Multilayer perceptron (MLP)

Amir.M Mousavi.H

Department of Computer Engineering
Shahid Rajee University

Tehran, Iran

AmirMahmood.Mousavi@yahoo.com

Abstract **- Since neural networks have universal approximation capabilities, therefore it is possible to postulate them as solutions for given differential equations that define unsupervised errors. Artificial neural networks are appearing as useful alternatives to traditional statistical modelling techniques in many scientific disciplines. This paper presents a general introduction and discussion of how a multilayer perceptron (MPL) works. then we show the implementation result of a simple multilayer perceptron (MPL) on Artificial data and then analyze the network's behavior in presence of small dataset and overlapped data..**

Keywords— perceptron, Neural Network, Multilayer perceptron, MLP, Classification, Artificial Data

I. INTRODUCTION

Artificial neural networks are a fascinating area of study, although they can be intimidating when just getting started. There are a lot of specialized terminology used when describing the data structures and algorithms used in the field. The field of artificial neural networks is often just called neural networks or multi-layer perceptrons after perhaps the most useful type of neural network. A perceptron is a single neuron model that was a precursor to larger neural networks.
It is a field that investigates how simple models of biological brains can be used to solve difficult computational tasks like the predictive modeling tasks we see in machine learning. The goal is not to create realistic models of the brain, but instead to develop robust algorithms and data structures that we can use to model difficult problems.
The power of neural networks come from their ability to learn the representation in your training data and how to best relate it to the output variable that you want to predict. In this sense neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm. The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example from lines, to collections of lines to shapes.
In the previous homework (HW1) we explained a perceptron neuron, in MLP Neurons are arranged into networks of neurons. A row of neurons is called a layer and one network can have multiple layers. The architecture of the neurons in the network is often called the network topology

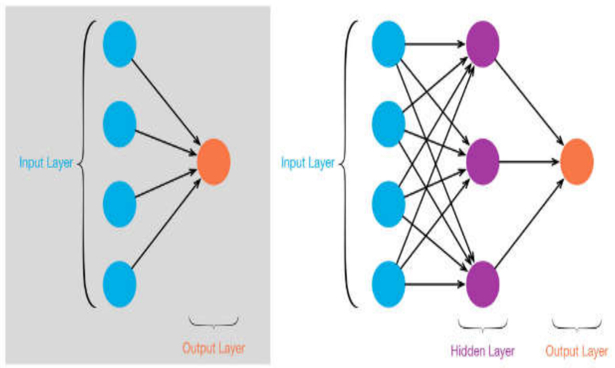


Figure.1. Left: Single-Layer Perceptron; Right: Perceptron with Hidden Layer

A network includes 3 part :

1. Input or Visible Layers

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset. These are not neurons as described above, but simply pass the input value though to the next layer.

2. Hidden Layers

Layers after the input layer are called hidden layers because that are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value.Given increases in computing power and efficient libraries, very deep neural networks can be constructed. Deep learning can refer to having many hidden layers in your neural network. They are deep because they would have been unimaginably slow to train historically, but may take seconds or minutes to train using modern techniques and hardware.

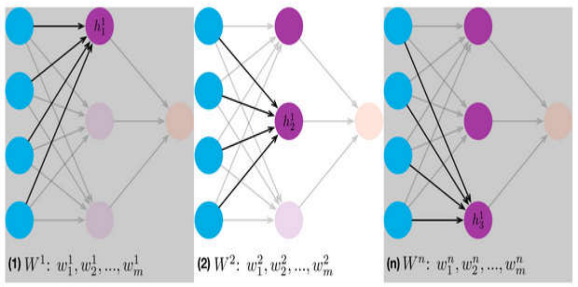

h_1^1
h_2^1
h_3^1
(1) $W^1: w_1^1, w_2^1, ..., w_m^1$
(2) $W^2: w_1^2, w_2^2, ..., w_m^2$
(n) $W^n: w_1^n, w_2^n, ..., w_m^n$

Figure.2. Procedure to Hidden Layer Outputs

3. Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem.

The choice of activation function in he output layer is strongly constrained by the type of problem that you are modeling. For example: A regression problem may have a single output neuron and the neuron may have no activation function.
A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the class 1. This can be turned into a crisp class value by using a threshold of 0.5 and snap values less than the threshold to 0 otherwise to 1.
A multi-class classification problem may have multiple neurons in the output layer, one for each class (e.g. three neurons for the three classes in the famous iris flowers classification problem). In this case a softmax activation function may be used to output a probability of the network predicting each of the class values. Selecting the output with the highest probability can be used to produce a crisp class classification value.

