Comparison between CNN and MLP: Application on Fashion-MNIST

Pengyu Zhang(UFID:94598565)
Dept. Electrical and Computer Engineering
University of Florida
pengyu.zhang@ufl.edu

Zihan Liao(UFID:19613125)
Dept. Electrical and Computer Engineering
University of Florida
liaozihan1@ufl.edu

***Abstract*—We trained the Convolutional Neural Network(MLP) and Multi-Layer Perceptron(MLP) on the Fashion-MNIST image recognition task. From the perspective of classification accuracy, we discussed about the influence of a variety of hyperparameters on the performance of both two architectures. To reduce the probability of encountering overfitting in the MLP network, we introduced stopping criterion by validating the network after each training epoch. The accuracy of classification achieved 88% on MLP and 91.32% on CNN. We also discussed about the advantages and weak points of the two structure based on the accuracy and performance during the training step.**

***Index Terms*—Neural Network, MLP, CNN**

I. Introduction

Neural Network has been widely used both in academic and industry. The Multi-Layer Perceptron(MLP) is one of the earliest architecture that has been proved to have the ability to recognize different images. The design of Convolutional Neural Network(CNN) architecture showed significant improvements specifically on image recognition tasks. One of the representative work is the AlexNet[1] published on September 30, 2012. The well-designed convolutional network achieved a error rate of 15.3%, more than 10.8% lower than results obtained by other approaches. The AlexNet carefully discussed about how essential the depth of convolutional layer is for the high performances, and deployed the network on GPU to accelerate the computation process. The MNIST[2] benchmark proposed for the character recognition competition was once very popular. Due to its relatively lower complexity of images, we therefore chose Fashion-MNIST[3], a more challenged dataset, to apply our CNN and MLP on.

In this project, we set up a variety of choices on hyperparameters to explore how different architectures, learning rates affect the final classification performance. All the materials and analysis of part A is finished by Pengyu Zhang. All the materials and analysis of part B is finished by Zihan Liao.

II. Experiments Design

A. MLP

- The two hidden layer MLP is chosen as the basic architecture. The optimizer we chose is Adam[3]
- Varying the value patience of early stopping criteria.
- Varying the learning rate of Adam while fixing number epochs and architecture
- Varying the number of epochs while fixing other configurations
- Varying the number of nodes in the 1^{st} layer, and 2^{nd} layer respectively.

B. CNN

- Feature extraction by the convolution layers
- Experiment on how learning rate affects the performance of the model
- Experiment on the early stopping method
- Experiment on how number of layers and PE's affects the performance of the model

III. Corresponding Results

A. Multiple Layer Perceptron

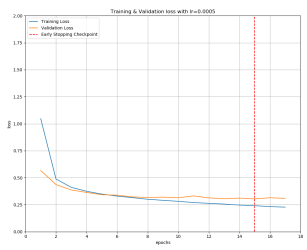
1) Early Stopping: The early stopping strategy is implemented during the training procedure. It is a form of regularization to avoid overtraining by dynamically comparing the validation loss with the training loss. The "Patience" parameter in the stopping method indices how long to wait after last time validation loss increased. The results are shown in Fig.1.

All of the results are based on setting learning rate as 0.005, batch size as 100 and epochs as 50. It's clear to see that the validation loss begins incresing after a certain number of epochs, and the stopping strategy helps to get rid of overfitting and store the model parameters at the checking point(red dotted line). Table I shows the accuracy on test data obtained when setting different value of patience, which also verifies the importance of determining when we should stop training.

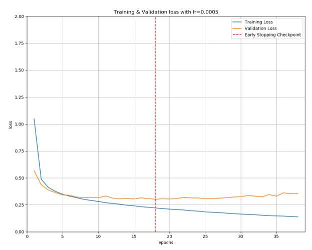
TABLE I: Accuracy

Patience	Accuracy
2	88%
20	88%
40	88%
80	88%

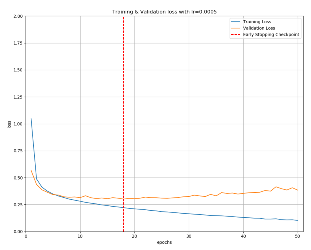
2) Learning rate: Based on the well-known Adam optimizer, 100 as the value of batch size and 128 to 128 as the two hidden layers, we initialized it with four different learning rate: 0.0001, 0.0005, 0.001, 0.01. We depicted four confusion matrix in Fig.2. while applying the stopping criteria, too. With the help of early stopping, we guaranteed well performance



