# Large-Margin Softmax Loss for Conv. Neural Networks

Weiyang Liu[1*], Yandong Wen[2*], Zhiding Yu[3], Meng Yang[4]

[1]Peking University [2]South China University of Technology

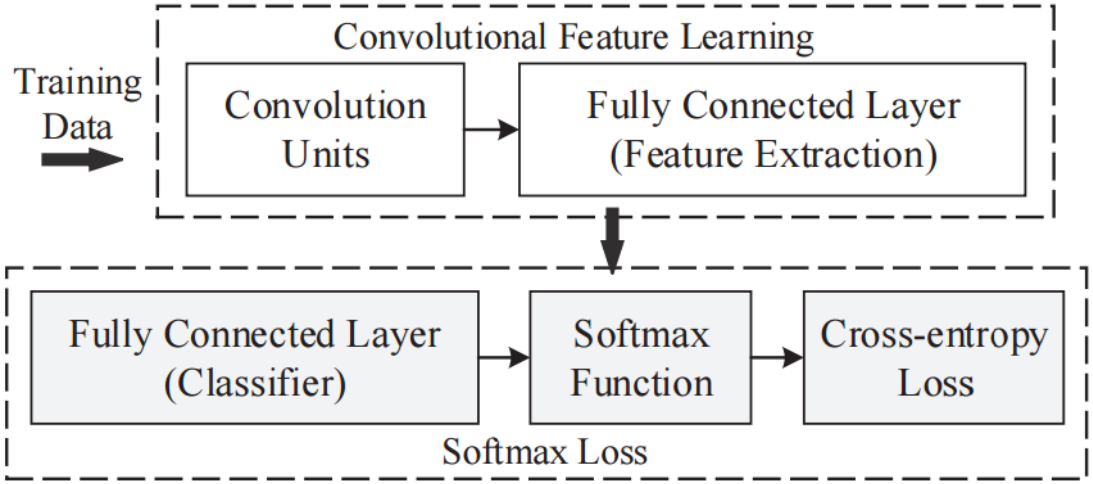
[3]Carnegie Mellon University [4]Shenzhen University

# Outline

- Introduction
- Softmax Loss
- Intuition: Incorp. Large Margin to Softmax
- Large-Margin Softmax Loss
- Toy Example
- Experiments
- Conclusions and Ongoing Works

# Introduction

- Many current CNNs can be viewed as conv feature learning guided by a softmax loss on top.




- Other popular losses include hinge loss (SVM loss), contrastive loss, triplet loss, etc.
- Softmax loss is easy to optimize but does not explicitly encourage large margin between different classes.

# Introduction

- **Hinge Loss:** explicitly favors the large margin property.

- **Contrastive Loss:** encourages large margin between inter-class pairs, and require distances between intra-class pairs to be smaller than a margin.

- **Triplet Loss:** similar to contrastive loss, except requiring selected triplets as input. The triplet loss first defines an anchor sample, and select hard triplets to simultaneously minimize the intra-class distances and maximize inter-class distance.

---

- **Large-Margin Softmax (L-Softmax) Loss:** generalized softmax loss with large inter-class margin.

# Introduction

The L-Softmax loss has the following advantages:

1. L-Softmax loss defines a **flexible learning task with adjustable difficulty** by controlling the desired margin.

2. With adjustable difficulty, L-Softmax can make better use of the "depth" and the learning ability of CNNs by **incorporating more discriminative information**.

3. Both contrastive loss and triplet loss require carefully designed pair/triplet selection to achieve best performance, while **L-Softmax loss directly addresses the entire training set**.

4. L-Softmax loss can be **easily optimized with typical stochastic gradient descent**.

# Softmax Loss

- Suppose the $i$-th input feature is $\boldsymbol{x}_i$ with label $y_i$ , the original softmax loss can be written as

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

where $f_j$ denotes the Euclidean dot product of the $j$-th class, and symbols the activations of a fully connected layer. The above loss can be further rewritten as:

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

# Intuition: Margin in Softmax

- Consider the ground truth is class-1. A necessary and sufficient condition for correct classification is: $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$

- L-Softmax makes the classification more rigorous in order to produce a decision margin. When training, we instead require

$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2) \ (0 \leq \theta_1 \leq \frac{\pi}{m})$$

  where $m$ is a positive integer.

