Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 5th, 2017

Handwritten Digit Recognition Using Stacked Autoencoders

Yahia Saeed, Jiwoong Kim, Lewis Westfall, and Ning Yang
Seidenberg School of CSIS, Pace University, New York

***Abstract*—This paper will focus on applying neural network machine learning method (Stacked Autoencoders) in order to obtain the highest accuracy possible of classification of handwritten digits. The main advantage of using neural network methods in this project is its adeptness at fitting non-linear data and its ability to work as an unsupervised algorithm. The algorithms will be run on a common, publically available dataset, namely the MNIST.**

***Index Terms*—Neural network, Autoencoder, Digit recognition, NMIST**

I. Introduction

Optical character recognition (OCR) is the interpretation of handwritten or printed text into a format that can be processed mechanically or electronically. OCR has been researched and used since the early 1900s to convert printed text into machine processable data [9]. The ability to process fixed font characters is well established but accuracy is hampered by the amount of noise in the image. Hand written or hand printed characters are a much more difficult problem than fixed font due to the noise and lack of consistency. In this paper we are exploring the use of a stacked autoencoders to function as an OCR.

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be similar to the inputs. I.e., it uses $y(i)=x(i)$y(i)=x(i).[1]

An autoencoder is a neural network (see Fig. 1) that is trained to encode an input x into some representation c(x) so that the input can be reconstructed from that representation. While the identity function - copying inputs to outputs - is not a useful function to learn, we can add additional constraints on either the network, or the error criterion in hopes that the network captures some useful features of the input (in contrast a vanilla neural network with s1 ≤ s2 would probably just learn the identity function). [2]

The autoencoder tries to learn a function $hW,b(x)\approx x$hW, b(x)≈x. In other words, it is trying to learn an approximation to the identity function, so as to output x^x^ that is similar to xx. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data [4].

II. Sparse Autoencoder

The Autoencoder is an artificial neural network used for unsupervised learning of efficient encodings. [5] Its aim is to learn a representation for a set of data, typically for the purpose of dimensionality reduction. This process is called encoding. Its structure can be described in figure.1.

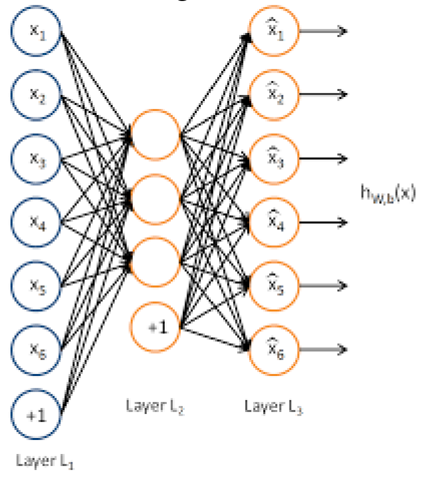


Figure. 1
Deep Autoencoder

The Autoencoder has three layers: the input layer, the hidden layer and the output layer. Its function is to compress the input data in the hidden layer and reconstruct them in the output layer to get the result $H_{w,b}(x)$ which is mainly the same as the input data.

$$H_{w,b}(x) \approx \text{x}.$$

In order to ensure the recovered data has less loss, we use a function $J(w,b)$ to calculate the differences between them.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(x' - x)^2$$

In order to learn useful structures in the input data, we would impose sparsity on the hidden units during training. Sparsity may be achieved by additional terms in the loss function during training (by comparing the probability distribution of the hidden unit activations with some low desired value) [6], or by manually zeroing all but the few strongest hidden unit activations. [7]

The output in the hidden layer l can be described in function.2:

$$\alpha_i^{(l)} = f\left(\sum_{j=1}^{n} W_{IJ}^{(l-1)} x_j + b_i^{l-1}\right)$$

The function $f(x)$is called activaton function, we usually use sigmoid function or hyperbolic tangent (tanh) function as activation signal. The sigmoid function has limits of 0 to 1, while the tanh function has limits of -1 to 1.

$$f(z) = \frac{1}{1+e^{-z}}$$

$$f(z) = \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

In our research we use a tanh function as our activation signal. It will be either 1 or -1. When it is 1, the neuron is activated and when it is -1 the neuron is not activated. Thus, we need to write $\alpha_j^{(2)}(x)$ to denote the activation of this hidden unit when the network is given a specific input x, the function is:

$$\rho_j' = \frac{1}{m}\sum_{i=1}^{m}\left[\alpha_j^{(2)}\left(x^{(i)}\right)\right]$$

It is the average activation of hidden unit j (averaged over the training set).