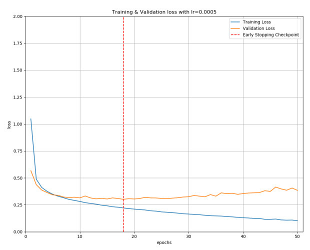
(a) Patience:2



(b) Patience:20

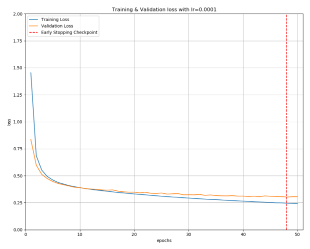


(c) Patience:40

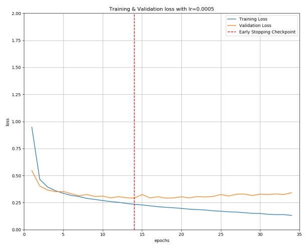


(d) Patience:80

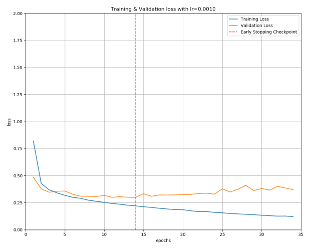
Fig. 1: Stopping training



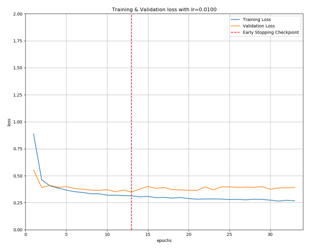
(a) Lr=0.0001



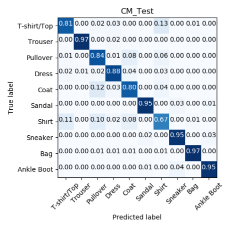
(b) Lr=0.0005



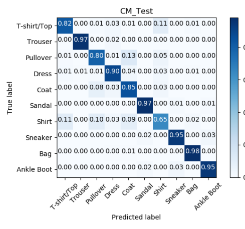
(c) Lr=0.001



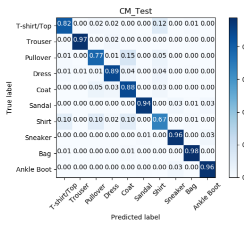
(d) Lr=0.01



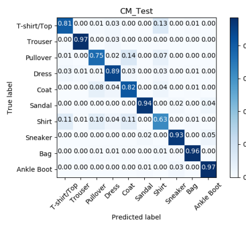
(e) ACC=86%



(f) ACC=88%



(g) ACC=88%



(h) ACC=86%

Fig. 2: Learning rate & Confusion Matrix

on test data with around 88% accuracy. We compared the location of the stopping points, found that the MLP model took approximate 50 epochs to meet its convergent point. If we increase the learning rate from 0.0001 to 0.0005, it only took no more than 20 epochs to achieve the ideal model. As we kept increasing the learning rate, we didn't obtain better results. Instead, we observed a more fluctuated validation loss curve, which can be explained as missing the local optimal because of unappropriated learning rate. Thus we decided to take 0.0005 as the best choice of learning rate.

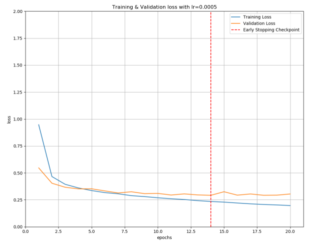
3) Influence of Epochs: We fixed the same architecture of two hidden layers(128 to 128) with 0.0005 as learning rate, then increasing the epochs during training procedure. The early stopping works fine to guarantee the final performance of different models.It's crystal clear to see from Fig.3. that the trade-off between training loss and validation loss is 14 epochs. Keeping increasing the epochs will not return a better model, but only gradually cause serious overfitting.

