Name: Muhammed Essam **ID:** 20190462

Name: Fatema Hesham **ID:** 20190373

Mnist dataset:

The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

Features Extraction:

1- At first divide the image to 16 blocks each block has 7x7 pixels so I reshaped the image from 28x28 to 16x7x7

2- Then I have to calculate the centroid of each block using this formula for x and y

$X=\frac{\sum f(x,y)*x}{\sum f(x,y)}$, $y=\frac{\sum f(x,y)*y}{\sum f(x,y)}$.

Normalization on Features:

1. Normalization on features because the use of sigmoid in activation function of layers
2. Min_Max normalization is used:

   $x^{new}=\frac{x^{old}-min}{max-min}$

Neural Network Structure:

1. Neural Network divided into two main parts:
   i) Layers: Each layer has number of neural and each neural has its own weights, input and output.
      (1) Input layer
      (2) Hidden layers
      (3) Output layer

   ii) Activation function: Output is a function of the output of neural
2. The output is calculated by (WeightxInput) and pass the result to the activation function which is the input of the second layer.

- At neural first take input and give a random value to weights at calculates output forward **(Forward_Function)
- Calculate partial of the error at output layer **(Backward_Function)

  i) $\frac{\partial E}{\partial Y}$ = predicted – target (output layer only)

  Then calculate **delta** = $\frac{\partial E}{\partial Y}$ * predicted(output) * (1-predicted)

  //will use to update the delta of the previous layer (the concept of the chain rule)//

  ii) Each layer is affected by the error of the output layer so go backward to update weights and delta of the layer.

  $W^{new} = W^{old}$-Ŋ $\frac{\partial E}{\partial w}$

  $\frac{\partial E}{\partial w} = delta^{next\_layer} \ * \ \text{Output}^{current\_layer}$

  $\boldsymbol{delta^{current\_layer}}$=predicted($\text{Output}^{current\_layer}$)*(1-predicted)* $weights^{current\_layer} \ * \ delta^{next\_layer}$

  Repeat till reach input layer and update its weights

Accuracy:

- Increasing number of neurals and layers accuracy decreases
  Best at num_neurals of hidden layer = 60
  One hidden layer only.
- Increasing learning rate accuracy decreasing best at 0.01
- Accuracy = 75.8

END