. A mean value of all centers Euclidean norm, written as $\frac{1}{m}\sum_{k=1}^{m}||\mathbf{c}_k||_2^2$, is calculated first. Given the i-th sample, the loss generates the cost from the norm of its class center to that mean value. In Figure 3, we give an illustration of the center invariant loss in 2-D surface. We analyze the case with 200 training samples for five of ten classes. The radius of the circle is the mean value of all centers Euclidean norm. We can see that the center invariant loss pull out the centers in side the circle and push in the centers outside the circle.

The center invariant loss is added with softmax loss to jointly supervise the training of the CNN model. softmax loss generates discriminative features and center invariant loss forces a unified separation of feature space for a good generalization ability. The joint supervision of softmax loss and center invariant loss can be

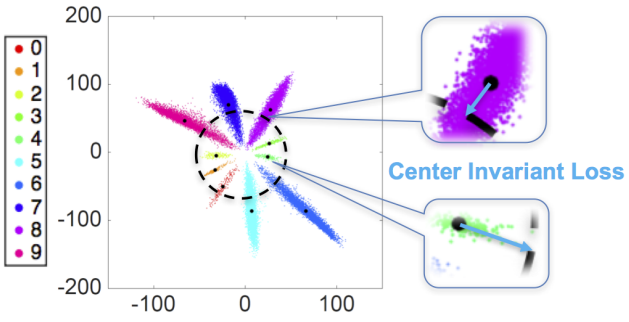

Figure 3: Illustration of center invariant loss. The circle is the mean center Euclidean norm. The center invariant loss pull out the centers inside the circle and push in the centers outside the circle.

written as:

$$\begin{aligned} L &= L_s + \gamma L_I \\ &= -\log(\frac{e^{\mathbf{w}_y^T\mathbf{x}_i+b_y}}{\sum_{j=1}^{m}e^{\mathbf{w}_j^T\mathbf{x}_i+b_j}}) + \frac{\gamma}{4}(||\mathbf{c}_y||_2^2 - \frac{1}{m}\sum_{k=1}^{m}||\mathbf{c}_k||_2^2)^2, \end{aligned} \tag{4}$$

where γ is a scalar to balance the center invariant loss.

2.3 Combination with Center Loss

The softmax loss and center invariant loss focus on the inter-class variations. Here we add one more supervision signal called center loss [34], which aims to minimize the intra-class variations. The loss function can be expressed as:

$$L_c = \frac{1}{2}||\mathbf{x}_i - \mathbf{c}_y||^2. \tag{5}$$

This loss function penalizes the distances between each sample and its class center. Adding Equation (5) to Equation (4), the final loss function becomes:

$$L = L_s + \gamma L_I + \lambda L_c \tag{6}$$

$$\begin{aligned} &= -\log(\frac{e^{\mathbf{w}_y^T\mathbf{x}_i+b_y}}{\sum_{j=1}^{m}e^{\mathbf{w}_j^T\mathbf{x}_i+b_j}}) + \frac{\gamma}{4}(||\mathbf{c}_y||_2^2 - \frac{1}{m}\sum_{k=1}^{m}||\mathbf{c}_k||_2^2)^2 \\ &\quad + \frac{\lambda}{2}||\mathbf{x}_i - \mathbf{c}_y||^2, \end{aligned} \tag{7}$$

where λ is a scalar to balance the center loss. When λ and $\gamma = 0$, this joint loss becomes identical to the original softmax loss.

Discussions: we discuss the influence of each part in the final objective function. The objective function (Equation (6)) consists of three parts: softmax loss, center invariant loss and center loss. softmax loss (L_s) aims to maximize the inter-class variations. Center invariant loss regularizes each class to be treated equally given imbalance data. These two losses both handle inter-class dispersion. Meanwhile, center loss (L_c) tries to minimize the intra-class variations, which is also essential to learn discriminative features.

