# CS4990- Assignment 2: Facial Expression Classification

**Dataset:** Dataset used for the mini competition has training set of 16,175 examples and 3,965 examples in testing data which is used for determining the final score making use of the trained model. The data consists of three types of expression, 0- Angry, 1-Happy, 2-Neutral.

**Overview:** Using keras, I constructed a sequential convolution neural network, which implies the neural network has a linear stack of layers. The network has the following components:

- Convolutional Layers: These are the layers which are the building blocks of our network. They compute the dot product between their weights and the small regions to which they are linked. This is how these layers learn certain features from these images.
- Activation functions: They are the functions that are applied to the outputs of all layers in the network. In this model we have used two functions – *Relu* and *Softmax*.
- Pooling Layers: These are the layers which will downsample the operation along the dimensions. This layer helps to reduce the spatial data and minimize the processing power which is required.
- Dense layers: These are the layers which are present at the end of a C.N.N. These layers take in all the feature data which is generated by the convolution layers and perform decision making.
- Dropout Layers: These randomly turns off a few neurons in the network for preventing the overfitting.

- Batch Normalization: These normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This speeds up the training process.

The network is compiled using Adamx optimizer which uses a variable learning rate. Used *categorical_crossentropy* as the loss function, because of dealing with classification problem which involves multiple categories. The hyper parameters defined for fitting the model are:

- epochs=100
- batch_size=100

The data passes through the model 100 times and in batches of 100 images. 20% of the training data is used to validate the model after every epoch. After, the model is trained, it is then saved along with the weights, which is then loaded and used on test dataset to receive the final prediction. Then the final prediction is then saved in csv file. The libraries used in the program are keras, numpy and sklearn.

Techniques used for increasing the accuracy of the model include:

- Tuning the hyperparameters
- Increasing the number of layers in the model

**Results:**

```
Epoch 1/100
 - 26s - loss: 1.3886 - acc: 0.4235 - val_loss: 1.0811 - val_acc: 0.4464
Epoch 2/100
 - 23s - loss: 1.0784 - acc: 0.4472 - val_loss: 1.0547 - val_acc: 0.4522
Epoch 3/100
 - 23s - loss: 1.0544 - acc: 0.4503 - val_loss: 1.0130 - val_acc: 0.4869
Epoch 4/100
 - 23s - loss: 1.0181 - acc: 0.4672 - val_loss: 0.9499 - val_acc: 0.5666
Epoch 5/100
 - 23s - loss: 0.9844 - acc: 0.4940 - val_loss: 0.9769 - val_acc: 0.5308
Epoch 6/100
 - 23s - loss: 0.9688 - acc: 0.5008 - val_loss: 0.9257 - val_acc: 0.5607
Epoch 7/100
 - 23s - loss: 0.9416 - acc: 0.5122 - val_loss: 0.9006 - val_acc: 0.5756
```

```
Epoch 8/100
 - 23s - loss: 0.9329 - acc: 0.5236 - val_loss: 1.1317 - val_acc: 0.5138
Epoch 9/100
 - 23s - loss: 0.9164 - acc: 0.5569 - val_loss: 0.8724 - val_acc: 0.5811
Epoch 10/100
 - 23s - loss: 0.9060 - acc: 0.5695 - val_loss: 0.8371 - val_acc: 0.6136

Loaded model from disk
The loss on testing data 0.2124703611509509
The accuracy on testing data 0.9448531684809158
```