II. METHODS & MATERIALS

In order to implement a multilayer perceptron, we used Keras in Tensorflow framework of python.
On the other hand to train the network, we generated artificial data, a random 2 dimensional array coming from a guassian distribution (for the first class(+1) with mean = [6, 0] cov = [[4, 0], [0, 2]] and and for second class (0) with mean = [-5,0] , cov = [[3, 0], [0, 2]]). For each class we generated 10000 sample. In figure.1 is shown :

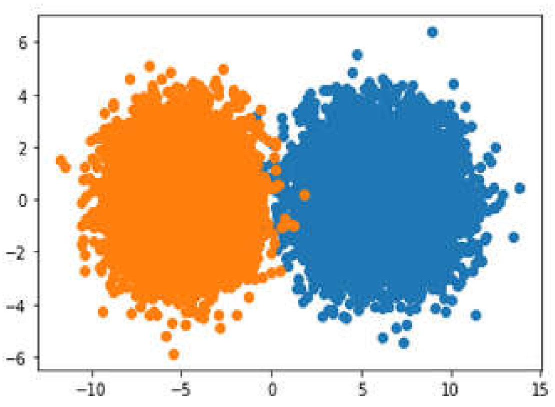
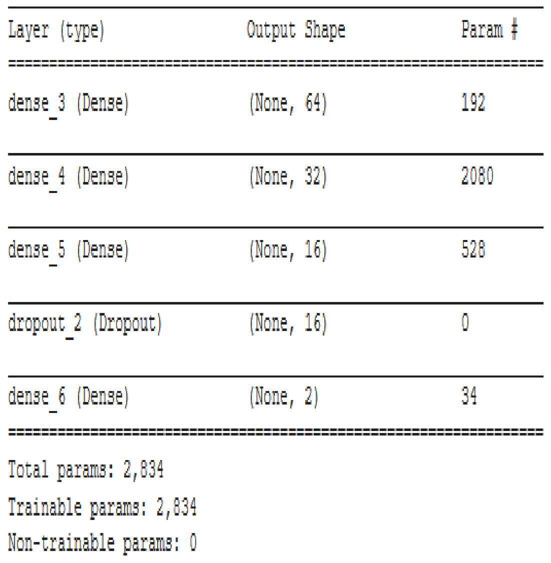


Figure.1 data distribution

For the hyperparameters we set:

- learning rate=0.2
- epoch numbers: 30
- batch size=64
- optimizer='rmsprop'
- validation split=0.2

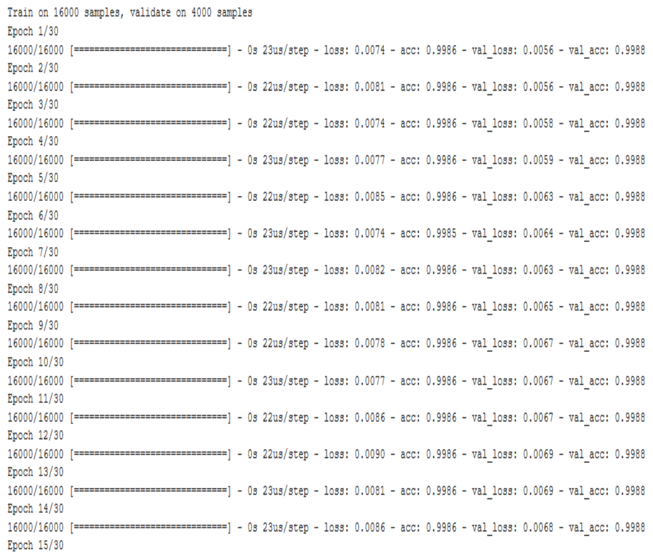
Our network include 2 input neurons, 3 hidden layers (64,32,16 neurons in each layer), 2 neuron in output layer. We also used Dropout technique in the network in order to prevent overfitting. Our model consist of 2,834 trainable parameter. in figure.2. a summary of the model is shown:



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 64)                192

dense_4 (Dense)              (None, 32)                2080

dense_5 (Dense)              (None, 16)                528

dropout_2 (Dropout)          (None, 16)                0

dense_6 (Dense)              (None, 2)                 34
=================================================================
Total params: 2,834
Trainable params: 2,834
Non-trainable params: 0
```

Figure.2. summary of model

In figure.3. is shown training procedure of network.