2.4 Optimization

In this subsection, we calculate the gradients of loss function for backward operation using stochastic gradient descent (SGD). In Equation (6), we have four variables: $\mathbf{w}_j, b_j, \mathbf{x}_i$ and $\mathbf{c}_k$. The optimization for $\mathbf{w}_j$ and b_j is the same with original softmax loss. $\mathbf{c}_k$ is first random initialized and updated based on mini-batch, of which details can be found at [34]. Gradients of multiple losses with respect to $\mathbf{x}_i$ can be written as:

$$\frac{\partial L}{\partial \mathbf{x}_i} = \frac{\partial L_s}{\partial \mathbf{x}_i} + \gamma \frac{\partial L_I}{\partial \mathbf{x}_i} + \lambda \frac{\partial L_c}{\partial \mathbf{x}_i}. \tag{8}$$

The first term $\frac{\partial L_s}{\partial \mathbf{x}_i}$ is the gradients from softmax loss. According to the chain rule, it can be written as:

$$\frac{\partial L_s}{\partial \mathbf{x}_i} = \frac{\partial L_s}{\partial g(\mathbf{x}_i)} \frac{\partial g(\mathbf{x}_i)}{\partial \mathbf{x}_i}, \tag{9}$$

where

$$g(\mathbf{x}_i) = \frac{e^{\mathbf{w}_y^T \mathbf{x}_i + b_y}}{\sum_{j=1}^{m} e^{\mathbf{w}_j^T \mathbf{x}_i + b_j}}. \tag{10}$$

The Equation (9) becomes:

$$\begin{aligned}\frac{\partial L_s}{\partial \mathbf{x}_i} &= \frac{1}{g(\mathbf{x}_i)} \left(\frac{e^{\mathbf{w}_y^T \mathbf{x}_i + b_y} \cdot \mathbf{w}_y}{\sum_{j=1}^{m} e^{\mathbf{w}_j^T \mathbf{x}_i + b_j}} - \frac{e^{\mathbf{w}_y^T \mathbf{x}_i + b_y}}{(\sum_{j=1}^{m} e^{\mathbf{w}_j^T \mathbf{x}_i + b_j})^2} e^{\mathbf{w}_y^T \mathbf{x}_i + b_y} \cdot \mathbf{w}_y \right) \\ &= \frac{1}{g(\mathbf{x}_i)} (g(\mathbf{x}_i)\mathbf{w}_y - (g(\mathbf{x}_i))^2 \mathbf{w}_y) \\ &= (1 - g(\mathbf{x}_i))\mathbf{w}_y. \end{aligned} \tag{11}$$

To simplify the calculation of the second term $\gamma \frac{\partial L_I}{\partial \mathbf{x}_i}$ in Equation (8), we make the notation:

$$\tau = \frac{1}{m} \sum_{k=1}^{m} ||\mathbf{c}_k||_2^2. \tag{12}$$

Taking Equation (12) into Equation (3), we get:

$$L_I = \frac{1}{4} (||\mathbf{c}_y||_2^2 - \tau)^2. \tag{13}$$

According to the chain rule, $\gamma \frac{\partial L_I}{\partial \mathbf{x}_i}$ in Equation (8) can be computed as:

$$\gamma \frac{\partial L_I}{\partial \mathbf{x}_i} = \gamma \frac{\partial L_I}{\partial \mathbf{c}_y} \frac{\partial \mathbf{c}_y}{\partial \mathbf{x}_i}. \tag{14}$$

$\frac{\partial L_I}{\partial \mathbf{c}_y}$ can be calculated as:

$$\frac{\partial L_I}{\partial \mathbf{c}_y} = \frac{1}{2} (||\mathbf{c}_y||_2^2 - \tau)(2\mathbf{c}_y - \frac{2}{m}\mathbf{c}_y) = (||\mathbf{c}_y||_2^2 - \tau)(1 - \frac{1}{m})\mathbf{c}_y. \tag{15}$$

Due to Equation (2), we have:

$$\frac{\partial \mathbf{c}_y}{\partial \mathbf{x}_i} = \frac{1}{n_y}. \tag{16}$$

