**ANN**

**Project members**

**Muhammad Gamal 120541**

**Muhammad Younis 130606**

**Mustafa Mosad 120794**

**Lab supervisors: Dr. Muhammad El-Gayar**

**Dr. Halaby Muhammad**

**Doctor. Ahmed Hamza**

**Neural Network Project**

**Cats Vs Dogs Classification**

We downloaded the dataset of cats and dogs in two folders train and test

**Train**: contains 25000 images of cats and dogs, 12500 for cats and 12500 for dogs

**Test:** contains 12500 images of cats and dogs together

**Classes:**

1. **Pre:**

* We create a training data folder and we use this class to create dataset to be trained by running it and create two folder one for cats and other for dogs inside training data
* We will use CV2 library to read images from the train data and save images to the cats data folder if it’s a cat and dogs data folder if it’s a file. And tqdm library to iterate our images through the folder.
* We resized the images to have the same dimensions using copymakeborder in CV2 library by determining the desired size and specifying new and old size then computing the top, bottom, left and right of the image and applying all the size to the copymakeborder.

1. **Model:**

After creating the training data we will build our model as follows:

* Importing all needed layers and models from keras
* Firstly, we initializing the sequential model to configure the model for training.
* We will use two layers conv2D by input shape 50x50 = 2500 pixels and window size (3, 3) with the activation function relu, and maxpooling layer for temporal data with size 2, then we flatten the input.
* Now we will use the dense to create the output by applying activation functions sigmoid on the input, weighted matrix.
* We will normalize the layers by applying batch normalization and it will differ a lot in the accuracy. By putting the Use bias false in all the layers and the applying the batch normalization and now we can apply the activation function sigmoid.
* Finally we will compile the model to configure it for training.
* Now we will use fit model to train it for a given number of epochs

If we applied 1000 steps per epoch and say we have 5 epochs the accuracy will reach 80% after finishing the 5 epochs and accuracy started from 30%, but by increasing number of steps and decreases number of epochs say 2000 steps per epoch and we have 2 epochs, the accuracy will start from 60% and increases to 75% in the first epoch only, and in the second epoch accuracy reach 85%, 4000 will reach 91% and the best accuracy achieved in this dataset is 94% using 8000 steps and 25 epoch.

* We will run the model to test the accuracy and loss for each epoch and the accuracy will be 91%.
* Then we will save our weights.

1. **Use the model:**

* We will load the model and weights, we will compile it and predict if dogs it will print 1 if cats it will print 0 using the test folder.