```
Train on 16000 samples, validate on 4000 samples
Epoch 1/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0074 - acc: 0.9986 - val_loss: 0.0056 - val_acc: 0.9988
Epoch 2/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0081 - acc: 0.9986 - val_loss: 0.0056 - val_acc: 0.9988
Epoch 3/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0074 - acc: 0.9986 - val_loss: 0.0058 - val_acc: 0.9988
Epoch 4/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0077 - acc: 0.9986 - val_loss: 0.0059 - val_acc: 0.9988
Epoch 5/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0085 - acc: 0.9986 - val_loss: 0.0063 - val_acc: 0.9988
Epoch 6/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0074 - acc: 0.9985 - val_loss: 0.0064 - val_acc: 0.9988
Epoch 7/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0082 - acc: 0.9986 - val_loss: 0.0063 - val_acc: 0.9988
Epoch 8/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0081 - acc: 0.9986 - val_loss: 0.0065 - val_acc: 0.9988
Epoch 9/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0078 - acc: 0.9986 - val_loss: 0.0067 - val_acc: 0.9988
Epoch 10/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0077 - acc: 0.9986 - val_loss: 0.0067 - val_acc: 0.9988
Epoch 11/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0086 - acc: 0.9986 - val_loss: 0.0067 - val_acc: 0.9988
Epoch 12/30
16000/16000 [==============================] - 0s 22us/step - loss: 0.0090 - acc: 0.9986 - val_loss: 0.0069 - val_acc: 0.9988
Epoch 13/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0081 - acc: 0.9986 - val_loss: 0.0069 - val_acc: 0.9988
Epoch 14/30
16000/16000 [==============================] - 0s 23us/step - loss: 0.0086 - acc: 0.9986 - val_loss: 0.0068 - val_acc: 0.9988
Epoch 15/30
```

Figure.3. training procedure of network

For testset, we also generated artificial data, a random 2 dimensional array coming from a guassian distribution (for the first class(+1) with mean = [6, 0] cov = [[4, 0], [0, 2]] and and for second class (0) with mean = [-5,0] , cov = [[3, 0], [0, 2]]). For each class we generated 8000 sample. In figure.2 is shown