Taking Equation (15) and (16) into Equation (14), we get:

$$\begin{aligned}\gamma \frac{\partial L_I}{\partial \mathbf{x}_i} &= \gamma (||\mathbf{c}_y||_2^2 - \tau)(1 - \frac{1}{m})\mathbf{c}_y \frac{1}{n_y} \\ &= \gamma \frac{1}{n_y} (1 - \frac{1}{m})(||\mathbf{c}_y||_2^2 - \tau)\mathbf{c}_y. \end{aligned} \tag{17}$$

From Equation (17), we can see that the gradients of center invariant loss with respect to $\mathbf{x}_i$ follow either the direction or the opposite direction of the center $\mathbf{c}_y$. When the Euclidean norm of center is smaller than the mean center norm ($||\mathbf{c}_y||_2^2 - \tau \leq 0$), $\frac{\partial L_I}{\partial \mathbf{x}_i}$ has the opposite direction of $\mathbf{c}_y$. It means the center $\mathbf{c}_y$ will be pulled out with $\mathbf{x}_i$ since the loss function is minimized. Similarly, the center $\mathbf{c}_y$ will be pushed in if Euclidean norm of center is greater than the mean center norm ($||\mathbf{c}_y||_2^2 - \tau \geq 0$).

The last term $\lambda \frac{\partial L_c}{\partial \mathbf{x}_i}$ in Equation (8) can be written as:

$$\lambda \frac{\partial L_c}{\partial \mathbf{x}_i} = \lambda(\mathbf{x}_i - \mathbf{c}_y). \tag{18}$$

Taking Equation (18) , (17) and (11) into Equation (8), the gradients become:

$$\frac{\partial L}{\partial \mathbf{x}_i} = (1 - g(\mathbf{x}_i))\mathbf{w}_y + \gamma \frac{1}{n_y}(1 - \frac{1}{m})(||\mathbf{c}_y||_2^2 - \tau)\mathbf{c}_y + \lambda(\mathbf{x}_i - \mathbf{c}_y), \tag{19}$$

where $g(\mathbf{x}_i)$ is defined in Equation (10), τ can be found at Equation (12). Given these gradients, we can optimize the CNN model with stochastic gradient descent (SGD).

3 EXPERIMENTS

In this section, we firstly describe implementation details about data preprocessing, training and testing. Then we show how the center invariant loss influences the distribution of deeply learned features. And we investigate the sensitiveness of the parameters on the face recognition problem. Finally, we compare our model with other state-of-the-art methods.

3.1 Implementation Details

In this subsection, we describe data preprocessing, training/test data and detailed settings in CNNs.

Data Preprocessing. All faces and landmarks are detected by Mutli-task Cascaded Convolutional Networks (MTCNN) [42]. We use five landmarks (two eyes, nose and mouth corners) for similarity transformation. These faces are cropped to 112x96 RGB images. Each pixel is normalized by subtracting 127.5 then dividing by 128.

Traininig/Test data We use the CASIA-WebFace [40] dataset for training CNN models with 0.5 million faces. After removing the faces which fail to be detected by MTCNN, the CASIA-WebFace data set has 0.45 million faces with 10,575 identities.

The CNN models are evaluated on two famous face benchmarks: Label Face in the Wild (**LFW**) [9] contains 13,233 web images from 5,749 different identities, with large variations in pose, expression and illuminations. We follow the standard protocol of unrestricted with labeled outside data and test on 6,000 face pairs and report the performance. Some faces are shown in Figure 5. Positive samples are at the first row and negative samples are shown in the second row.

