Neural Handwritten Recognition System

Federico Cardoni
federico.cardoni@city.ac.uk
City, University of London

Emanuele Cappella
emanuele.cappella@city.ac.uk
City, University of London

Abstract:

This article presents a comparison of two neural networks applied to a digit recognition task on data from the widely-used MNIST dataset. We contrasted the Feedforward Multilayer Perceptron (MLP), a supervised learning algorithm, with a Stacked sparse Autoencoder (SAE), a deep network unsupervised one. Hyperparameters were varied in the training phase to achieve the best possible performance for each model, whose classification rate was compared on a separate test sample using Confusion Matrices. The SAE algorithm proved slightly better.

1 - Introduction

There are many areas where we need to recognize the words, alphabets and digit. In fact, handwriting recognition is a very popular and challenging area of research, with many potential real-word applications, like postal addresses [1] and bank cheques or mobile app for decoding handwritten notes. The purpose of this paper is to develop a simple handwritten recognition system for numbers, based on models with the aim to learn patterns in a pixel based grid.
The models employed to perform this task are: Feed-Forward Neural Network and the Auto-Econcoder (AE) neural network. we are going to investigate various kind of configurations of these models to train a neural network being able to recognize human writing with a sufficient level of generalization.

1.1 Multi-Layer Perceptron

A feed-forward neural network is an artificial neural network where connections between units do not perform a cycle. They are called feed-forward because information only travels forward in the network, first through the input nodes, then through the hidden nodes and finally through the output nodes.
They are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent. In this case has been used a multi-layer perceptron able to learn non linearly separable functions, They are considered to be very versatile for both regression and classification *[2].*

1.2 Stacked Sparse Autoencoder:

A stacked autoencoder is a deep learning neural network consisting of multiple layers of sparse autoencoders [3], which are trained independently in a greedy layer-wise fashion [4].
Its basic unit is the autoencoder, a feed-forward, non-recurrent neural network. The autoencoder is an unsupervised learning model, whose is to reconstruct its own inputs [5]. This operation is conducted by two parts of the autoencoder: the encoder and the decoder. The **encoder** compresses the input into a smaller set of features. The **decoder** takes this

compressed representation and from that reconstructs the original input, as figure 1 shows. An autoencoder is defined sparse when its neurons are highly specialized and activate only for a small range of features from the input values [6, 7]. In this case, most of their weights will be 0 and thus the matrix representing their layer will be ‘sparse’. This result is achieved by adding to the loss function a **sparsity** constraint.

2 - Data Description

Our dataset is the widely-known MNIST, featuring a large collection of handwritten numbers. A set of numbers from 0 to 9 are stored in a 31500x784 matrix, each image is colorimetry scaled by pixel intensity from 0 to 255, where 0 is white and 255 is max intensity of whatever color scale you adopted. Handwritten numbers are from 0 to 9, every number is written in a different way as shown in figure 2.

