CENG 562 – TERM PROJECT PROGRESS REPORT

Group Members: **Barışkan Süvari – 1814755**

Serdar Oğuz Ata – 1554369

1. Introduction

Machine learning shows up in almost every aspect of our lives nowadays. We try to teach computers our perception of world as we humans see it. Perceiving objects, figures and patterns, classifying them is the most dealt area of machine learning. In the face detection problem first stage of the problem is training a machine to recognize shapes of human faces. Basic concept is here to teaching parameters to a machine so that when an image is given it will seek for specific patterns.

As teaching and detection processes can be implemented by many algorithms and methods, we in this project will focus on two well-known methods. These methods are Restricted Boltzmann Machine and Convolutional Neural Network. With a processor that has large enough compute capability detection can be done by the time video is taken. As the progress is made in the machine learning field, new algorithms may arise and the effort to solve the problem may be reduced. In this project we will first try to optimize our algorithm while teaching algorithm for the dataset and then evaluate them to observe which one performs more efficient results.

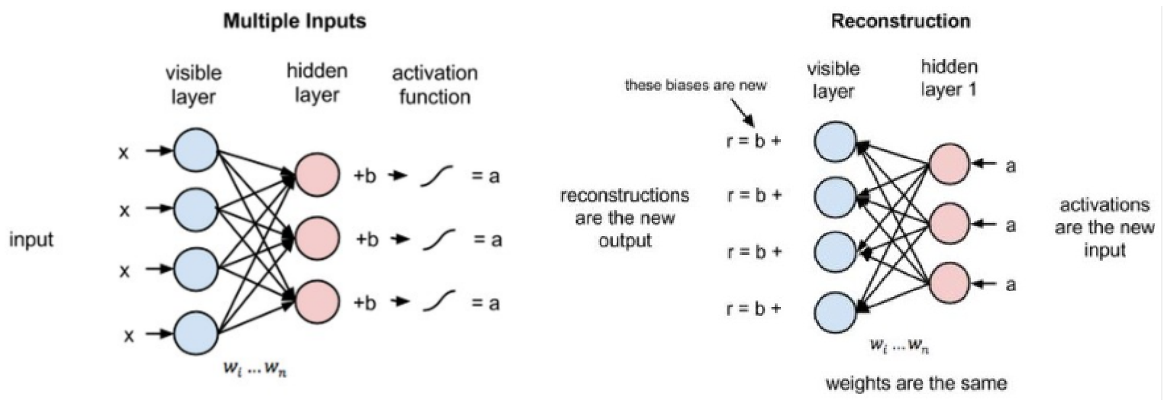
2. Methods to be used

Restriction in the **Restricted Boltzmann Machine** means that no intra layer connections are allowable. Network has only two layers and all connections exist between visible and hidden layer nodes. Each visible node takes input from a given greyscale image and performs the below computation on the value.

F (Weight * InputValue + Bias) = Output

F is an activation function, which gives the predetermined output value, if a value greater than the threshold is given as input.

RBM also process a backpropagation algorithm trying to reconstruct input image. The error between reconstructed image and the input image is used to recompute weight and bias values. Each time these two processes are done RBM tries to learn parameters for a better reconstruction. Processes done on layers of RBM can be seen on Figures 1.



https://deeplearning4j.org/restrictedboltzmannmachine

Figure 1: Construction of the output and backpropagation phases of RBM

Convolutional neural networks are very similar to the Feed- Forward Neural Networks in terms of architecture. Just like known neural networks CNNs are also composed of neurons with weight and bias like parameters. An input comes to the neuron weighted and biased with neurons parameters and then a non-linear function creates an output from it. Because of down sampling property of CNNs they are preferable compared to ordinary Neural Networks.

CNN architecture can be examined in 3 main layers, Convolution Layer, Max Pooling Layer (Sub-Sampling Layer) and Fully Connected Layer.