Youtube Face Database (**YTF**) [35] consists of 3,425 videos of 1,595 different identities, with an average of 2.15 videos per person. Clip durations vary from 48 frames to 6,070 frames. An average length is 181.3 frames. Again, we follow the unrestricted with labeled outside data protocol and report results on 5,000 video pairs. Some face

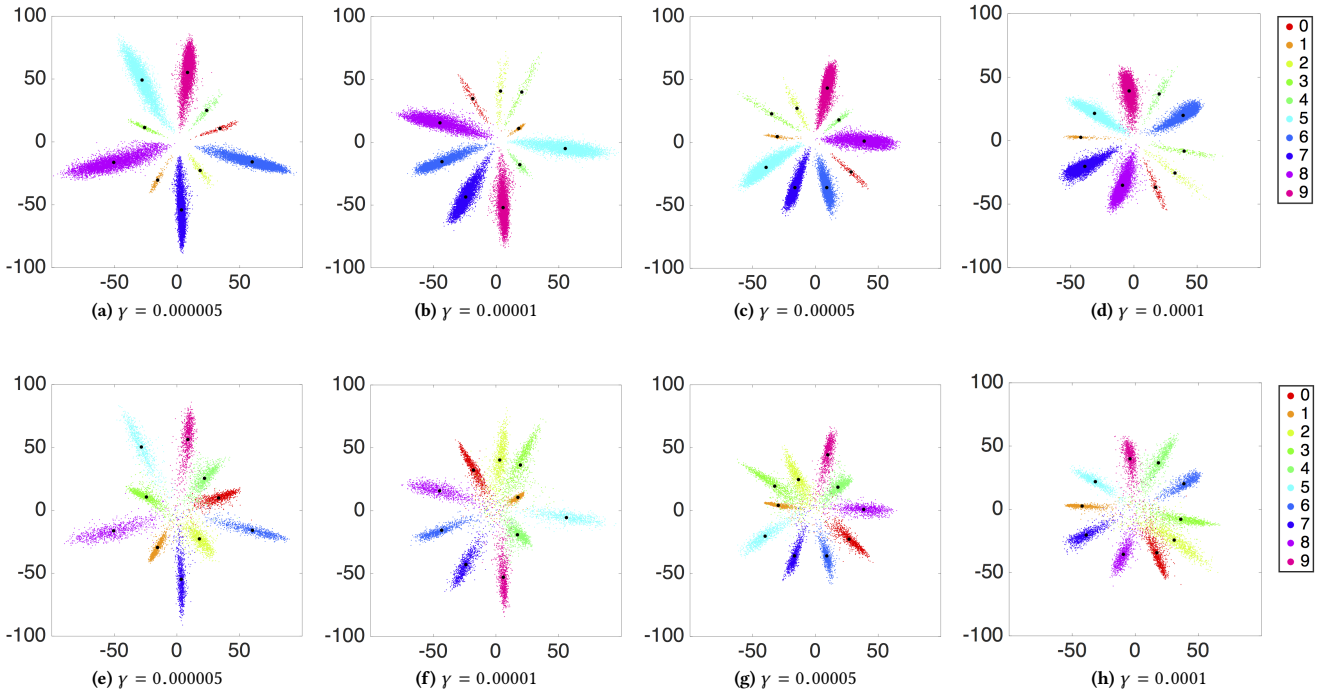

Figure 4: Distributions of deeply learned features with different γ on train/test sets. Centers of each class are denoted as black dots. Will increasing of the γ, centers of each glass get aligned gradually.

Figure 5: Face images in LFW and YTF datasets.

images of two different people are shown in the last two rows in Figure 5.

Detailed settings in CNNs The CNN model is a 22-layer residual network[1] implemented using Caffe [10] library. For a fair comparison, we train two kinds of models under supervision of softmax loss (**Baseline A**.) and softmax + center loss (**Baseline B**). The two models are trained with batch size of 256 on four PASCAL Titan X GPUs. The learning rate is started from 0.1 and divided by 10 at the 16K, 24K iterations. A complete training is finished at 28K iterations. The training takes roughly 3 hours. For the model with new loss function (Equation (6)), it is fine-tuned from the baseline B model with the same data. The learning rate is fixed to 0.001 and the fine-tuning takes 20K iterations with roughly 2 hours.

After the CNN models finish training, the deep features are taken from the last hidden layer. We extract features for each face and its horizontally flipped image. Then the features are concatenated as the representation, which follows [34]. The similarity score between one pair of images is computed by the Cosine Distance after PCA.