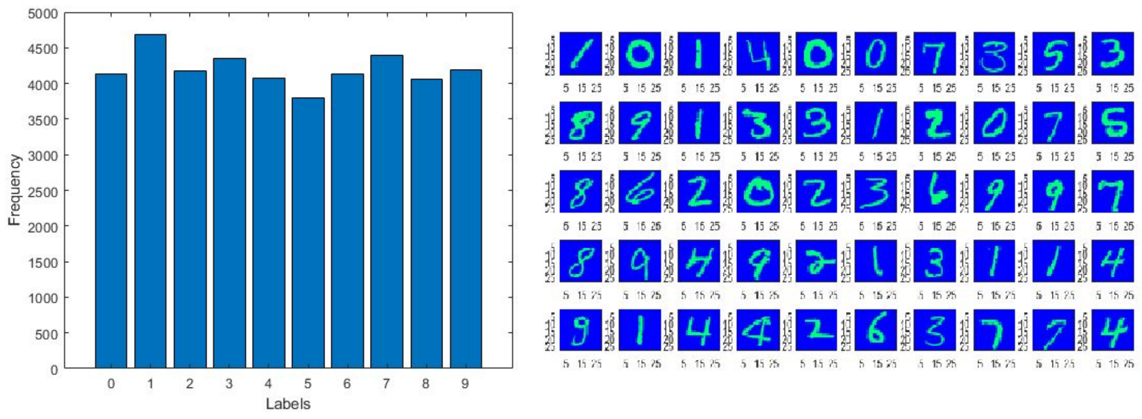


Figure 1: Handwritten Image and Label Frequency

For each number we have a certain amount of different writings, barplot in figure 1 shows that the number of samples is more or less the same.

3 - Methods

The Methodology we applied consist in separate model evaluation and algorithm’s performance comparison in two different phases using different chunks of the original dataset. In order to do this we applied what the theory call “Holdout” method, consisting in create 3 different datasets one for training (90%), the remaining is for testing (10%) used for algorithm comparison.

Model selection was done using grid-search to adjust hyperparameters of both the Multi Layer Perceptron Network (MPL) and the Stacked AutoEncoder (SAE). Training process and validation of both neural networks were done by partitioning the training-dataset again into training, validation and testing. This decision was taken according to papers who dealt with a very similar problem [8]. Moreover, due to the high demands in terms of computational power of both networks, a simple method was preferred to a more demanding one like k-fold cross validation. After training the network, test results were took into account as performance meter for grid-searching the best model.

For the **network comparison,** both the algorithms, once trained were tested on the test data, which correspond to another dataset of 10’000 entries. For the MPL network has been adopted a “validation check” - early stopping criteria - in order to increase the number of epochs. By giving more time to the network to train will hopefully increase its performance.

3.1 - Architecture of Multi-Layer Perceptron Network (MLP)

Carter obtained one of the best performance on MNIST dataset with a multi-layer perceptron feed-forward neural network [9]. He used 130 neurons, with scaled conjugate gradient as training function, a single hidden layer with step function and a cross-entropy performance function. In the present paper it was tried to use a modified version of a standard feed-forward network, with the following architecture:

- 1 hidden layer with **sigmoid** activation function and variable number of neurons;
- 1 output layer with **softmax** activation function for classification;
- MSE (Mean Square Error) and Cross-Entropy loss functions will be compared.

For the MLP network it was performed a **double grid-search** between two different training functions: **scaled conjugate gradient** as chosen by [10] and **Resilient Backpropagation**. Different hidden layer size was also took into consideration.
A sub-grid search here was performed to choose the best configuration for resilient backprop. by tweaking the learning rate in order to avoid to be stuck in a local minimum. The other parameters' values were left by default: 6 validation checks, 1000 epochs, best performance to 0.

3.2 - Architecture of Stacked Auto Encoder (SAE)

Our SAE consisted of two sparse autoencoders and a softmax classifier (figure 2).
(1) **First Autoencoder***.* The first layer receives as input a series of 28x28 matrices. Each matrix contains the pixel structure of the MNIST numbers, for a total of 784 pixels. The size of the hidden layer is 150, which means that in this step the encoder is trying to describe with 150 features the MNIST images. The decoder takes this 'compressed' version of the input and tries to recreate the original input.
(2) **Second Autoencoder***.* The second autoencoders starts from the 150 primary features of the first step, and its encoder tries to further shrink them to 50 secondary features (Xu et al., 2015). Afterwards, the decoder replicates from the 50 secondary features the compressed representation of data based on the original 150 features.
(3) **Softmax layer***.* This final layer is the only one using label data to classify the 50 features representation of data into different 10 digit categories - that is the original numbers. The three layers are combined together to form a stacked autoencoder, which is finally fine-tuned using back-propagation.