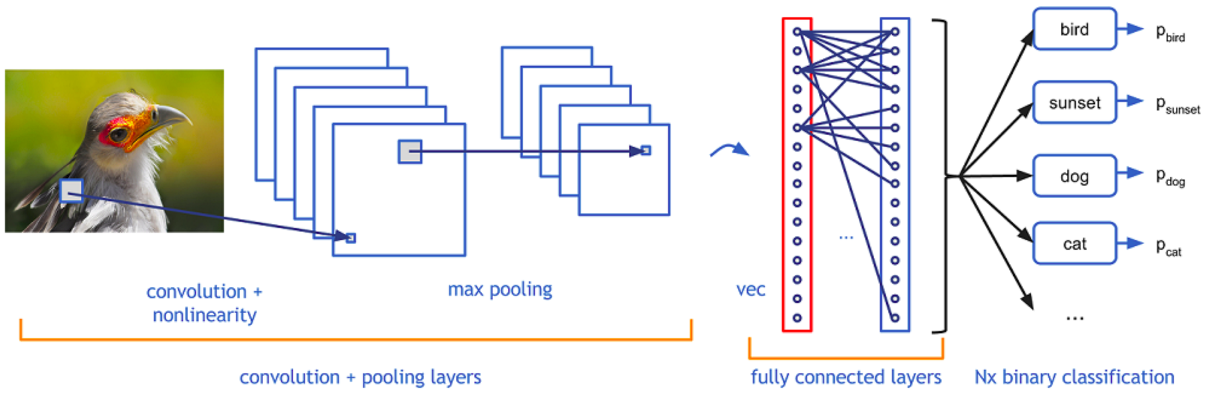
Convolution layer performs dot product with a sliding window on a given image. Each color channel is processed with different windows. Then all three results are summed with bias and written to an activation map matrix. Stacked sliding windows are also called as filter or kernel. Increasing the number of filters allows examining more features.

At the end of convolution layer a **ReLU(Rectified Linear Units) layer** might be added to perform an activation function thresholding at zero on the obtained activation maps. This operation can be represented by the following formula:

$f(x)=\max(0,x)$

In **Max Pooling Layer** layer a down sampling operation will be done on the matrix obtained from convolution layer. Main purpose in performing down sampling is to decrease number of parameters and computation hardness. First a filter size and stride is selected for down sampling operation. Then, just like in the convolution layer, filter window moves on the matrix and takes the largest value in the window. According to the selected filter size and stride a new matrix is created and obtained max numbers are placed in according with the positions in the previous matrix.

After computations of convolution and pooling layers, obtained results are classified one label per node in the **fully connected layer**. Since the neurons in this layer have full connections to previous nodes, activations are calculated by multiplication of weights and sum of biases. In this way, from an original image, class scores can be computed.



https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Figure 2: Layers and operations of CNN

3. Overall Design

Overall project is based on the comparison of the two well-known machine learning methods. As seen on the block diagram below, our path to follow on this project begins with training both algorithms with selected datasets. We will try to optimize parameters for both algorithms to obtain better results. Then, we will compare their performances with respect to the performance metrics given below.