[1]https://github.com/ydwen/caffe-face

Table 1: Accuracy of classification On the MNIST test set with imbalance training data

Method	Accuracy (%)
softmax	93.95
Ours	95.03

3.2 Experiments on MNIST

In this subsection, we conduct experiments to illustrate how the γ influences the distribution of the deeply learned features on the MNIST dataset. The loss that is used in this subsection is Equation (4). We take the 200 samples setting in section 2.1. In Figure 4, we can observe that different γ leads to different deep feature distributions. With proper γ, the feature space is almost equally split by all classes, even there is a huge difference in the number of training samples (e.g. 200 vs 6,000). With the increase of γ, the penalty of differences among class center Euclidean norms gets larger and larger. As shown in Figure 4a, 4b, 4c and 4d, all centers are aligned to have more and more similar magnitudes and better separation of the feature space with the increase of γ. From Figure 4e, 4f, 4g and 4h for the deeply learned features on the test set, we can also see the same phenomenon. We also compare the classification accuracy between the softmax loss and center invariant loss, shown in Table 1. The result of ours refers to the model with $\gamma = 0.0001$. We can observe an improvement from 93.95% to 95.03%. Thus, center invariant loss could help deeply learned features to separate the feature space

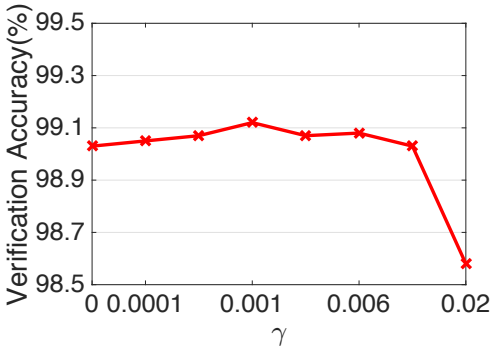

Figure 6: Face verification accuracies on LFW dataset. Different γ was tested given fixed $\lambda = 0.08$.

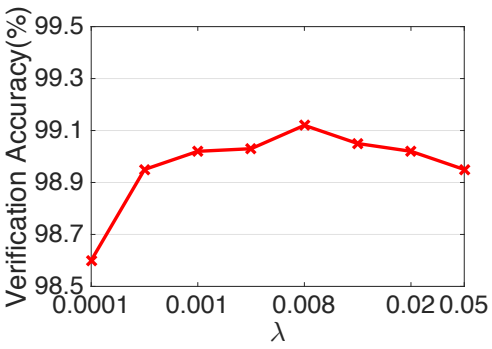

Figure 7: Face verification accuracies on LFW dataset. Different λ was tested given fixed $\gamma = 0.001$.

equally for all classes given imbalanced training data and improve the classification performance.

3.3 Experiments on Parameters

We have two hyper parameters λ and γ in Equation (6). λ is for the intra-class variations in Equation (5) and γ is for the variations between centers in Equation (3).

We first investigate the sensitiveness of γ given fixed $\lambda = 0.008$. γ is varied from 0 to 0.02. The verification accuracies of models with different γ on LFW dataset are shown in Figure 6. We can observe that accuracies go up from 0 to 0.001, which demonstrates that our method could boost performance. And we can see our model is stable across a wide range of γ. When γ goes to 0.02, accuracies drop since models overfit the data.

The sensitivity of λ was initially investigated in [34]. However, since we introduce a new term called the center invariant loss (Equation (3)), we re-investigate the sensitiveness of λ given fixed $\gamma = 0.001$. Results on LFW are shown in Figure 7. We can see that a proper λ is important for a good performance. Smaller λ is not a good choice since it can not reduce large intra-class variations. A wide range of λ can also make the model stable. Large λ, *e.g.* 0.05, leads to overfitting and decreases the accuracy.

3.4 Results on LFW and YTF

In this subsection, we evaluate our model on LFW and YTF face recognition benchmarks. γ is fixed to 0.001 and λ is set to 0.008.

