Q2. **TJEDC - Tom and Jerry Emotion Detection Challenge**

Input format :

The path of train images are stored in the local variable **‘train_images_path’**
The path of the csv containing names of training set images are stored in the local variable **‘train_csv_path’**
The path of test images are stored in the local variable **‘test_images_path’**
The path of the csv containing names of testing set images are stored in the local variable **‘test_csv_path’**

Data augmenting :

In the dataset that was given, we had 5 labels, but only 1941 images. We know that machine learning algorithms, and especially deep learning ones, do not work well with so less data.
So, we augmented data using the ImageDataGenerator class from Keras. We perform various transformation on the images like rotation, shearing, rescaling, etc.

The results we obtained, though were comparable to the result we got without appending data.

Data preprocessing :

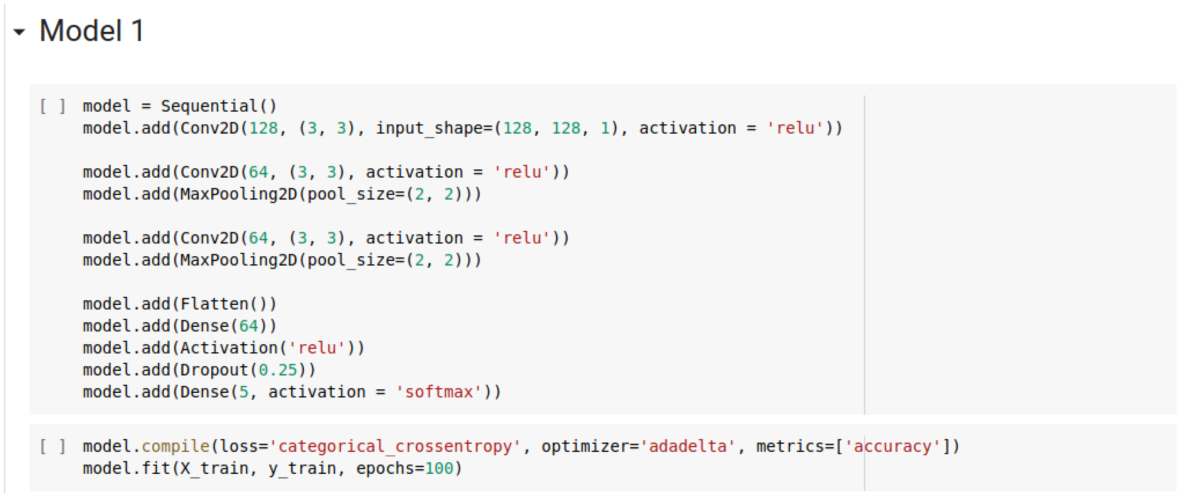
As part of data preprocessing, we store the images in grayscale format, and reduce their sizes to 128*128.

Model selection :

Since we had to detect the emotion of the characters present at each frame, we chose to do it using **Convolutional Neural Network (CNN)** models.

To test the models, we performed train validation data split and tested using 5 different CNN architectures.
In all of them, we vary the number of layers, activation functions and other parameters.

Some of the models and their summary is shown below -

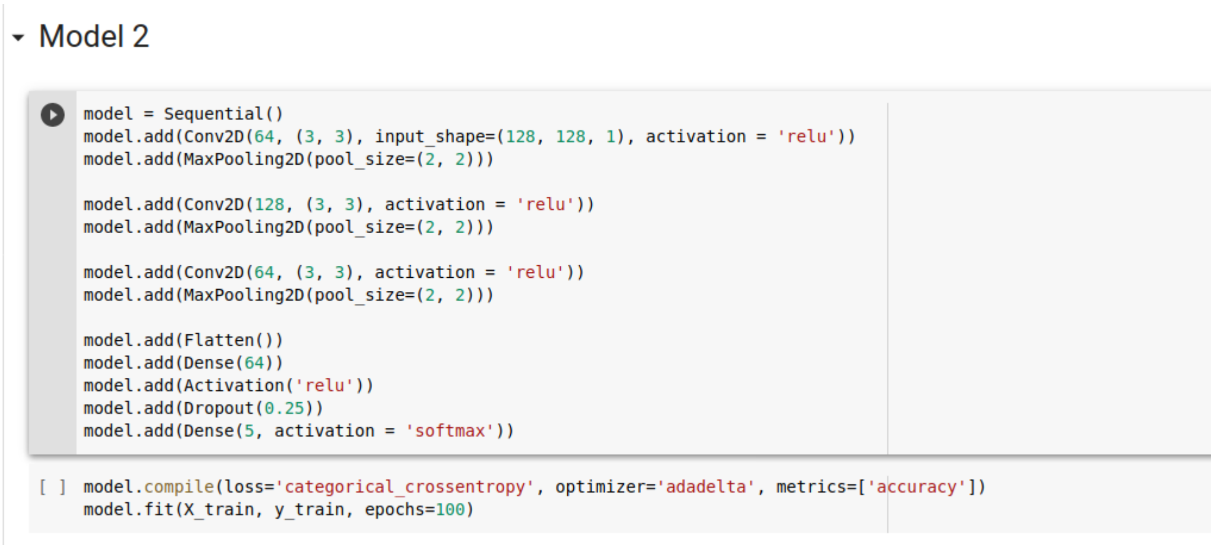


Model 1

```
[ ] model = Sequential()
    model.add(Conv2D(128, (3, 3), input_shape=(128, 128, 1), activation = 'relu'))

    model.add(Conv2D(64, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(64, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('relu'))
    model.add(Dropout(0.25))
    model.add(Dense(5, activation = 'softmax'))
```

```
[ ] model.compile(loss='categorical_crossentropy', optimizer='adadelta', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=100)
```