In order to enforce the constraint $\rho_j'=\rho$ where ρ called sparsity parameter. It is typically a small value close to zero (we will use a default value of 0.05). To satisfy this constraint, the hidden unit's activations must mostly be near 0. We add an extra penalty term to make sure the penalizes ρ_j' would not deviate significantly from ρ. The cost function should be:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(x' - x)^2 + \beta * PenalTerm$$

$$PenalTerm = \sum_{j=1}^{s2} \rho * \log\frac{\rho}{p_j} + (1-\rho)\log\frac{1-\rho}{1-p_j}$$

It can also be written $\sum_{j=1}^{s2} KL\left(\rho||\rho_j'\right)$, thus it should be:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(x' - x)^2 + \beta * \sum_{j=1}^{s2} KL\left(\rho||\rho_j'\right)$$

β controls the weight of the sparsity penalty term, and the term also depends on *W,b*. Because it is the average activation of hidden unit j, and the activation of a hidden unit depends on the parameter *W,b*.

The parameter we used in this paper are showed in table1:

TABLE1
Parameters Used

Symbol	Meanings
x	Input features for a training example, x$\in \Re^n$
$H_{w,b}(x)$	Output of our hypothesis on input x, using parameters *W, b*. This should be a vector.
$W_{IJ}^{(l)}$	The parameter associated with the connection between unit *j* in layer *l*, and unit *i* in layer *l+1*
b_i^l	The bias term associated with unit *i* in layer *l+1*. Can also be thought of as the parameter associated with the connection between the bias unit in layer *l* and unit *i* in layer *l+1*.
$f(z)$	The activation function using sigmoid function or tanh
ρ	Sparsity parameter, which specifies our desired level of sparsity
ρ_j'	The average activation of hidden unit *j* (in the sparse autoencoder).
β	Weight of the sparsity penalty term (in the sparse autoencoder objective).

III. SoftMax Regression

SoftMax regression is a supervised learning algorithm [12]. It will be used to classify the output from the stacked autoencoders into multiple classifications. The goal is to match the output of the stacked autoencoders of the MNIST images with their numerical value.

IV. Data

The **MNIST database** (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 20x20 pixel bounding box and anti-aliased, which introduced grayscale levels. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field [13]. Figure 2 is a sample of the MNIST images.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from MNIST's training dataset, while the other half of the training set and the other half of the test set were taken from MNIST's testing dataset. There have been a number of scientific papers on attempts to achieve the lowest error rate; one paper, using a hierarchical system of convolutional neural networks, manages to get an error rate on the MNIST database of 0.23 percent. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support vector machine to get an error rate of 0.8 percent.

V. Previous Work

The following table (Table2) is excerpted from LeCun et al. [13]

TABLE2

Method	Accuracy	Reference
Linear Classifier	92.4%	[15]

K Nearest Neighbor	99.3%	[16]
Boosted Stumps	99.1%	[17]
Non-Linear Classifier	96.4%	[15]
SVM	99.4%	[18]
Neural Net	99.6%	[19]
Convolutional Net	99.7%	[20]