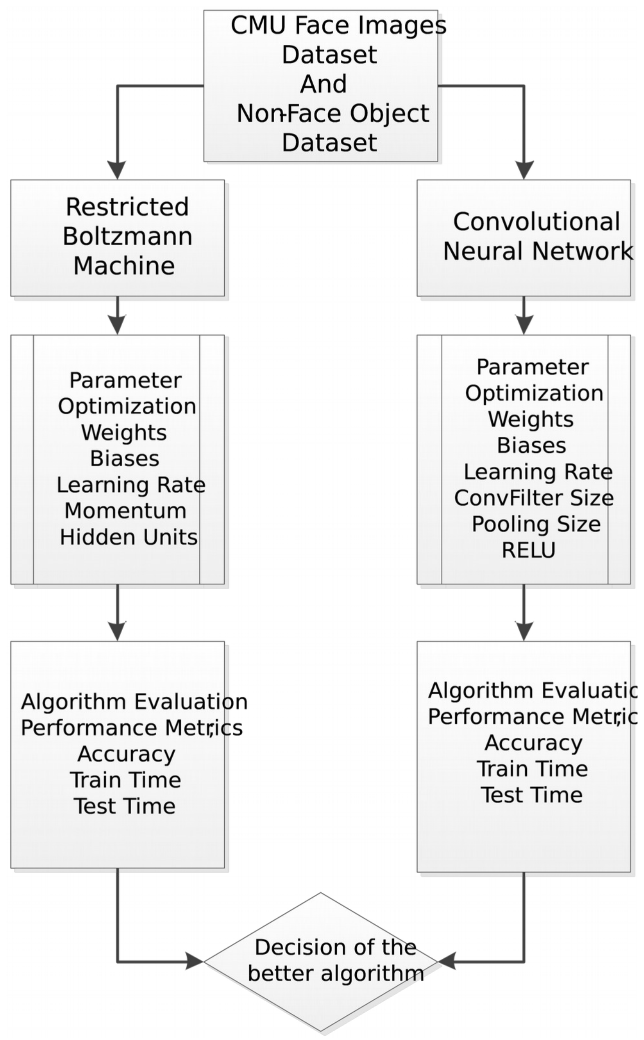

CMU Face Images Dataset And Non-Face Object Dataset
Restricted Boltzmann Machine
Convolutional Neural Network
Parameter Optimization Weights Biases Learning Rate Momentum Hidden Units
Parameter Optimization Weights Biases Learning Rate ConvFilter Size Pooling Size RELU
Algorithm Evaluation Performance Metrics Accuracy Train Time Test Time
Algorithm Evaluation Performance Metrics Accuracy Train Time Test Time
Decision of the better algorithm

Figure 3: Overall block diagram of the project

4. Dataset Characteristics

For train and test processes two datasets are used. These are CMU face images dataset and

Caltech 256-Object Categories dataset. To avoid overloading on our computers we choose to use

images with 60x64 pixel sizes and in greyscale. We choose 144 images from both face dataset and

non-face dataset. Train and test operations will be done with these 288 images by choosing random

batches. Two sample images from both datasets can be seen below.