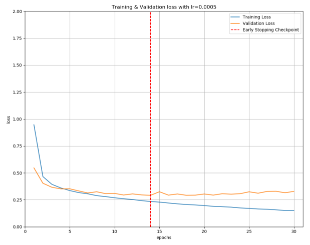
4) Variety of nodes: To explore how the number of nodes in different layer affect the model, We fixed one hidden layer with 128 nodes, while only changing the number of nodes in another layer during experiments. The results are shown in Fig.4. and Fig.5.. We observed slight improvements by increasing the number of nodes in the first hidden from 32 to 128. And it showed that the first hidden layer is more sensitive to different number of nodes compared with the second hidden layer.

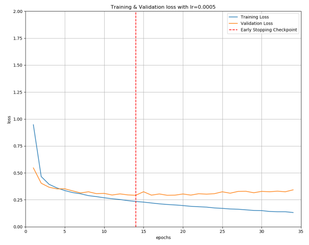
B. Convolutional Architecture

1) Architecture of CNN: The architecture of CNN model is shown below. The model has two convolution layers and max pooling layers to extract feature from the image. The purpose of the convolution layer is to extract high-order features from the image as well as reduce the dimensionality. Each convolutional layer has 64 different kernels to filter the image. Those filters are different from each other aiming to extract different features from the image. The convolution layer is followed by a 1 hidden layer and output layer. The architecture will maintain for 1 hidden layer in the whole experiment. Meanwhile, the model uses dropout method to prevent the model from overfitting. In the training set, 80% of data will be used as training and 20% of data is used as validation.

2) Feature extraction by the convolutional layer: First let us see what feature we can extract from the convolutional layer.


(a) Epochs:20


(b) Epochs:30


(c) Epochs:40

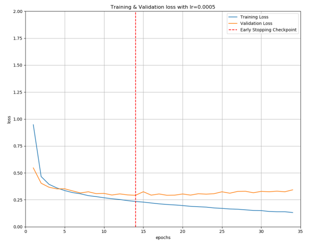
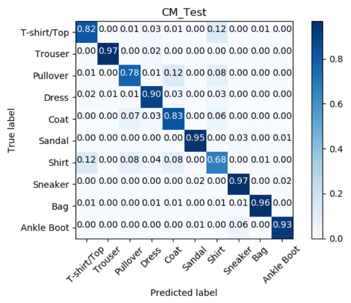
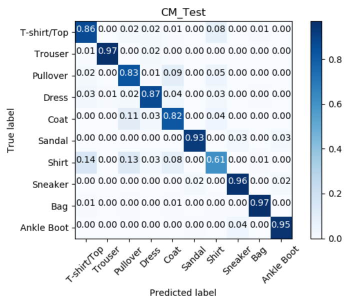

(d) Epochs:50

Fig. 3: Different Epochs


(a) Nodes:32,ACC:87%


(b) Nodes:64,ACC:86%

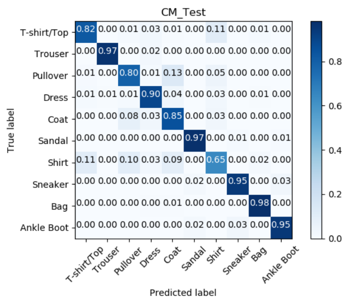
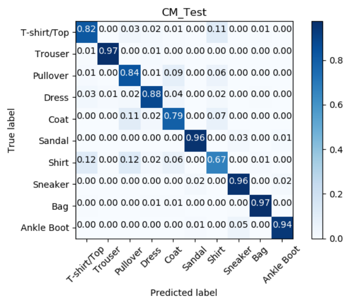
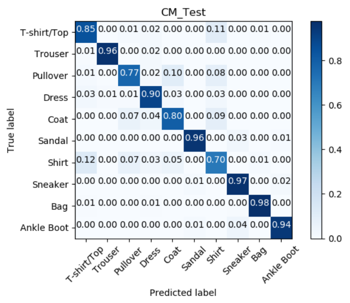