- The following inequality holds:

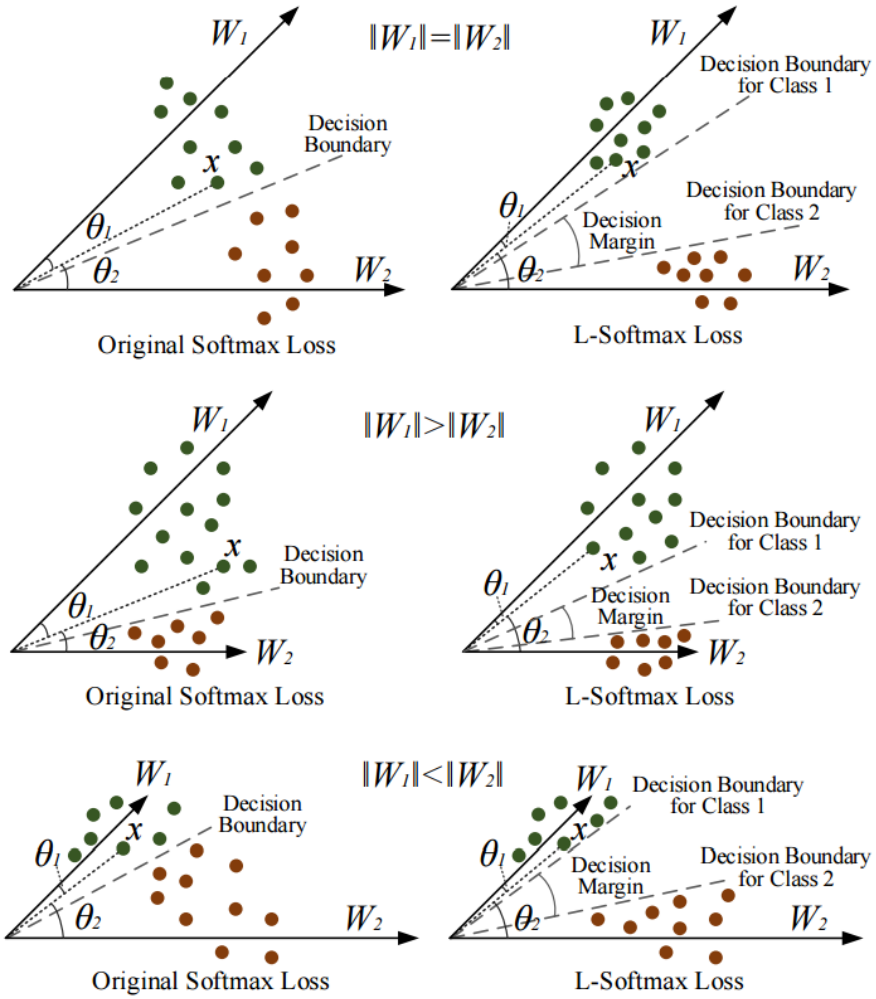
  Margin comes here! “>>” when m>1

$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) \geq \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2).$$

- The new classification criteria is a stronger requirement to correctly classify $\boldsymbol{x}$ , producing a more rigorous decision boundary for class-1.

# Geometric Interpretation

- We use binary classification as an example.
- We consider all three scenarios in which $\|\boldsymbol{W}_1\| = \|\boldsymbol{W}_2\|$, $\|\boldsymbol{W}_1\| > \|\boldsymbol{W}_2\|$ and $\|\boldsymbol{W}_1\| < \|\boldsymbol{W}_2\|$.
- L-Softmax loss always encourages an angular decision margin between classes.



