Playing with Neural nets compression

Author: Hy Truong Son

The University of Chicago

Chicago, April 2017

Contents

1. Neural Networks Architectures
2. Compression Methods
3. Visualization
4. Low-rank approximation
5. Sparsification
6. K-Means quantization
7. Fourier Transformation
8. Conclusions
9. How to compress Deep CNNs?

Architectures

Weight matrices: W, W_0, W_1
Input data matrix: $\mathcal{X}$
Prediction matrix: $\hat{\mathcal{Y}}$
Activation function: σ

- Softmax: $\hat{\mathcal{Y}} = \sigma(W\mathcal{X})$
- Autoencoder (unsupervised learning):

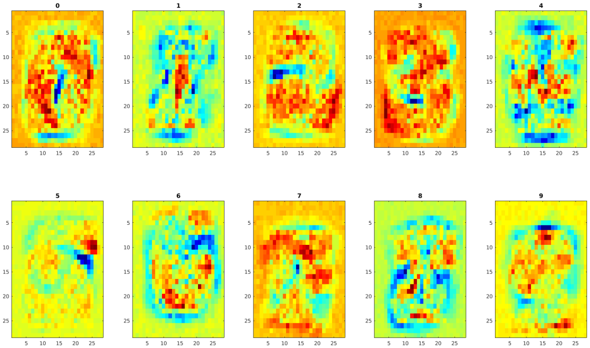
  $$\hat{\mathcal{Y}} = \sigma(W^T \sigma(W\mathcal{X}))$$

  This case the second-layer's weight matrix is the **transpose** of the first-layer's one.
- Multi-Layer Perceptron: $\hat{\mathcal{Y}} = \sigma(W_1 \sigma(W_0 \mathcal{X}))$

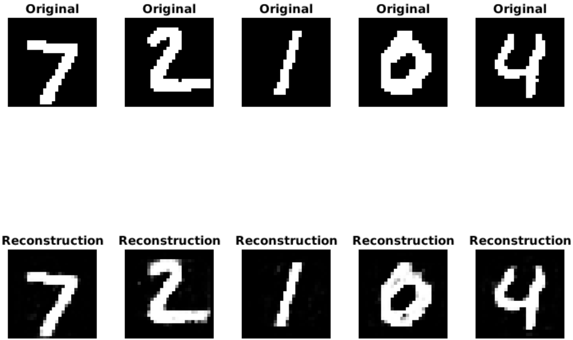
Compression Methods

- Low-rank approximation: Singular Value Decomposition
- Sparsification
- K-Means quantization
- Fast Fourier Transformation: Compression in the frequency domain