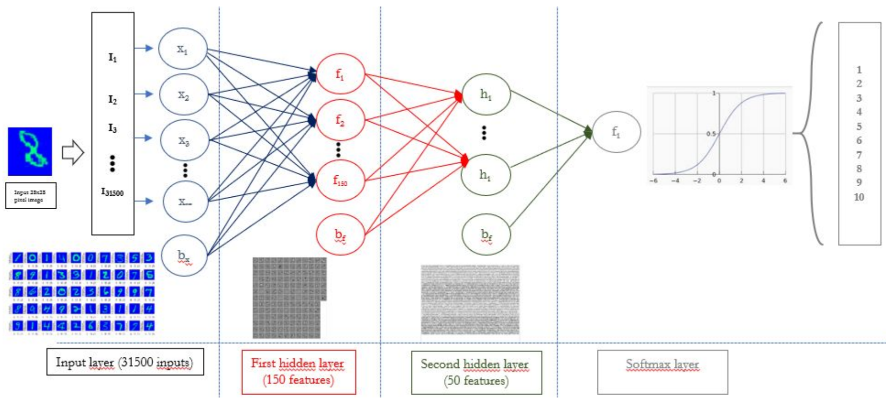
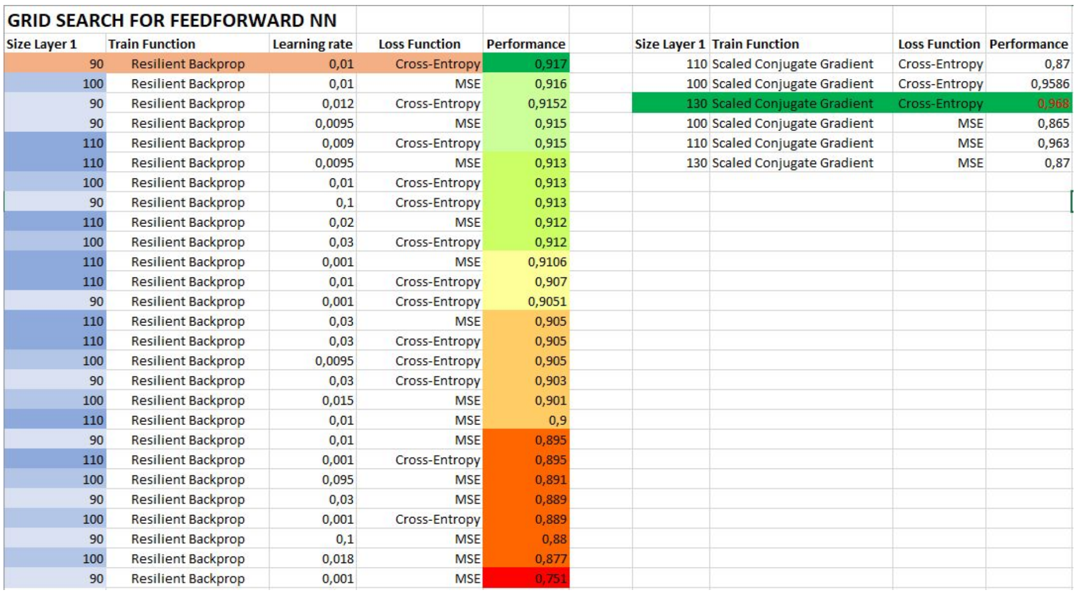


Figure 2: Stacked Autoencoder (SAE)

4 - Results and Comparison

4.1 - Model Selection for MLP

Table 1 shows the Hyperparameters search grid for the Multi Layer Perceptron and Table 2 for the SAE. The performances were estimated using a cross-validation method, comparing the predicted labels obtained by simulate the network and the test data labels. In the process of selecting the best MLP it was noticed that changing hyperparameters and neurons with resilient backpropagation as training function, didn't produced a noticeable effect over performance ratio. The average fluctuation is more or less 0.05. On the contrary, using Scaled Conjugate Gradient produced excellent results, especially by increasing the number of neurons. <u>The best result was produced with Cross-Entropy as loss function, 130 neurons in one hidden layer and Scaled Conjugate Gradient as training function.</u>