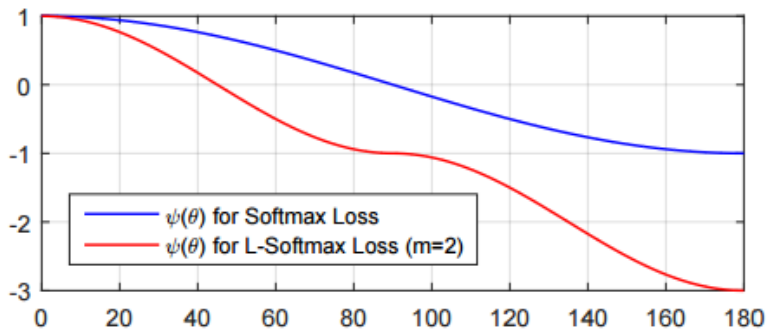
# L-Softmax Loss

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

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

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

- The parameter $m$ controls the learning difficulty of the L-Softmax loss. A larger $m$ defines a more difficult learning objective.



# Optimization

- Transform cos($m\theta$) into combinations of cos($\theta$):

$$\cos(m\theta_{y_i}) = C_m^0 \cos^m(\theta_{y_i}) - C_m^2 \cos^{m-2}(\theta_{y_i})(1-\cos^2(\theta_{y_i})) + C_m^4 \cos^{m-4}(\theta_{y_{y_i}})(1-\cos^2(\theta_{y_{y_i}}))^2 + \cdots (-1)^n C_m^{2n} \cos^{m-2n}(\theta_{y_{y_i}})(1-\cos^2(\theta_{y_i}))^n + \cdots$$

- Represent cos($\theta$) as $\frac{\boldsymbol{W}_j^T \boldsymbol{x}_i}{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|}$

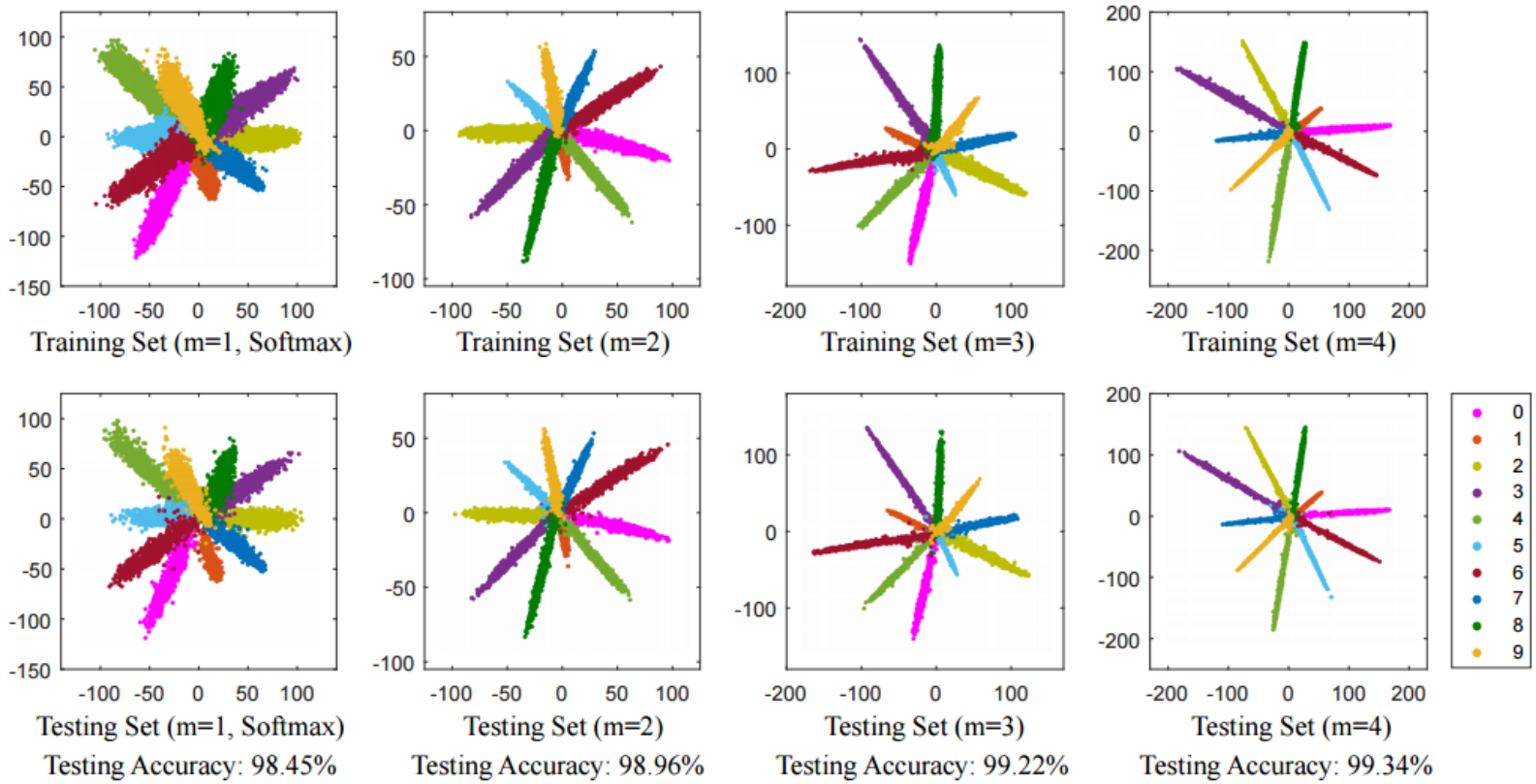
- In practice, we seek to minimize:

$$f_{y_i} = \frac{\lambda\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i}) + \|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}{1+\lambda}$$

- Start with large λ and gradually reduce to a very small value.

# A Toy Example

- A toy example on MNIST. CNN features visualized by setting the output dimension as 2.



# Experiments

- We use standard CNN architecture and replace the softmax loss with the proposed L-Softmax loss.

| 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.

- We adopt conventional setup in all datasets.
- We compare our L-Softmax loss with the same CNN architecture with standard softmax loss and other state-of-the-art methods.

# Experiments

- MNIST dataset

| 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.

- We can observe that CNN with L-Softmax loss achieves better results with larger m.

# Experiments

- CIFAR10, CIFAR10+, CIFAR100

| 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.

| 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.

- CNN with L-Softmax loss achieves the state-of-the-art performance on CIFAR 10, CIFAR10+ and CIFAR100.

# Experiments

- CIFAR10, CIFAR10+, CIFAR100

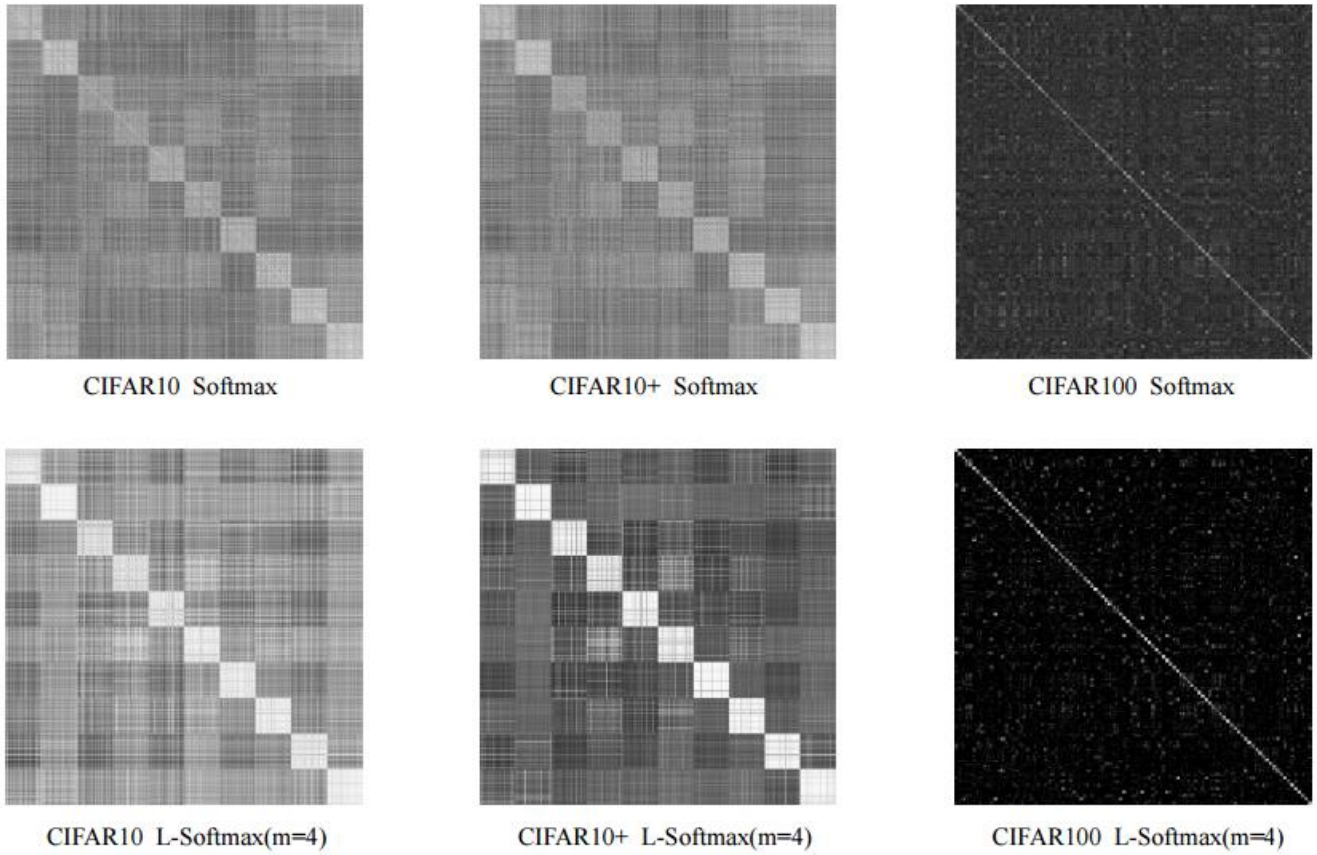



*Figure 5.* Confusion matrix on CIFAR10, CIFAR10+ and CIFAR100.

We observe that the deeply learned features through L-Softmax are more discriminative.

# Experiments

- CIFAR10, CIFAR10+, CIFAR100
- Classification error vs. iteration. Left: training. Right: testing.

