**REPORT**:

In the neural network, there are two modes, the training phase and the testing phase. First, we read our txt file (train.txt or test.txt) and store it into an array dataset.

**K-means clustering:** On the dataset we apply K-means clustering in which we initially choose random 30 images out of the dataset. These are our centroids. Then for each image, we find the Euclidean distance to each of the centroids and assign it to the closest centroid to form clusters. Now we have 30 clusters in which all the dataset points are classified. To update the centroids, we find the average of each cluster to get new centroids and then re-assign the points to the centroids. We repeat this process until, our new centroids when updated are approximately the same as the old centroids. These optimum centroids will be our 30 means in the RBF function.

**The Neural Network:** Our neural network has a hidden layer of 30 neurons, and we set each neuron with one of the means we obtained earlier. When given an image to the neuron it gives us a single value output using the RBF formula:

**Output = e^ (-((x-mean)^2)/variance);**

Where variance is equal to sigma^2 (and in my network sigma is equal to 8).

The 30 outputs we get from the hidden layer are used as input for the next layer consisting of 10 neurons. These inputs are multiplied with weights to get weighted sum and then activation function (in this case sigmoid function) is applied to the weighted sum to get final values for the output layer. The maximum from this layer is considered the label for the image.

**Weight Updating**: During the training phase, the weights for the very first image are generated randomly within the given range. After receiving the output, we use the formula to update the weights for the next image by comparing our output “y” with the target output:

Note: the targets are converted into array form i.e. 5 is converted to [0,0,0,0,0,1,0,0,0,0]

i = (0-29) and k =(0-9)

**Wki(new) = Wki(old) + (Learning rate) \* dE**

Where,

**dE = (-1) \* (target[k] – y[k]) \* y[k](1-y[k]) \* hidden-layer-output[i]**

The weights are updated after each image and it goes for two epochs of the dataset. The weights at the end of two epochs are saved and then used as the initial weights for test phase.

Results:

**Training** **Phase**:

Epoch 1 ---> Accuracy usually around 55%

Epoch 2 ---> Accuracy usually around 80%

**Testing Phase**:

Epoch 1: Accuracy usually ranges from 35-60%

**Best Results my network has achieved so far are 66.7%** and the weights are stored in optimumweights.txt.