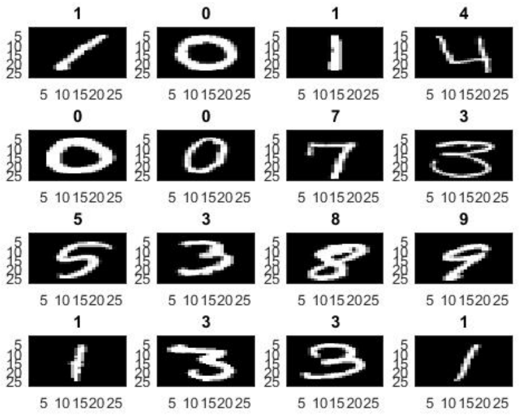


Figure. 2
Sample of The Raw Dataset

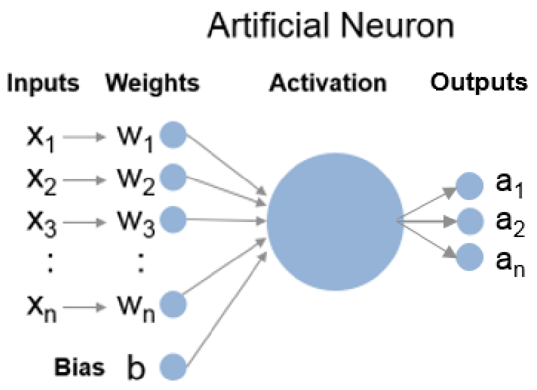


Figure. 3
Fully connected Neural networks

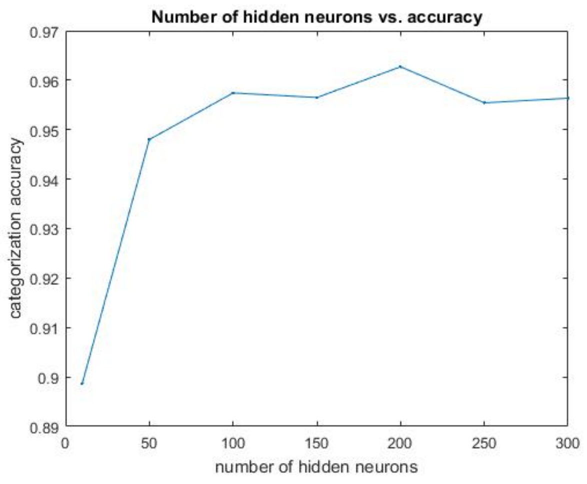


Figure. 4
Curve of Number of hidden neurons vs.accury

VI. Methodology

Previously [10] we had test the same dataset with a fully connected neural networks technique as shown in fig 3, and we used simple XY plot, as shown in fig 4, to plot the accuracy we got (96.3) According to this result we needed to build an Autoencoder classifier to compare it with our previous work.
Matlab Neural Network Toolbox autoencoder functionality will be used to build, train, test the stacked autoencoder classification system.

A dataset containing 10,000 images was downloaded from the MNIST database. Each with 784 data points.Once the training data was loaded into memory, the first autoencoder was defined. The random number generator was explicitly set to default value. The number of nodes in the hidden layer was set to 100. The first autoencoder was trained with 400 epochs and a sparsity proportion of 0.15.

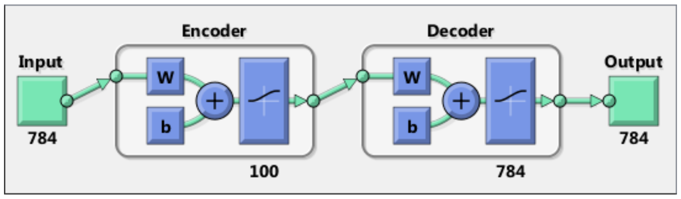


Figure. 5
The diagram of first the autoencoder

The second autoencoder was trained similarly but with 50 nodes in the hidden layer, 100 epochs and sparsity proportion of 0.10. The first autoencoder had 784 input and output nodes (28*28). The input for the second autoencoder was taken from the features determined by the first autoencoder's 100 node hidden layer.

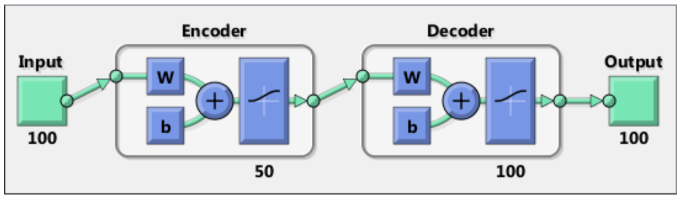


Figure. 6
The diagram of second autoencoder

The softMax layer was trained from the 50 features from second autoencoders hidden layer and trained for 400 epochs.