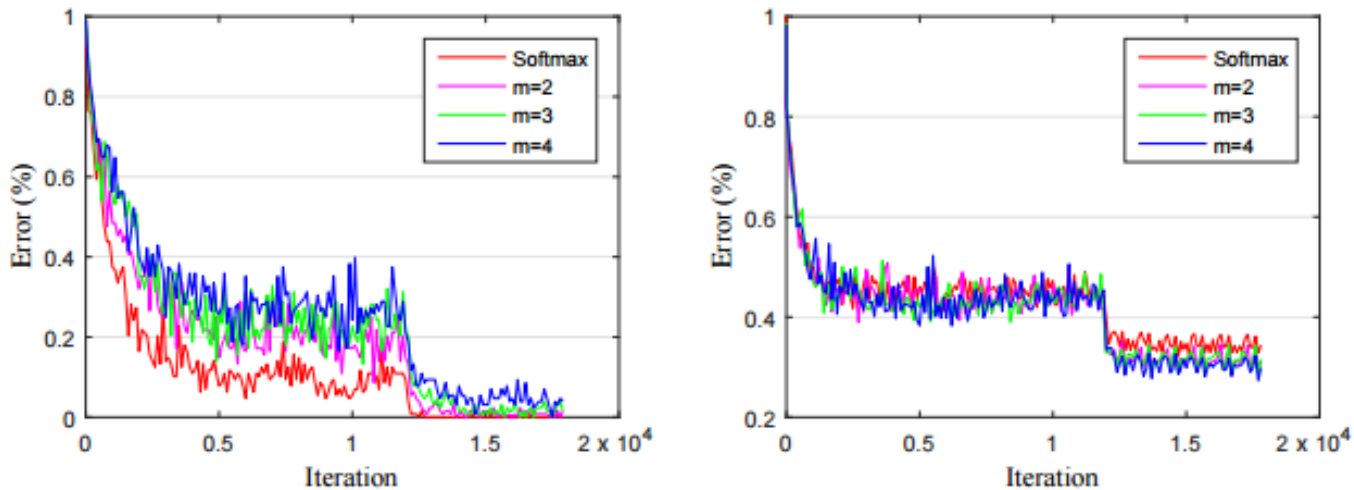



*Figure 6.* Error vs. iteration with different value of $m$ on CIFAR100. The left shows training error and the right shows testing error.

- From the above figures, we see that L-Softmax is far from overfitting.
- L-Softmax loss does not achieve the state-of-the-art performance by overfitting the dataset.

# Experiments

- CIFAR10, CIFAR10+, CIFAR100
- Classification error vs. iteration. Left: training. Right: testing.

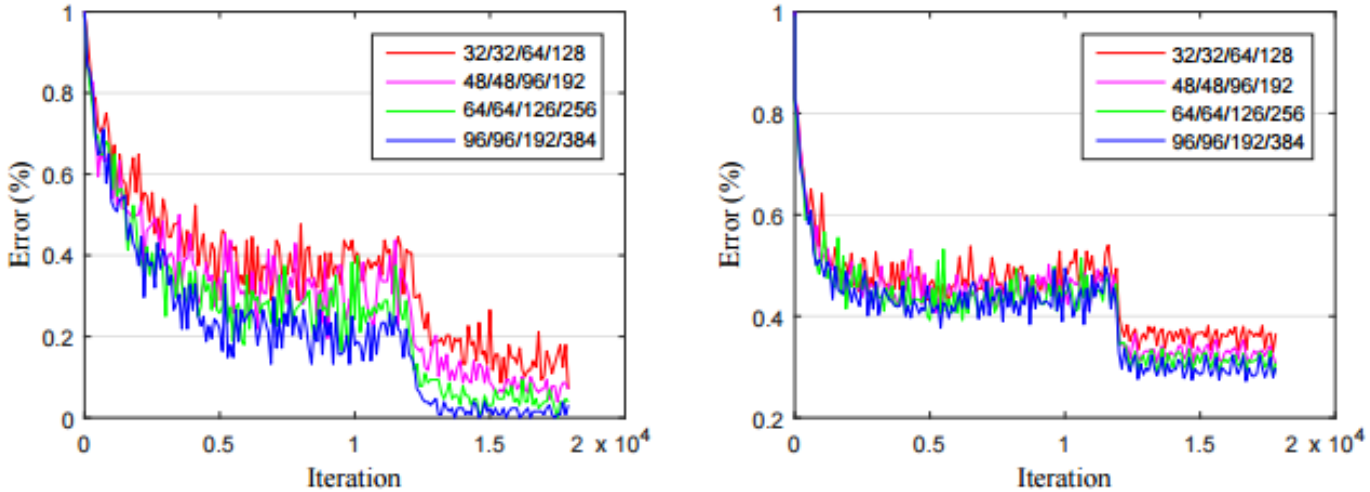



*Figure 7.* Error vs. iteration (m=4) with different number of filters on CIFAR100. The left (right) presents training (testing) error.

- More filters could also improve the performance, showing that our L-Softmax still have great potential.

# Experiments

- LFW face verification
- We train our CNN model on publicly available WebFace face dataset and test on LFW 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).

- We achieve the best result with WebFace outside training dataset.

# Conclusions

- L-Softmax loss has very clear intuition and simple formulation.
- 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 modules.
- L-Softmax loss can be easily optimized using typical stochastic gradient descent.
- L-Softmax achieves state-of-the-art classification performance and prevents the CNNs from overfitting, since it provides a more difficult learning objective.
- L-Softmax makes better use of the feature learning ability brought by deeper structures.