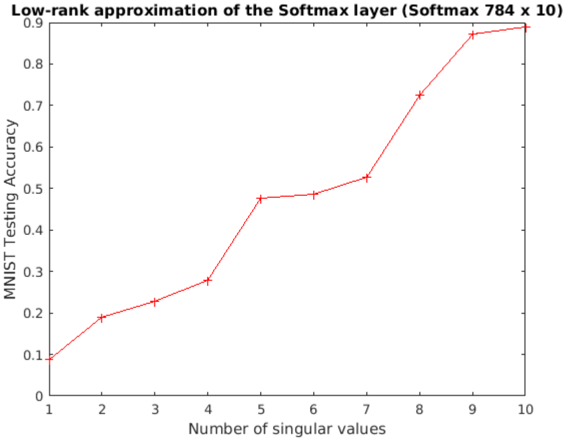
Softmax



Autoencoder

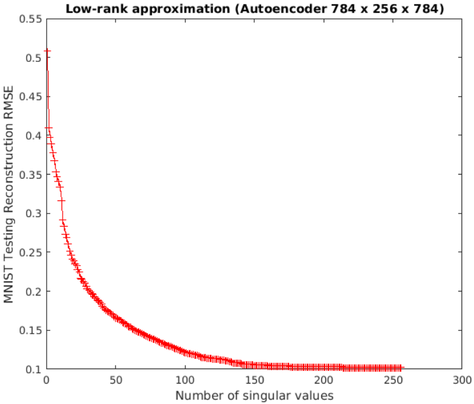


SVD - Softmax



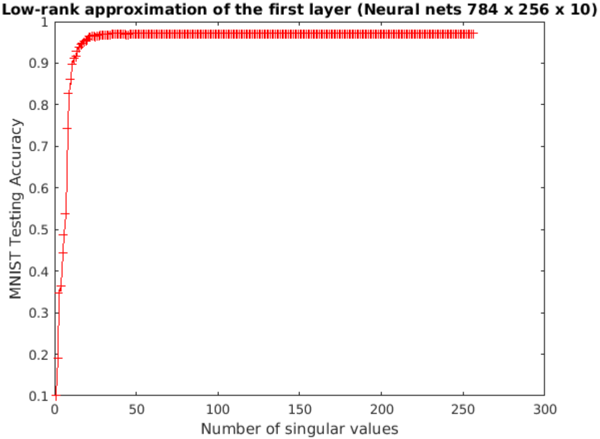
This kind of **shallow** neural network is **sensitive** to compression!

SVD - Autoencoder



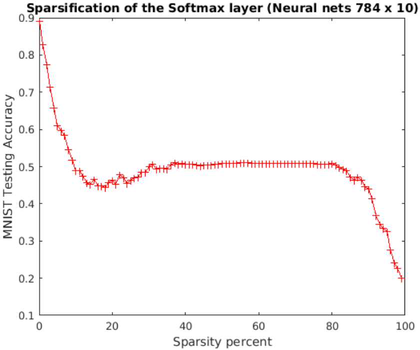
We can cut more than half of the number of singular values to get **acceptable** reconstruction error.

SVD - MLP



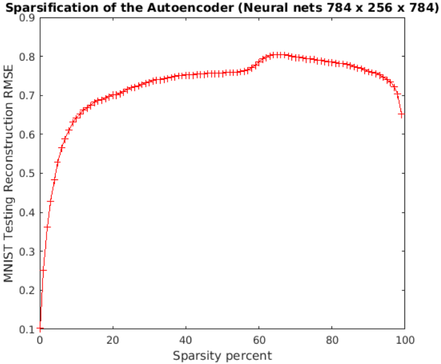
Only need to keep 18 first singular values to get more than 95% testing accuracy.

Sparsification - Softmax



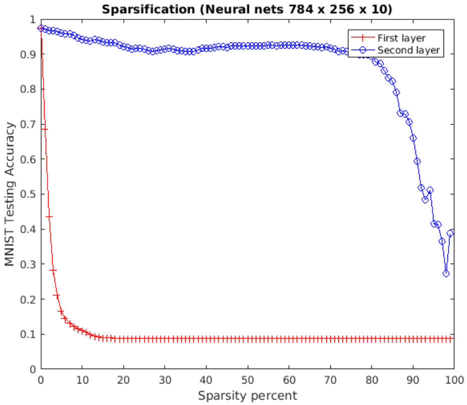
Still, **shallow** nets are **sensitive** to compression!

Sparsification - Autoencoder



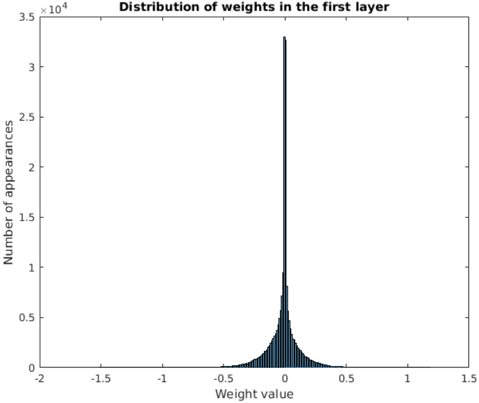
Autoencoder is **sensitive** to **sparsification**!

Sparsification - MLP

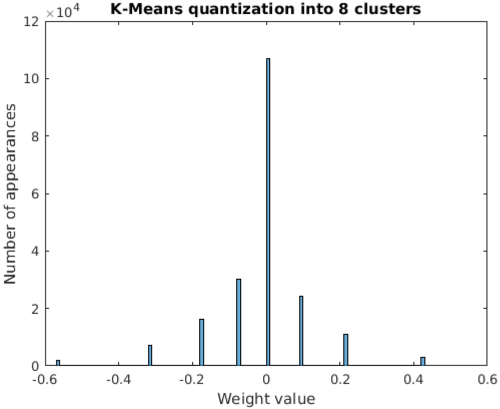


We can **throw out** 80% of the second layer to get more than 90% testing accuracy. That is **5**-time compression.