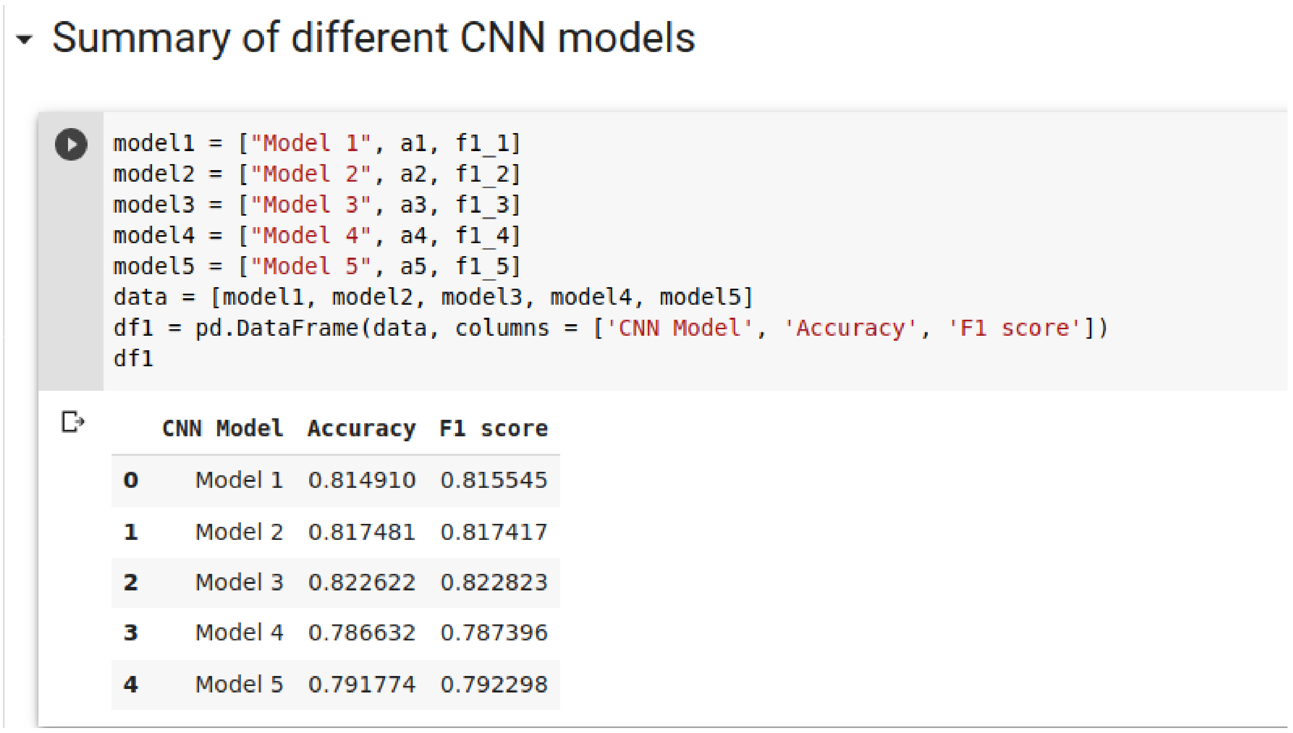
Sagnik Gupta 2019201003


Model 2

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(128, 128, 1), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(5, activation = 'softmax'))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adadelta', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100)
```

We test 5 such models.
Summary :


Summary of different CNN models

```
model1 = ["Model 1", a1, f1_1]
model2 = ["Model 2", a2, f1_2]
model3 = ["Model 3", a3, f1_3]
model4 = ["Model 4", a4, f1_4]
model5 = ["Model 5", a5, f1_5]
data = [model1, model2, model3, model4, model5]
df1 = pd.DataFrame(data, columns = ['CNN Model', 'Accuracy', 'F1 score'])
df1
```

	CNN Model	Accuracy	F1 score
0	Model 1	0.814910	0.815545
1	Model 2	0.817481	0.817417
2	Model 3	0.822622	0.822823
3	Model 4	0.786632	0.787396
4	Model 5	0.791774	0.792298

From, the above results, we see that we get comparable results using the different models.

We finally, load the test data, use our model to predict the outcomes of the emotions, and store them in a csv file.

We get a f1 score and accuracy of 1.0 on the final test data.

Sagnik Gupta 2019201003