GRID SEARCH FOR FEEDFORWARD NN

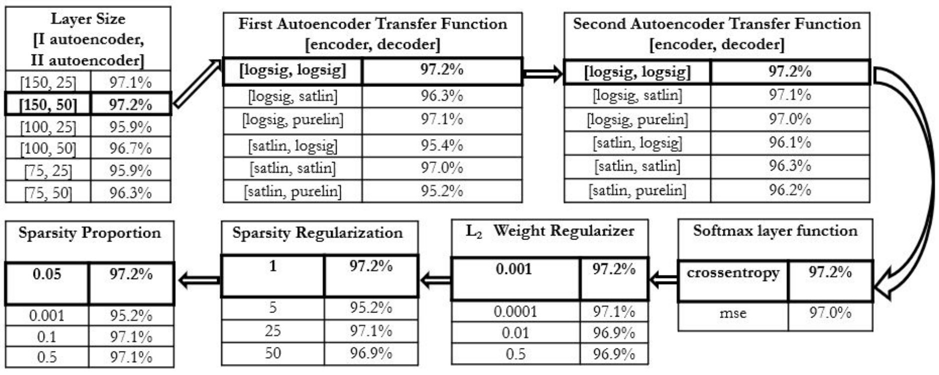
Size Layer 1	Train Function	Learning rate	Loss Function	Performance
90	Resilient Backprop	0,01	Cross-Entropy	0,917
100	Resilient Backprop	0,01	MSE	0,916
90	Resilient Backprop	0,012	Cross-Entropy	0,9152
90	Resilient Backprop	0,0095	MSE	0,915
110	Resilient Backprop	0,009	Cross-Entropy	0,915
110	Resilient Backprop	0,0095	MSE	0,913
100	Resilient Backprop	0,01	Cross-Entropy	0,913
90	Resilient Backprop	0,1	Cross-Entropy	0,913
110	Resilient Backprop	0,02	MSE	0,912
100	Resilient Backprop	0,03	Cross-Entropy	0,912
110	Resilient Backprop	0,001	MSE	0,9106
110	Resilient Backprop	0,01	Cross-Entropy	0,907
90	Resilient Backprop	0,001	Cross-Entropy	0,9051
110	Resilient Backprop	0,03	MSE	0,905
110	Resilient Backprop	0,03	Cross-Entropy	0,905
100	Resilient Backprop	0,0095	Cross-Entropy	0,905
90	Resilient Backprop	0,03	Cross-Entropy	0,903
100	Resilient Backprop	0,015	MSE	0,901
110	Resilient Backprop	0,01	MSE	0,9
90	Resilient Backprop	0,01	MSE	0,895
110	Resilient Backprop	0,001	Cross-Entropy	0,895
100	Resilient Backprop	0,095	MSE	0,891
90	Resilient Backprop	0,03	MSE	0,889
100	Resilient Backprop	0,001	Cross-Entropy	0,889
90	Resilient Backprop	0,1	MSE	0,88
100	Resilient Backprop	0,018	MSE	0,877
90	Resilient Backprop	0,001	MSE	0,751

Size Layer 1	Train Function	Loss Function	Performance
110	Scaled Conjugate Gradient	Cross-Entropy	0,87
100	Scaled Conjugate Gradient	Cross-Entropy	0,9586
130	Scaled Conjugate Gradient	Cross-Entropy	0,968
100	Scaled Conjugate Gradient	MSE	0,865
110	Scaled Conjugate Gradient	MSE	0,963
130	Scaled Conjugate Gradient	MSE	0,87

Table 1 : Grid-Search for Multi-Layer Perceptron

4.2 - Model Selection for SAE

Table 2 shows the sequential order we used to train our SAE model. As baseline model, we kept all the default options provided by MATLAB. We then varied one parameter at a time. The model with the best performance had the first autoencoder with Logistic Sigmoid function ('logsig') as transfer function and a hidden layer size of 150; the second one with