In Table 2, we compared our method against many state-of-the-art methods and baseline models. From the results, we can see that our model beats the baseline A (softmax loss only) by a significant margin. The performance is improved from 97.33% to 99.12% on LFW and from 90.88% to 93.88% on YTF. Compared with baseline B (softmax loss + center loss), our model still achieves better results given very high recognition rate (more than 99% accuracy) on LFW and boost the performance from 93.84% to 93.88% on YTF. This shows that the proposed center invariant loss can improve the generalization ability of deeply learned features. Compared with other models [2, 19, 22, 32, 40] trained on CASIA-WebFace, our model achieves the best result on LFW. This demonstrates the advantage of our model. Moreover, our model is trained with 0.45 million faces, which are much less than many other state-of-the-art models [23, 25, 29, 30, 43]. But we achieve comparable results, which show the advantages of the proposed model.

Table 2: Accuracy of verification performance of different methods on LFW and YTF datasets

Method	#Images/Dataset	#Net	LFW	YTF
DeepFace [29]	4M	3	97.35%	91.40%
FaceNet [25]	200M	1	99.63%	95.10%
DeepID-2+ [28]	-	1	98.70%	-
DeepID-2+ [28]	-	25	99.47%	93.20%
VGG Face [23]	2.6M	1	98.95%	92.80%
MFM [36]	8.9M	1	98.80%	93.40%
Web-Scale [30]	500M	1	98.00%	-
Web-Scale [30]	500M	4	98.37%	-
CenterFace [34]	0.7M	1	99.28%	94.90%
Rangeloss [43]	1.5M	1	99.52%	93.70%
LFS [40]	CASIA-WebFace	1	97.73%	92.24%
FSS [32]	CASIA-WebFace	1	97.45%	-
MMD [2]	CASIA-WebFace	1	98.43%	-
PSEA [22]	CASIA-WebFace	1	98.06%	-
LMS [19]	CASIA-WebFace	1	98.71%	-
Baseline A	CASIA-WebFace	1	97.33%	90.88%
Baseline B	CASIA-WebFace	1	99.03%	93.82%
Ours	CASIA-WebFace	1	99.12%	93.88%

4 CONCLUSION

In this paper, we proposed a center invariant loss which aligns the center of each person to enforce the learned feature to have a general representation for all people. The center invariant loss penalized the difference between each center of classes. With joint supervision of softmax loss, center invariant loss and Center loss, we could train a robust CNN that treats each class equally regardless the number of class samples. Extensive experiments demonstrated the effectiveness of the proposed approach. In the future, we will apply our model to other applications, such as, object detection, object recognition, scene classification, one-shot learning and low-shot learning.

5 ACKNOWLEDGMENTS

This work is supported in part by the ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

REFERENCES

[1] Laura Cabrera-Quiros and Hayley Hung. 2016. Who is where?: Matching People in Video to Wearable Acceleration During Crowded Mingling Events. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 267–271.

[2] Changxing Ding and Dacheng Tao. 2015. Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia* 17, 11 (2015), 2049–2058.

[3] Zhengming Ding, Ming Shao, and Yun Fu. 2016. Deep Robust Encoder Through Locality Preserving Low-Rank Dictionary. In *European Conference on Computer Vision*. Springer, 567–582.

[4] Yuan Dong and Yue Wu. 2015. Adaptive cascade deep convolutional neural networks for face alignment. *Computer standards & interfaces* 42 (2015), 105–112.

[5] Yandong Guo and Lei Zhang. 2017. One-shot Face Recognition by Promoting Underrepresented Classes. *arXiv preprint arXiv:1707.05574* (2017).

[6] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. 2016. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*. Springer, 87–102.

[7] Bharath Hariharan and Ross Girshick. 2016. Low-shot visual object recognition. *arXiv preprint arXiv:1606.02819* (2016).

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on CVPR*.

[9] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments.* Technical Report. Technical Report 07-49, University of Massachusetts, Amherst.

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[11] Shuhui Jiang, Ming Shao, Chengcheng Jia, and Yun Fu. 2016. Consensus Style Centralizing Auto-Encoder for Weak Style Classification.. In *AAAI*. 1223–1229.