K-Means - Figure 1

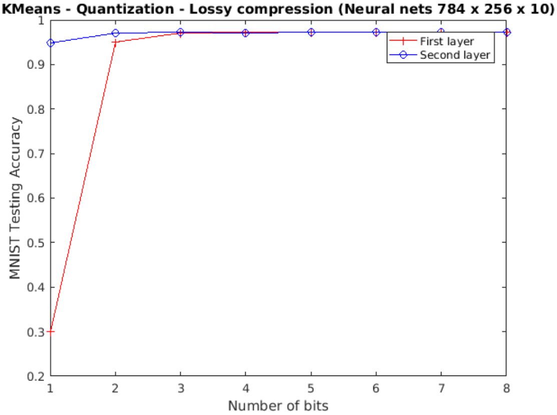


K-Means - Figure 2



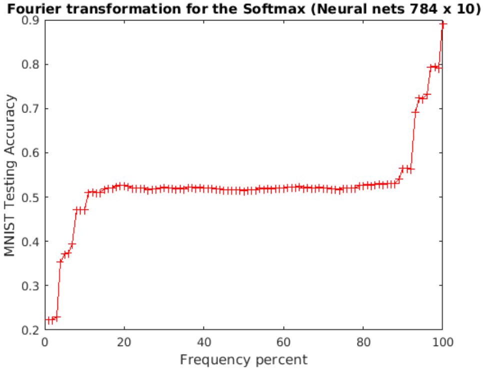
I did the K-Means quantization with a very efficient algorithm $O(k \cdot d \cdot log(n))$ where k is the number of clusters, d is the number of iterations, and n is the number of data points.

K-Means - Figure 3



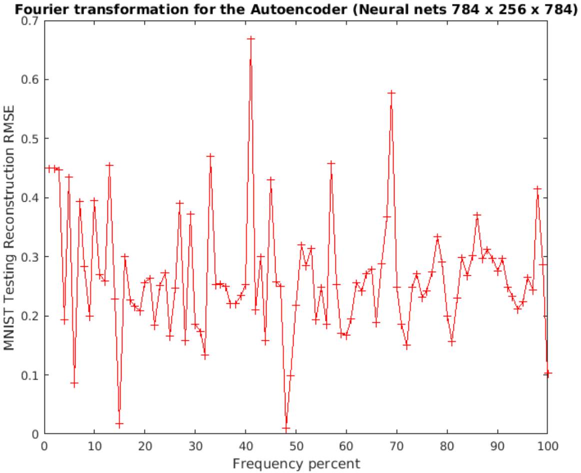
We need only **2** bits for each weight in the first layer, and **1** bit for the second layer to get 95% testing accuracy. Comparing to **4**-byte double-floating-point, we can obtain **32** time compression.

FFT - Softmax



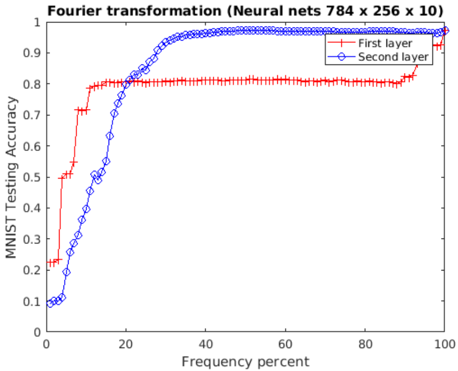
Basically, we cannot compress the **shallow** nets!

FFT - Autoencoder



Autoencoder is **sensitive** in the **frequency** domain!

FFT - MLP



We need to keep **10%** of the frequencies in the first layer to get **80%** testing accuracy. On the second layer, to obtain **95%** accuracy, we need to keep only **30%** of the frequencies.

Conclusions

- There are a lot of redundancy in MLP
- We only need **2** bits for each weight in the first layer, and **1** bits for the second layer
- Further lossless compression by Huffman code or LZW can be applicable

My source code:
https://github.com/HyTruongSon/Neural-Nets-Compression

How to compress Deep CNNs?

Reference:

- Multiresolution Matrix Compression/Factorization, Prof. Risi Kondor (UChicago)
- Soft Weight-Sharing For Neural Network Compression, ICLR 2017, Karen Ullrich, Max Welling

Some more practical works:

- Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications, ICLR 2016
- Deep Compression: Compressing Deep Neural Networks With Prunning, Trained Quantization and Huffman Coding, ICLR 2016