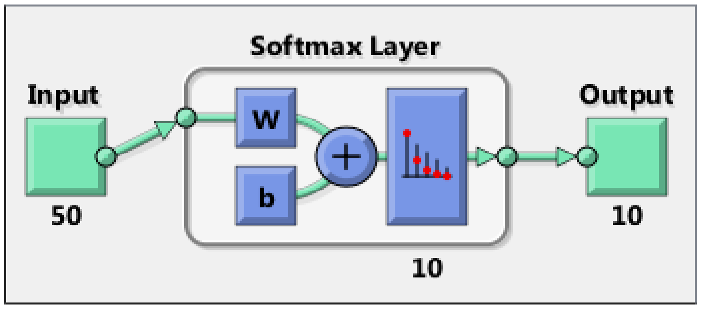


Figure. 7
The diagram of softmax Layer

Now the autoencoders and the softmax layer were stacked.

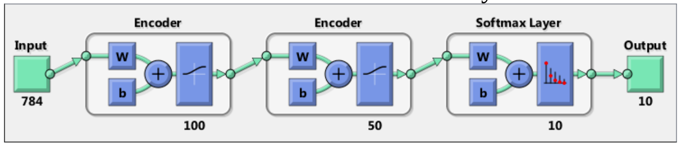
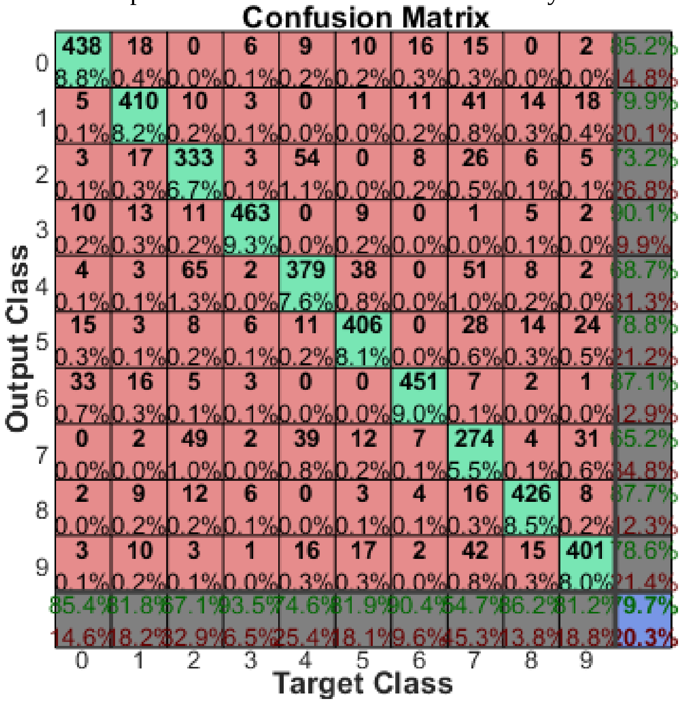


Figure. 8
diagram of the stacked network

VII. Results

The first pass of the test data showed an accuracy of 79.7%.