Layer Size [I autoencoder, II autoencoder]	
[150, 25]	97.1%
[150, 50]	**97.2%**
[100, 25]	95.9%
[100, 50]	96.7%
[75, 25]	95.9%
[75, 50]	96.3%

First Autoencoder Transfer Function [encoder, decoder]	
[logsig, logsig]	**97.2%**
[logsig, satlin]	96.3%
[logsig, purelin]	97.1%
[satlin, logsig]	95.4%
[satlin, satlin]	97.0%
[satlin, purelin]	95.2%

Second Autoencoder Transfer Function [encoder, decoder]	
[logsig, logsig]	**97.2%**
[logsig, satlin]	97.1%
[logsig, purelin]	97.0%
[satlin, logsig]	96.1%
[satlin, satlin]	96.3%
[satlin, purelin]	96.2%

Softmax layer function	
crossentropy	**97.2%**
mse	97.0%

L_2 Weight Regularizer	
0.001	**97.2%**
0.0001	97.1%
0.01	96.9%
0.5	96.9%

Sparsity Regularization	
1	**97.2%**
5	95.2%
25	97.1%
50	96.9%

Sparsity Proportion	
0.05	**97.2%**
0.001	95.2%
0.1	97.1%
0.5	97.1%

Table 2 : Grid-Search for SAE

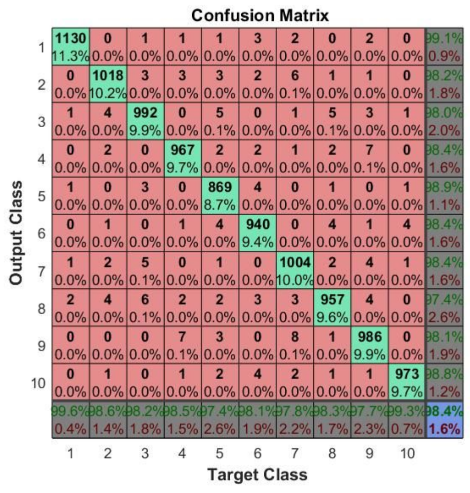
logsig and 50 hidden neurons; the softmax layer used the cross-entropy function. The best values for the L_2 sparsity regularizer was 0.001; 1 for the sparsity regularization term and 0.05 for the sparsity proportion parameter. The final model achieved a performance of 97.2% after fine-tuning. The improvement of performance is mostly attributable at the increase of the hidden layers' size.

4.3 - Algorithm Comparison

The difference in performance of the two algorithms was minimal: SAE reached 97.2% and MLP 96.8% with the first test set. The difference was therefore minimal, a mere 0.4%. As previously reported, we performed a second comparison on a separate test dataset of 10k observations, and plotted their results in a confusion matrix to compare them. This second test set is bigger than the first one, with 1600 more observations. We shuffled the training and the algorithm comparison test data to make the estimation more reliable. In this case we expected a slightly higher accuracy rate for both models and a widening performance gap in favour to SAE. Indeed, by looking at the confusion matrix in figure [3] the performance advantage of SAE rose to 1%: 1.6% (SAE) vs 2.6 (MLP) Still, the difference between models remained somewhat inferior to what we foresaw.

As for digit recognition, both algorithms tend to recognize quite efficiently numbers from 1 to 6, MLP lost accuracy when dealing with the interval between 7-10, especially within 8 and 9, this could probably be due to the fact that both numbers are quite similar. Other relevant misclassification for the MLP are within 4 and 9, 7 and 9. The difficulty in classifying 7 and 9 is shared with the SAE.

Considering the overall performance, SAE obtained a better result, even if it is considered an unsupervised deep neural network MLP with a single hidden layer, keep its pace in an impressive way.



