**Noureldin Ghazal 43-15747**

**Kareem Fathy 43-6912**

**Mohamed Ahmed 43-16620**

**Mahmoud Yehia 43-12312**

**Task 2 Report**

* **Class Neuron:**
* This class can be initialized with or without weights and biases and it is the building block for our task. If initialized without an activation function is assumes the function to be sigmoid
* The class contains train method which is used in training our neuron to be able to modify weights for inputs. The output of this method depends on the activation function
* The class contains the method predict which uses the train function and compares its output to our ground truth to decide if the training function predicted the output correctly
* The Class contains the method backprobagation that takes as input the learning rate and calculates the derivative function to modify the weights accordingly
* Lastly the class also contains the method get\_error that calculates the errors resulting from the training process
* **Class input layer:**
* This class is initialized with the number of neurons in the layer and the inputs to be passed to each neuron.
* The class then initializes an array of neurons with length equal to number of neurons and pass the input to those neurons.
* **Class hidden layer:**
* This class is initialized with the number of neurons in the layer and the inputs to be passed to each neuron as well as the activation function of its neurons.
* The class then initializes an array of neurons with length equal to number of neurons and pass both the input and activation function to those neurons.
* **Class output layer:**
* This class is initialized with the number of neurons in the layer and the inputs to be passed to each neuron as well as the activation function of its neurons.
* The class then initializes an array of neurons with length equal to number of neurons and pass both the input and activation function to those neurons.
* **Class NN\_MLP:**
* This class is where all of the action happens
* It is initialized with an input layer, an output Layer and an array of hidden layers as well as a learning rate for the network
* The class contains a method train that takes as input the desired image as well as the ground truth, the method then passes each element of the input to the input layer with the ground truth for each element, the input layer then passes the those inputs to our first hidden layer with random weights at first, the layer then starts training while updating the weights using back propagation, after the first layer finishes training it passes the outputs of this process to the next hidden layer that works the same way, after the hidden layers are done training they send the final output to the output layer to train and update the weights as well.
* The class contains method test that is very similar to the train method except the weights are never updated during the test.
* Finally, the class contains the main method which reads both the grey image and RGB images, create the required layers with the number of neurons and the activation function and initial neuron input is an empty list. The method calls the training and testing method and then prints out the accuracy for both RGB image and grey scaled image.