(c) Nodes:128,ACC:88%

Fig. 4: Different number of Nodes in $1^{st} Layer$


(a) Nodes:32,ACC:88%


(b) Nodes:64,ACC:88%

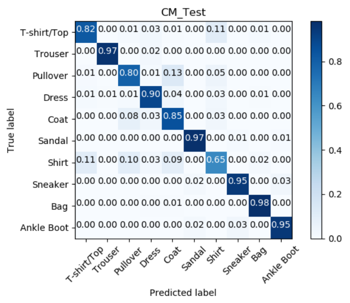
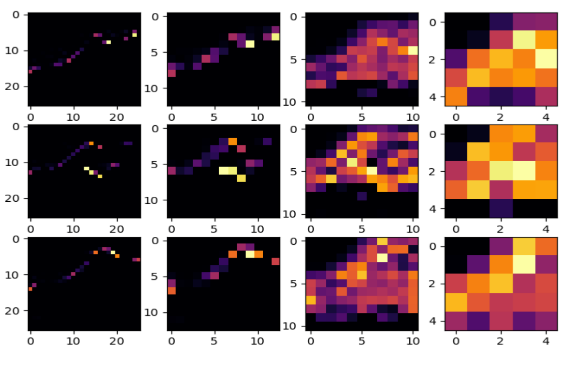
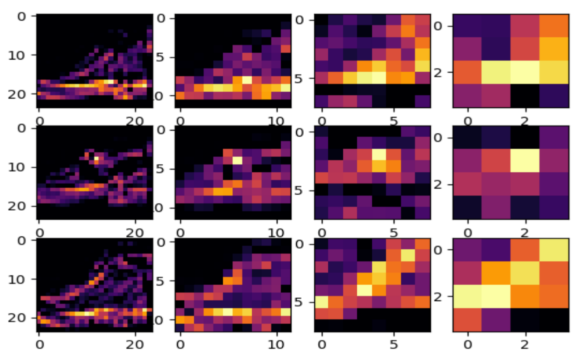

(c) Nodes:128,ACC:88%

Fig. 5: Different number of Nodes in $2^{nd} Layer$

. Fig 6 below shows the results in between the convolutional layers. Three different rows represent three different pictures. Four columns represent four different convolutional layers with order conv, maxpooling, conv and maxpooling. We can see from this picture that after the first convolution layer, most features of a shoe are maintained in the picture. After the first maxpooling layer, we can still identify this feature as a shoe. After the second convolutional layer, we can still get information of a shoe from the picture (e.g shoelace, color, shape). The fourth column is comparably vague because the picture has been through two maxpooling layers which means the picture is very pixelated.Therefore, during the experiment, we found that the model performs better when it abandoned the last maxpooling layer (About 0.5% accuracy improvement).


(a) 3x3 kernel



(b) 5x5 kernel

Fig. 6: Outputs from the convolution layers

3) *Learning rate:* This part will show the impact of the learning rate to the model. The optimizer I choose is Adam algorithm which can self-adapt the learning rate in every stepsize. Basically, what we can adjust for Adam algorithm is the initial value of the learning rate. The method I used to choose the learning rate is to allocate test number to space 0.001 to 0.1 by randomly generate a number r and reach 10 to the power 4*r. This method will separate the testing space in an average way comparing for example, increase the learning rate from 0.001 to 0.1 by stepsize 0.001. In each experiment, the model will be trained 10 epochs. We can see from those four confusion matrices(Fig 7) shown below. The model performs good when using low initial learning rate and awful when using comparably large learning rate. Therefore, I decide to fix the range of learning curve into 0.001 to 0.01 to draw the learning curve of training set and validation set to find the best learning rate.

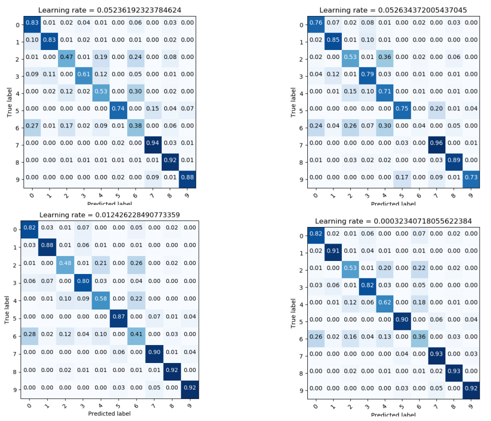


Fig. 7: Performances of the model by using different learning rate

The learning curves(Fig 8) of the training and validation set is shown below. We can see that when the learning rate equals to 0.001, the model performs the best in terms of the training and validation loss. In conclusion, the initial learning rate will be chosen as 0.001.