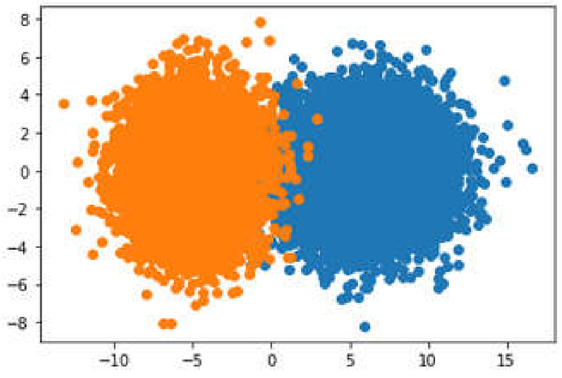


Figure.1 data distribution

III. EXPERIMENTAL RESULTS

our results have shown that a multilayer perceptron can classify a non-linear problem with a high accuracy.

Figure.3. accuracy and loss of model

For further experiment we increased the overlap of our data and also changed the amount of data. (figures.4-6)

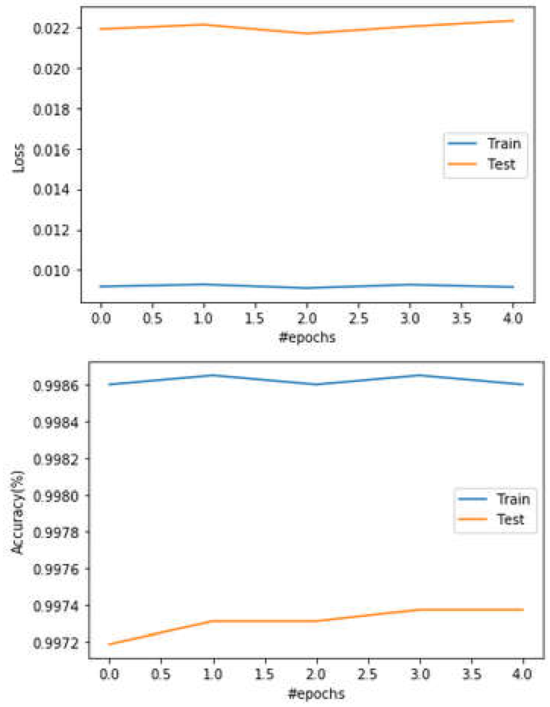


Figure.4. with 20000 training samples and 16000 test samples in 5 epoch 10% overlapping data

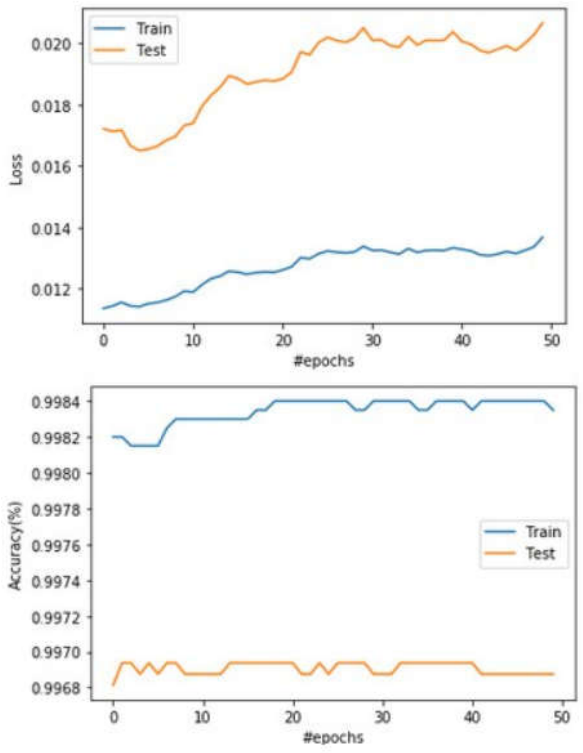


Figure.5. with 20000 training samples and 16000 test samples in 50 epoch 30% overlapping data

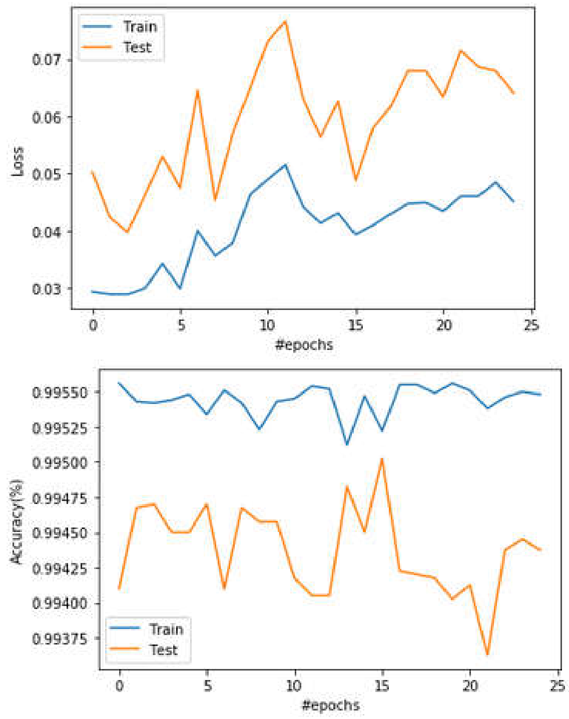
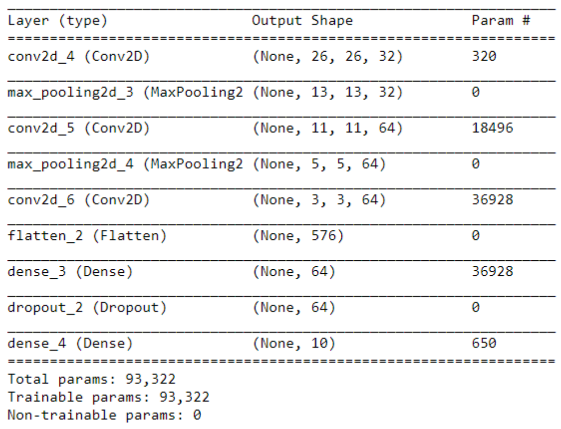


Figure.6. with 120000 training samples and 80000 test samples in 25 epoch 30% overlapping data

As we can observe because of simple data set and also lack of features (only 2 dimension), the network can reach up on the 99% performance, on the other if we increase the epoch because of simplicity and density of the generated data, it will goes toward overfitting (figure.5.) and we believe that is the reason why there is a fluctuation in small range (figure.6.) output. Also we observed that overlapping between data, is a problem that our current architecture can not perform in a best way. We believe that complexity of problems is more than complexity of network.

For further experiment we changed the network architecture and made it deeper, we also changed the amount of data, in order to test whether increasing amount of data can effect the performance or not. (figures.7-)



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_2 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                36928
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

Figure.7. new architecture of model

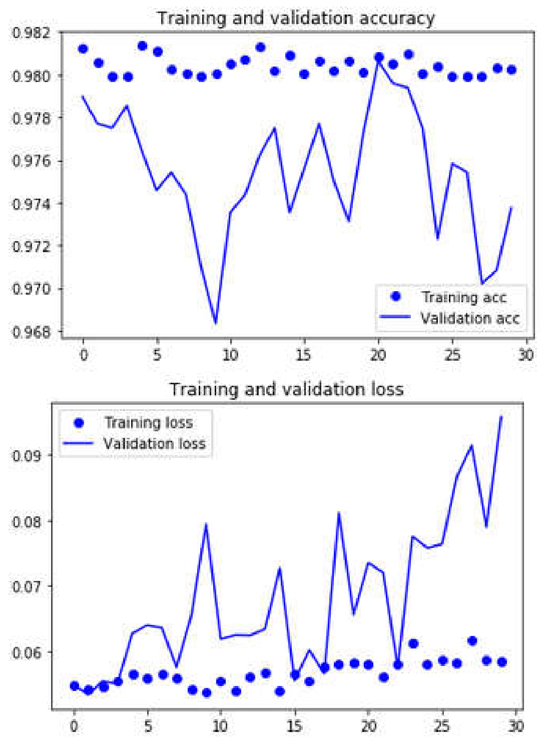


Figure.8. with 40000 training samples and 2000 test samples in 30 epoch 30% overlapping data