[12] Shuhui Jiang, Yue Wu, and Yun Fu. 2016. Deep Bi-directional Cross-triplet Embedding for Cross-Domain Clothing Retrieval. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 52–56.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[14] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST database of handwritten digits. (1998).

[15] J. Li, H.Y. Chang, and J. Yang. 2015. Sparse Deep Stacking Network for Image Classification. In *Proc. of the AAAI Conf. on Artif. Intell.* 3804–3810.

[16] Jun Li, Tong Zhang, Wei Luo, Jian Yang, Xiaotong Yuan, and Jian Zhang. 2017. Sparseness Analysis in the Pertraining of Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 6 (2017), 1425–1438.

[17] Jianshu Li, Jian Zhao, Fang Zhao, Hao Liu, Jing Li, Shengmei Shen, Jiashi Feng, and Terence Sim. 2016. Robust Face Recognition with Deep Multi-View Representation Learning. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 1068–1072.

[18] Ying Li, Xiangwei Kong, Liang Zheng, and Qi Tian. 2016. Exploiting Hierarchical Activations of Neural Network for Image Retrieval. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 132–136.

[19] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-Margin Softmax Loss for Convolutional Neural Networks. In *Proceedings of The 33rd International Conference on Machine Learning*. 507–516.

[20] Wei Luo, Jun Li, Jian Yang, Wei Xu, and Jian Zhang. 2017. Convolutional Sparse Autoencoders for Image Classification. *IEEE Trans. Neural Netw. Learn. Syst.* doi:10.1109/TNNLS.2017.2712793 (2017).

[21] Haiyi Mao, Yue Wu, Jun Li, and Yun Fu. 2016. Super Resolution of the Partial Pixelated Images With Deep Convolutional Neural Network. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 322–326.

[22] Iacopo Masi, Anh Tuáž̌n Trážğn, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. 2016. Do we really need to collect millions of faces for effective face recognition?. In *European Conference on Computer Vision*. Springer, 579–596.

[23] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition.. In *BMVC*, Vol. 1. 6.

[24] Ravi Kiran Sarvadevabhatla, Shiv Surya, Srinivas SS Kruthiventi, and others. 2016. SwiDeN: Convolutional Neural Networks For Depiction Invariant Object Recognition. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 187–191.

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 815–823.

[26] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*. 1988–1996.

[27] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1891–1898.

[28] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2015. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2892–2900.

[29] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708.

[30] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2015. Web-scale training for face identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2746–2754.

[31] Vassilios Vonikakis, Ramanathan Subramanian, and Stefan Winkler. 2016. Shaping datasets: Optimal data selection for specific target distributions across dimensions. In *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 3753–3757.

[32] Dayong Wang, Charles Otto, and Anil K Jain. 2015. Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242* (2015).

[33] Yuhang Wang, Jing Liu, Yong Li, Junjie Yan, and Hanqing Lu. 2016. Objectness-aware Semantic Segmentation. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 307–311.

[34] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*. Springer, 499–515.

[35] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 529–534.

[36] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. 2015. A Light CNN for Deep Face Representation with Noisy Labels. *arXiv preprint arXiv:1511.02683* (2015).

[37] Yue Wu, Jun Li, Yu Kong, and Yun Fu. 2016. Deep Convolutional Neural Network with Independent Softmax for Large Scale Face Recognition. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 1063–1067.

[38] Shohei Yamamoto and Tatsuya Harada. 2016. Video Generation Using 3D Convolutional Neural Network. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 576–580.

[39] Hao Ye, Weiyuan Shao, Hong Wang, Jianqi Ma, Li Wang, Yingbin Zheng, and Xiangyang Xue. 2016. Face Recognition via Active Annotation and Learning. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 1058–1062.

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

[41] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. 2016. UnitBox: An Advanced Object Detection Network. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 516–520.

[42] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (2016), 1499–1503.

[43] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. 2016. Range Loss for Deep Face Recognition with Long-tail. *arXiv preprint arXiv:1611.08976* (2016).