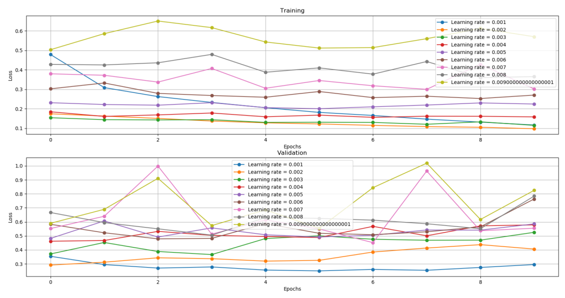


Fig. 8: Learning Curve

4) *Early stoping:* Sometimes, we would like to stop the model from training early in case the model will overfitting when doing too much training. A way to decide the timepoint of early stop is to see the learning curve of training and validation process. Fig 9 shows the training and validation loss. We can see that after 5 epochs, the validation curve starts to increase which means the model is overfitting. Therefore, we would like to stop training at the 5th or 6th epoch to keep model's best performance on the testing dataset.

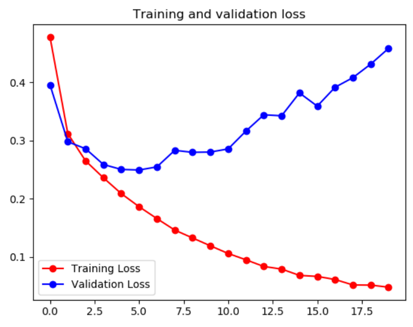


Fig. 9: Training and Validation curve

5) *Number of PE's and layers:* In the first part, the structure of convolution layer has been introduced. This part will mainly experiment on how to decide the structure of the fully connected layer which mainly include two hyperparameters, number of PE's and layers. First, we will see how number of layers will affect the performance. Fig 10 shows the confusion matrix when using one 128 units hidden layer and two hidden layers with 128 and 64 units respectively. According to the confusion matrices, there is not much improvement on the performance when using more layers.

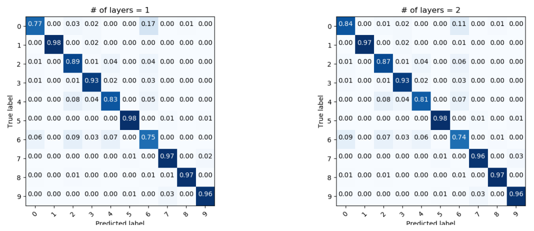


Fig. 10: Performance by using different fc layers

Second, the number of PE's will be decided. Four different confusion matrices are shown below with 32, 64,128 and 256 PE's in one hidden layer. When using 128 PE's in one hidden layer, the accuracy is approximately equals to 91%. When using 64 PE's in one hidden layer, the accuracy is approximately equals to 90%. When the number of PE's is greater than 128, the performance of the model does not have any improvement. Therefore, in terms of keeping the model as simple as possible (Ockham's Razor), we will keep the number of PE as 128 with one hidden layer.

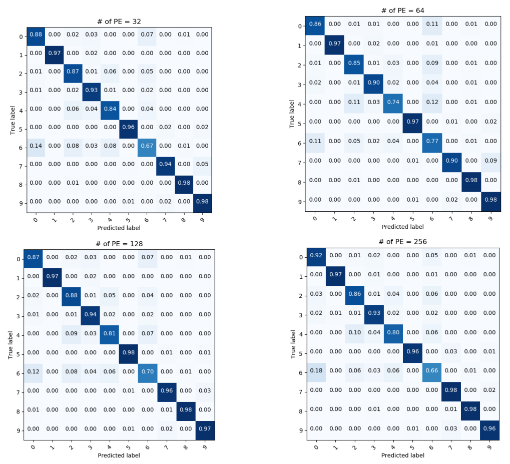


Fig. 11: Performances of the model by using different learning rate

6) *Other algorithms:* Besides the CNN, we also tried the RBF network on this dataset. Radial Basis Function network basically abandoned the metric on measuring similarity by the inner product in the neural network, it uses the gaussian kernel to measure the similarity between data points. In the radial basis function network, we first have to find the centroid of the gaussian kernel which is used as a prototype. One way is to randomly initialize the centroid. One way to find the centroid is to use clustering algorithm[5][6]. In the experiment, RBFN uses K-means clustering algorithm first to cluster the whole dataset into 10 clusters and then find the centre of each cluster as the prototype of each class respectively. The RBF layer connects to a fully connected layer with dropout. Suprisingly, the RBFN preforms well in this task. The final test accuracy of the RBFN can achieve 80.95% after 500 epochs. The training and validation process is shown by Fig 12 and the confusion matrix is shown by fig 13. However, a drawback of the RBFN is that it has to first find a optimum centre by using clustering algorithm. For a very large dataset, this can be very slow to converge.