# Ongoing Works

- We found such large-margin design is very suitable for verification problems since the essence of verification is learning the distances.

- Out latest progress on face verification has achieved state-of-the-art performance on **LFW** and **MegaFace Challenge**.

- Trained with **CASIA-WebFace (~490K)**, we achieved:

  **MegaFace:**
  **72.729%** with 1M distractors (**Rank-1** on small protocol)
  **85.561%** with TAR for 10e-6 FAR (**Rank-1** on small protocol)

  **LFW: 99.42%** Accuracy.

- Our result is comparable to (with **490K** data) **Google FaceNet** (with **500M** data).

# Ongoing Works

**LFW**

| Method | Models | Data | Acc. on LFW | Acc. on YTF |
|---|---|---|---|---|
| DeepFace [21] | 3 | 4M* | 97.35 | 91.4 |
| FaceNet [16] | 1 | 200M* | **99.65** | 95.1 |
| Deep FR [15] | 1 | 2.6M | 98.95 | **97.3** |
| DeepID2+ [20] | 1 | 300K* | 98.7 | N/A |
| DeepID2+ [20] | 25 | 300K* | 99.47 | 93.2 |
| Baidu [11] | 1 | 1.3M* | 99.13 | N/A |
| Yi et al. [23] | 1 | WebFace | 97.73 | 92.2 |
| Ding et al. [3] | 1 | WebFace | 98.43 | N/A |
| CNNs with L-Softmax [12] | 1 | WebFace | 98.71 | N/A |
| Standard CNNs 1 (4 conv layers) | 1 | WebFace | 96.63 | 91.1 |
| Standard CNNs 2 (10 conv layers) | 1 | WebFace | 97.12 | 92.0 |
| Standard CNNs 3 (20 conv layers) | 1 | WebFace | 97.50 | 92.6 |
| Standard CNNs 4 (36 conv layers) | 1 | WebFace | 97.75 | 92.9 |
| Standard CNNs 5 (64 conv layers) | 1 | WebFace | 97.88 | 93.1 |
| LAM-CNNs 1 (4 conv layers) | 1 | WebFace | 98.20 | 93.4 |
| LAM-CNNs 2 (10 conv layers) | 1 | WebFace | 99.03 | 93.7 |
| LAM-CNNs 3 (20 conv layers) | 1 | WebFace | 99.26 | 94.1 |
| LAM-CNNs 4 (36 conv layers) | 1 | WebFace | 99.35 | 94.2 |
| LAM-CNNs 5 (64 conv layers) | 1 | WebFace | **99.42** | **94.5** |

# Ongoing Works

**MegaFace**

| Method | protocol | Rank-1 Id. Acc. with 1M distractors | Ver. TAR for $10^{-6}$ FAR |
|---|---|---|---|
| NTechLAB - facenx large | Large | **73.300** | 85.081 |
| Google - FaceNet v8 | Large | 70.496 | **86.473** |
| Beijing Faceall Co. - FaceAll_Norm_1600 | Large | 64.804 | 67.118 |
| Beijing Faceall Co. - FaceAll_1600 | Large | 63.977 | 63.960 |
| Barebones FR - cnn | Small | 59.363 | 59.036 |
| NTechLAB - facenx_small | Small | 58.218 | 66.366 |
| 3DiVi Company - tdvm6 | Small | 33.705 | 36.927 |
| LAM-CNNs 1 (4 conv layers) | Small | 57.529 | 68.547 |
| LAM-CNNs 2 (10 conv layers) | Small | 65.335 | 78.069 |
| LAM-CNNs 3 (20 conv layers) | Small | 69.623 | 83.159 |
| LAM-CNNs 4 (36 conv layers) | Small | 71.257 | 84.052 |
| LAM-CNNs 5 (64 conv layers) | Small | **72.729** | **85.561** |

# Thank you