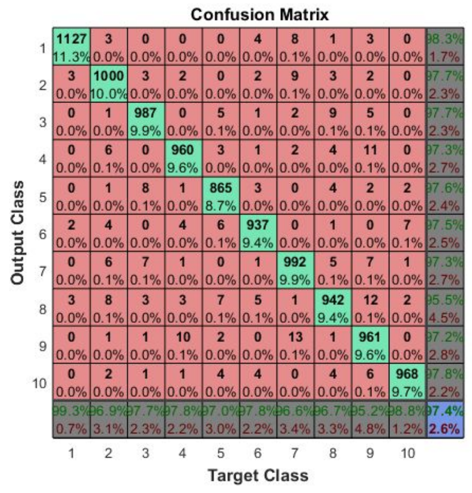


Figure (3) : Confusion matrices for SAE(left) and MLP(right)

5 - Conclusion

This study compared the performances of a Stacked Autoencoder and a Multilayer Perceptron for a classic task of digit recognition.

Though the two models are based on opposite approaches, where the MLP is a supervised algorithm and the SAE is unsupervised, their performances diverged by a mere 1% on test

data. This result run contrary to our expectations, because we thought the SAE results would have been higher than the MLP, a much simpler model.
We believe that part of the problem may lie in the suboptimal way which the SAE was trained with. For a future improvement, we would opt for a more systematic way to vary the parameters. Moreover, the SAE resulted very sensitive to the size of its hidden layer, but not to the change in its other parameters. Thus, another possible improvement would be to increase further the number of hidden neurons to increase performance.
On the other hand, the improvement in performance registered for the MLP was mainly attributable to the choice of the transfer function, while changing the other parameters did not achieve much. Therefore, a possible improvement would be to dedicate more time to the variation of training functions and less to parameter testing.
Another big difference between the two models consists in the difference in training time, where the SAE resulted computationally much heavier, a particular that prevented us to verify more combinations of parameters.
The classification matrix was a very useful visualization both for model comparison and for understanding the difficulties our algorithms encountered in the classification of digits, where the number 9 resulted the most difficult to recognize.

6 - References

[1] Xu, Jun, et al. "Stacked Sparse Autoencoder (SSAE) based framework for nuclei patch classification on breast cancer histopathology." *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*. IEEE, 2014.
[2] Brilliant team, 2016 (page revised: 2018), Brilliant, , viewed 21 February 2018, <https://brilliant.org/wiki/feedforward-neural-networks/>.
[3] Oliveira, Tiago Prado, Jamil Salem Barbar, and Alexsandro Santos Soares. "Multilayer perceptron and stacked autoencoder for Internet traffic prediction." *IFIP International Conference on Network and Parallel Computing*. Springer, Berlin, Heidelberg, 2014.
[4] Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of Machine Learning Research* 11.Dec (2010): 3371-3408.
[5] Netzer, Yuval, et al. "Reading digits in natural images with unsupervised feature learning." *NIPS workshop on deep learning and unsupervised feature learning*. Vol. 2011. No. 2. 2011.
[6] Ng, Andrew, and Sparse Autoencoder. "CS294A Lecture notes." *Dosegljivo: https://web. stanford. edu/class/cs294a/sparseAutoencoder_2011new. pdf. [Dostopano 20. 7. 2016]* (2011).
[7] Arpit, Devansh, et al. "Why regularized auto-encoders learn sparse representation?." *International Conference on Machine Learning*. 2016
[8] V. Musoko, M. Kol´ınov´a, A. Prochazka, "*IMAGE CLASSIFICATION USING COMPETITIVE NEURAL NETWORKS*" - Institute of Chemical Technology, Prague - 27-Sep-2004.
[9] Carter W. Blum "*On the Effectiveness of Neural Networks Classifying the MNIST Dataset*", University of Minnesota Digital Conservancy (UDC), March 2017.
[10] Moller, M. F. "*A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning*", *Neural Networks*, Vol. 6, 1993, pp. 525–533.