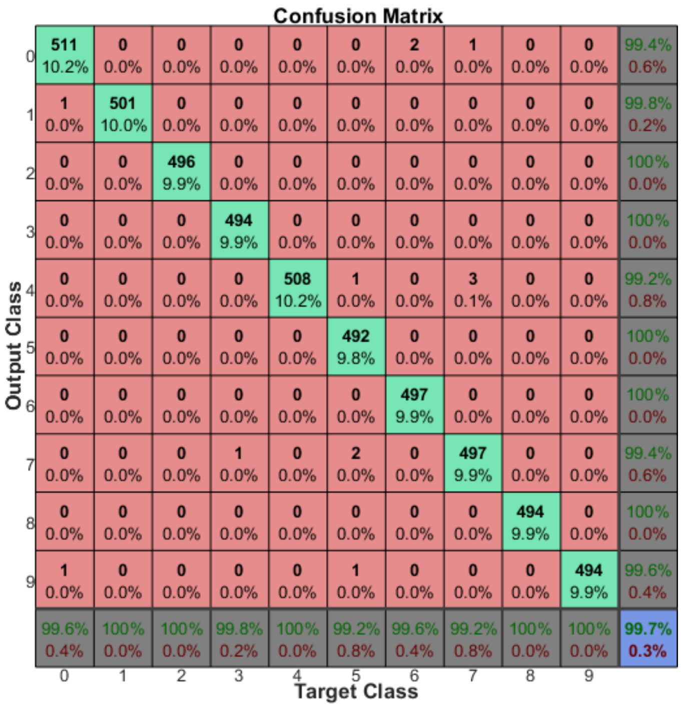
Confusion Matrix

Output Class / Target Class

	0	1	2	3	4	5	6	7	8	9	
0	438 8.8%	18 0.4%	0 0.0%	6 0.1%	9 0.2%	10 0.2%	16 0.3%	15 0.3%	0 0.0%	2 0.0%	85.2% 14.8%
1	5 0.1%	410 8.2%	10 0.2%	3 0.1%	0 0.0%	1 0.0%	11 0.2%	41 0.8%	14 0.3%	18 0.4%	79.9% 20.1%
2	3 0.1%	17 0.3%	333 6.7%	3 0.1%	54 1.1%	0 0.0%	8 0.2%	26 0.5%	6 0.1%	5 0.1%	73.2% 26.8%
3	10 0.2%	13 0.3%	11 0.2%	463 9.3%	0 0.0%	9 0.2%	0 0.0%	1 0.0%	5 0.1%	2 0.0%	90.1% 9.9%
4	4 0.1%	3 0.1%	65 1.3%	2 0.0%	379 7.6%	38 0.8%	0 0.0%	51 1.0%	8 0.2%	2 0.0%	68.7% 31.3%
5	15 0.3%	3 0.1%	8 0.2%	6 0.1%	11 0.2%	406 8.1%	0 0.0%	28 0.6%	14 0.3%	24 0.5%	78.8% 21.2%
6	33 0.7%	16 0.3%	5 0.1%	3 0.1%	0 0.0%	0 0.0%	451 9.0%	7 0.1%	2 0.0%	1 0.0%	87.1% 12.9%
7	0 0.0%	2 0.0%	49 1.0%	2 0.0%	39 0.8%	12 0.2%	7 0.1%	274 5.5%	4 0.1%	31 0.6%	65.2% 34.8%
8	2 0.0%	9 0.2%	12 0.2%	6 0.1%	0 0.0%	3 0.1%	4 0.1%	16 0.3%	426 8.5%	8 0.2%	87.7% 12.3%
9	3 0.1%	10 0.2%	3 0.1%	1 0.0%	16 0.3%	17 0.3%	2 0.0%	42 0.8%	15 0.3%	401 8.0%	78.6% 21.4%
	85.4% 14.6%	81.8% 18.2%	67.1% 32.9%	93.5% 6.5%	74.6% 25.4%	81.9% 18.1%	90.4% 9.6%	54.7% 45.3%	86.2% 13.8%	81.2% 18.8%	79.7% 20.3%

Figure. 9
Result with Confusion matrix before fine tuning

We then did backpropagation on the stacked autoencoder and obtained an accuracy of 99.7%.



Confusion Matrix

Output Class / Target Class

	0	1	2	3	4	5	6	7	8	9	
0	511 10.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	1 0.0%	0 0.0%	0 0.0%	99.4% 0.6%
1	1 0.0%	501 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.8% 0.2%
2	0 0.0%	0 0.0%	496 9.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	494 9.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	508 10.2%	1 0.0%	0 0.0%	3 0.1%	0 0.0%	0 0.0%	99.2% 0.8%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	492 9.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	497 9.9%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	2 0.0%	0 0.0%	497 9.9%	0 0.0%	0 0.0%	99.4% 0.6%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	494 9.9%	0 0.0%	100% 0.0%
9	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	494 9.9%	99.6% 0.4%
	99.6% 0.4%	100% 0.0%	100% 0.0%	99.8% 0.2%	100% 0.0%	99.2% 0.8%	99.6% 0.4%	99.2% 0.8%	100% 0.0%	100% 0.0%	99.7% 0.3%

Figure. 10
Final result with confusion matrix after fine tuning

VIII. Conclusion

We have shown that stacked autoencoders with a softMax classifier can return good results when applied to the MNIST digit database. Giving an accuracy of 99.7%.