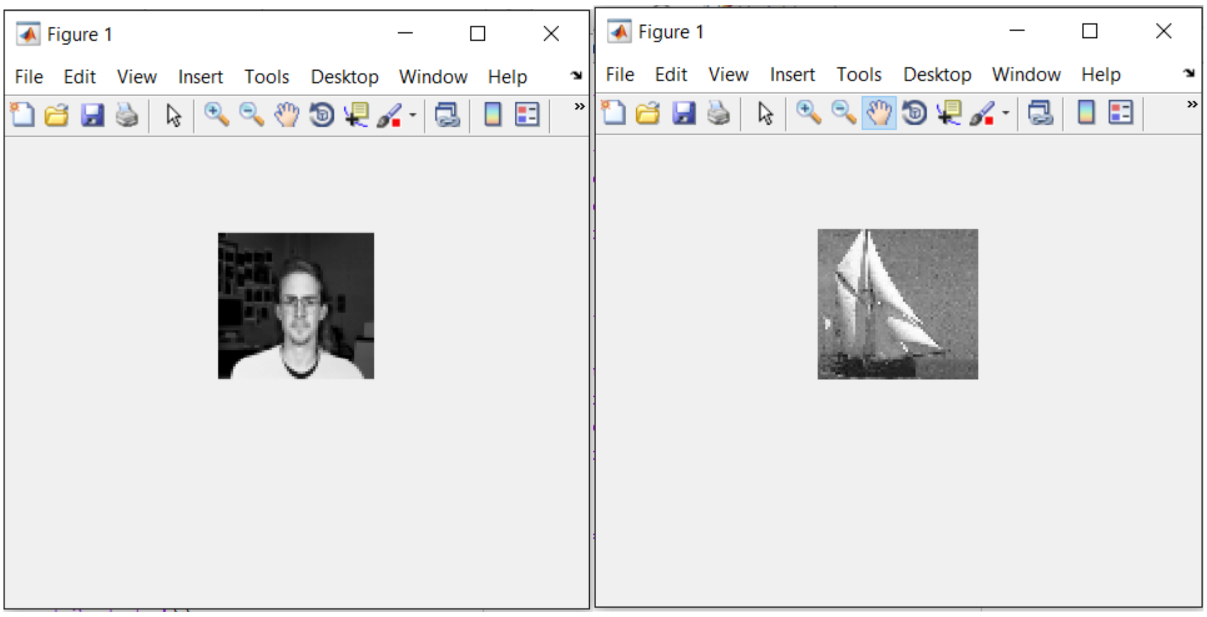


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help

Figure 4: Sample images from CMU face images dataset and Caltech 256-Object Categories dataset

5. Implementation

For the implementation of CNN algorithm, nnet toolbox of Matlab 2017a version is used. This toolbox provides the necessary structure to establish a CNN required for the project.

For this task, by using the given utilities, a Convolutional Neural Network with 2 convolutional layers is constructed. The network created will be trained to classify the images. The parameters of the CNN will be optimized to minimize the resulting error. The number of layers may also be increased if the error values are not low after training. Increasing the number of layers improves the performance and decreases the error but the complexity and the computation cost increases.

The codes for the mnist dataset and the results of the training are added with the files CNN_train_mnist.m and results_mnist.mat respectivey. The code for training the CNN for the images is given as CNN_train.m. The training for CNN_train.m code is not yet finalized. Its parameters should be optimized for best results.

Implementation of the RBM network is also done through a Matlab toolbox called ‘nnbox’. Within this toolbox layered structure of the RBM with essential parameters is implemented.

In RBM algorithm first the input images taken from datasets are read and put into matrix forms with ReadImgs.m file. Since the traning is done in a supervised manner, targetvalue matrix is also created. Then, first layer is created in accordance with the input image sizes. Number of the input neurons in the first layer is taken same as the total pixel sizes in the images. Hidden units in the first and second layer can be changed to optimize performance. Then, the second layer of the network is added to the network again with random parameter values. To classify images in test phase as face and non-face on the output, a simple perceptron with two output neurons is used. After creating overall network structure, different parameter values will be tried for optimizing performance of algorithm.

6. Evaluation of the Results

Training of CNN for images takes a long time. For now, the network is trained using mnist dataset to recognize hand written digits. The code for this task is added to the website. For this dataset, the accuracy of correctly classifying the digits is 0.9812.

After training for the mnist dataset, CNN for images is constructed, but it is not yet trained and the parameters are not yet optimized for this project. CNN is a very efficient method for classification purposes, but training and optimizing the parameters of CNN for images takes a long time.

As in the CNN, Mnist dataset has also been trained on RBM algorithm. Before going further on the algorithm, we thought a well-known dataset like mnist should be trained to observe faults. Training and testing classification errors on the mnist dataset are obtained 0.0298 and 0.0343 respectively. Error rate change in the training phase and classification error results can be found in the ‘RBM_Mnist_Results’ file. After some work on the RBM algorithm we will be able to train and test our selected datasets on the network.

7. **Conclusion**

In our point of view, we have made a huge progress in our project. Methods to be implemented studied in detail. The layer wise structure of networks and main blocks of the algorithms are constituted. On the other hand, exercising on the mnist dataset showed us our faults in the algorithms and the parts we need to focus on more. We will focus on these parts to implement better algorithms. Also, we should include more performance measurement techniques while evaluating the algorithms, like training time and CPU consumption.

8. **References**

http://cs231n.github.io/convolutional-networks/

http://deeplearning.net/tutorial/rbm.html

toolboxes:

http://www.mathworks.com/help/nnet/deep-learning.html

https://github.com/pixelou/nnbox

datasets:

http://kdd.ics.uci.edu/databases/faces/faces.data.html

http://authors.library.caltech.edu/7694/