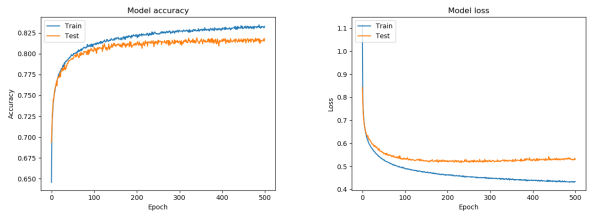


Fig. 12: RBF acc and loss

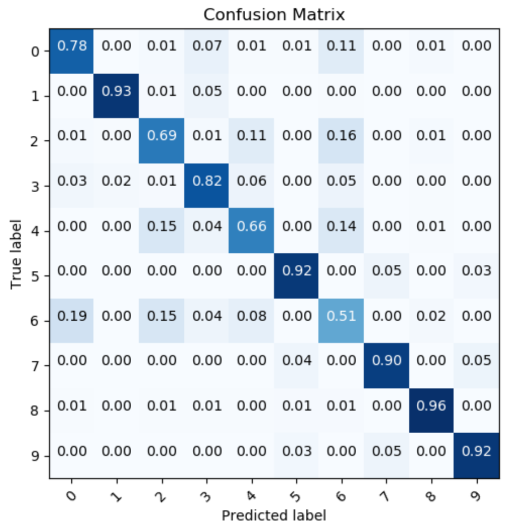


Fig. 13: Confusion Matrix(RBF)

7) *Summary and Thinking:* In conclusion, the accuracy the model can get in the training set is 93.13% and 91.73% on the validation set. On the real testing set, the accuracy of the model can achieve 91.32%. When we looking at the confusion matrix, the model can not really do a good classification on class 1, 3, 5 and 7 which represent respectively top, pullover, coat and shirt. Because in each epoch, the training data is shuffled and separated into training and validation set, therefore, if we get unfortunately a bad shuffle which means have more catagories that shown below, it might compromise the model's performance. The reason why the model cannot separate those special catagories well we guess might be the similarity among those objects like T-shirt and shirt which is very similar to each other comparing to other pairs for example bags and sandals.

IV. Conclusion

Overall, we carefully compared the performance of CNN and MLP in the Fashion-MNIST dataset. By introduction early stopping criteria, we successfully retained the best model during training procedure. Compared with CNN, the MLP model is less sensitive to overfitting since the loss of validation didn't rapidly increase after a certain number of epochs. However, the recognition accuracy of MLP is 3.12% less than that of CNN. which represented the bottleneck of accuracy of MLP. From the perspective of training epoch, we observed that it only took 6 epochs to obtain the best CNN model, however, it took nearly 15 epochs to approach the best MLP model in our experiments. The less training epoch required by CNN indicated the power of convolutional layer on feature extraction. It's also worth mentioning that the time-consuming of CNN for each epoch is approximately 60s, whereas MLP only needs no more than 7s. The comprehensive comparison between number of epochs and unit time consuming reveals that CNN takes more advantage of the data while requiring more training time for each epoch due to the complexity of convoluational layers. Conversely, MLP is computation-friendly but at the cost of being unable to capture more information of the data. The RBFN can achieve 80.95% accuracy on the testing dataset.

References

[1] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). ”ImageNet classification with deep convolutional neural networks” (PDF). Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386. ISSN 0001-0782.

[2] LeCun, Y. Cortes, C. (2010). MNIST handwritten digit database.

[3] Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf. arXiv:1708.07747

[4] Kingma, D. P. Ba, J. (2014). Adam: A Method for Stochastic Optimization (cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015)

[5] Chris McCormick: Radial Basis Function Network (RBFN) Tutorial (2013 Aug 15)

[6] Friedhelm Schwenker*; Hans A. Kestler;GuÈnther Palm (2000-12-18). Three learning phases for radial-basis-function networks