References

[1] Bartosz Witkowski, “Autoencoders for image classification”, unpublished, 2013, retrieved 25 November 2016 from http://like-a-boss.net/files/autoencoders-for-image-classification.pdf

[2] http://like-a-boss.net/files/autoencoders-for-image-classification.pdf.

[3] Nikhil Buduma, “The Curse of Dimensionality and the Autocoder”, Blog 10 March 2015, retrieved 26 November 2016 from http://nikhilbuduma.com/2015/03/10/the-curse-of-dimensionality

[4] Wikipedia, “MNIST Database”, retieved 25 November 2015 from https://en.wikipedia.org/wiki/MNIST_database

[5] Liou, C.-Y., Huang, J.-C. and Yang, W.-C., “Modeling word perception using the Elman network”, Neurocomputing, Vol. 71, no. 16-18, pp 3150–3157, Oct 2008

[6] Andrew Ng, “CS294A Lecture notes – Sparse autoencoder”, unpublished, Retrieved 26 November 2015 from

https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

[7] Alirez Makhzani, and Brendan Frey, "k-sparse autoencoder", International Conference on Learning Representations, ICLR 2014, 2014, arXiv:1312.5663, retrieved 26 November 2016 from https://arxiv.org/pdf/1312.5663v2.pdf

[8] Alireza Makhzani, and Brendan Frey, Winner-Take-All Autoencoder, retrieved 10/18/2016 from https://pdfs.semanticscholar.org/4064/696e69b0268003879c0bcae6527d3b786b85.pdf

[9] Wikipedia, "Optical Character Recognition", Retrieved 25 November 2015, from https://en.wikipedia.org/wiki/Optical_character_recognition

[10] Julia Nomee, Avery Leider, Stephanie Haughton, Yahia Saeed, "Use Neural Networks to analyzi handwriting Format", 2016, unpublished.

[11] R. O. Duda, P. E. Hart, D. G. Stork, "Algorithm-Independent Machine Learning" in *Pattern Classification*, 2nd ed. Wiley 2000

[12] "Softmax Regression", retieved 29 November 2016 from http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression

[13] Y. LeCun, C. Cortes, C.Burges, "The MNIST Database of Handwritten Digits", retrieved 29 November 2016 from http://yann.lecun.com/exdb/mnist/

[14] Y. LeCun, L. Bottou, Y. Bengio, "Reading Checks with Multilayer Graph Transformer Networks", Acoustics, Speech, and Signal Processing, IEEE International Conference on vol. 01 no. undefined, p. 151-154, 1997, retrieved 2 December 2016 from http://yann.lecun.com/exdb/publis/pdf/lecun-bottou-bengio-97.pdf

[15] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE, 86(11):2278-2324*, November 1998, retrieved 2 December 2016 from http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

[16] S.Belongie, J. Malik, J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 24, pp 509-522 , 2002, retrieved 2 December 2016 from https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/belongie-pami02.pdf

[17] B. Kegl, R. Busa-Fekete, " Boosted products of base classifiers", Proceedings of the 26th International Conference on Machine Learning, 2009. Retrieved 2 Decenber 2016 from http://www.machinelearning.org/archive/icml2009/papers/231.pdf

[18] D. Decoste, B. Scholkopf, "Training Invariant Support Vector Mahines", Machine Learning 46, pp 161-190,2002, Retrieved 2 December 2016 from https://people.eecs.berkeley.edu/~malik/cs294/decoste-scholkopf.pdf

[19] D. C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber, "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition", 2010, retrieved 2 December 2016 from https://arxiv.org/pdf/1003.0358v1.pdf

[20] D. Ciresan, U. Meier, J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification", 2012, retrieved 2 December 2016 from https://arxiv.org/pdf/1202.2745v1.pdf