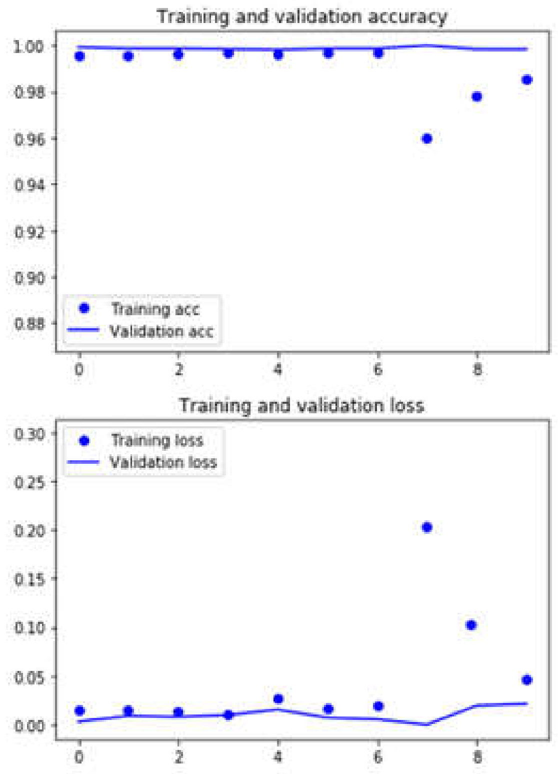


Figure.9. with 1920000 training samples and 48000 test samples in 10 epoch 30% overlapping data

As we expected with increasing data and also deeper network our performance has been improved.
For last experiment we analyzed how different size of training sample will effect on performance. And as shown in figure.10 it will increase by increasing data set.
We also tested different batch sizes, our result wasn't show a significant difference.

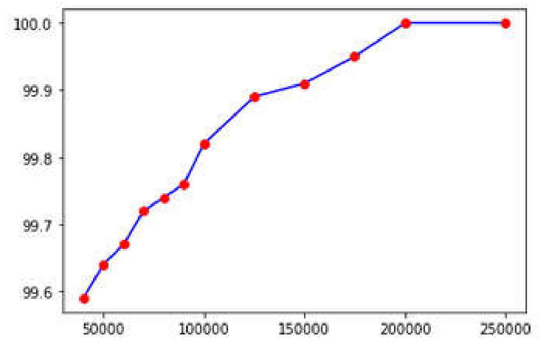


Figure.8. different size of training samples vs performance

IV. CONCLUSION

In the discipline of machine learning, classification is a quickly moving field. Its growth has been fueled by technological advances in digital imaging, computer processors and mass storage devices. In this paper an attempt is made to a famous classifier which is called multilayer perceptron (MLP). Here we tested how MLP works and how the amount of data will effect on performance. We observe that as much as our data amount grows our performance will increase and our perceptron can achieve to the optimal point. Also it was tested overlapped data, and observed with a complex network and a large amount of data we can overcome the problem with high performance. For future work we can focus on a complex data and different and deeper networks.

REFERENCES

[1] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.

[2] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[3] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[4] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), 303–314.

[5] Haykin, Simon (1998). Neural Networks: A Comprehensive Foundation (2 ed.). Prentice Hall. ISBN 0-13-273350-1.

[6] Neural networks. II. What are they and why is everybody so interested in them now?; Wasserman, P.D.; Schwartz, T.; Page(s): 10-15; IEEE Expert, 1988, Volume 3, Issue 1

[7] R. Collobert and S. Bengio (2004). Links between Perceptrons, MLPs and SVMs. Proc. Int'l Conf. on Machine Learning (ICML).

[8] Tensorflow.com