Capstone Project – 5
Face Emotion Recognition

By,

Pravin. M

Premise

Global E-learning is estimated to witness an 8X over the next 5 years to reach USD 2B in 2021. India is expected to grow with a CAGR of 44% crossing the 10M users mark in 2021. Although the market is growing on a rapid scale, there are major challenges associated with digital learning when compared with brick and mortar classrooms. One of many challenges is how to ensure quality learning for students. Digital platforms might overpower physical classrooms in terms of content quality but when it comes to understanding whether students are able to grasp the content in a live class scenario is yet an open-end challenge.

Problem Description

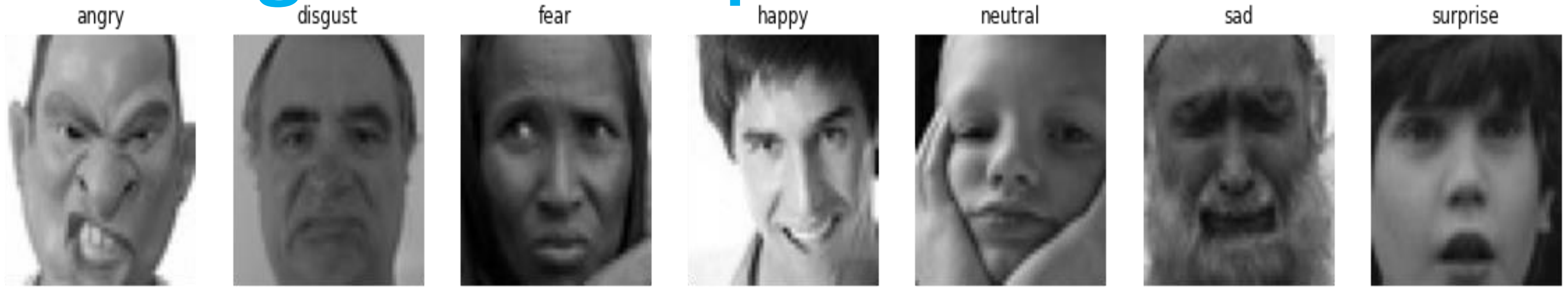
In a physical classroom during a lecturing teacher can see the faces and assess the emotion of the class and tune their lecture accordingly, whether he is going fast or slow. Digital classrooms are conducted via video telephony software program where it’s not possible for medium scale class (25-50) to see all students and access the mood. Because of this drawback, students are not focusing on content due to lack of surveillance. While digital platforms it comes with the power of data and machines in the form of video, audio, and texts which can be analyzed using deep learning algorithms. Deep learning backed system not only solves the surveillance issue, but it also removes the human bias from the system, and that can be analyzed and tracked.

Dataset Description

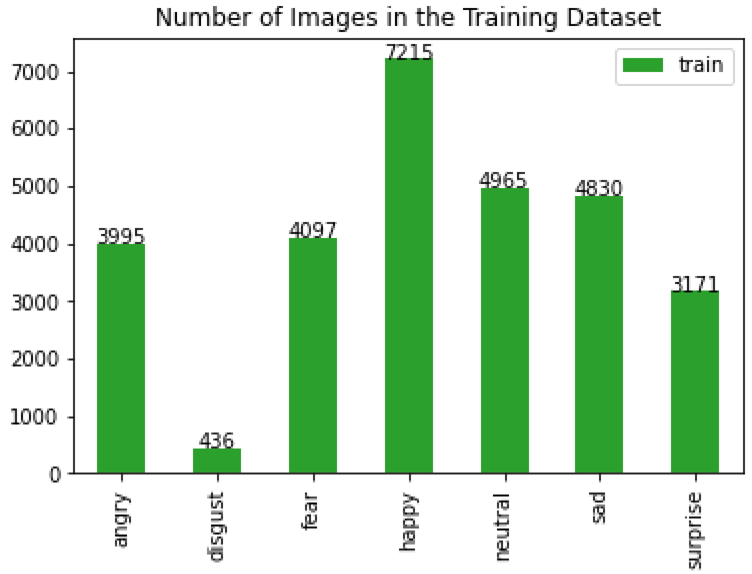
Fer2013 : The dataset consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The whole dataset is splitted into train and test sets. With seven emotions as follows

1. Angry
2. Disgust
3. Fear
4. Happy
5. Neutral
6. Sad
7. Surprise

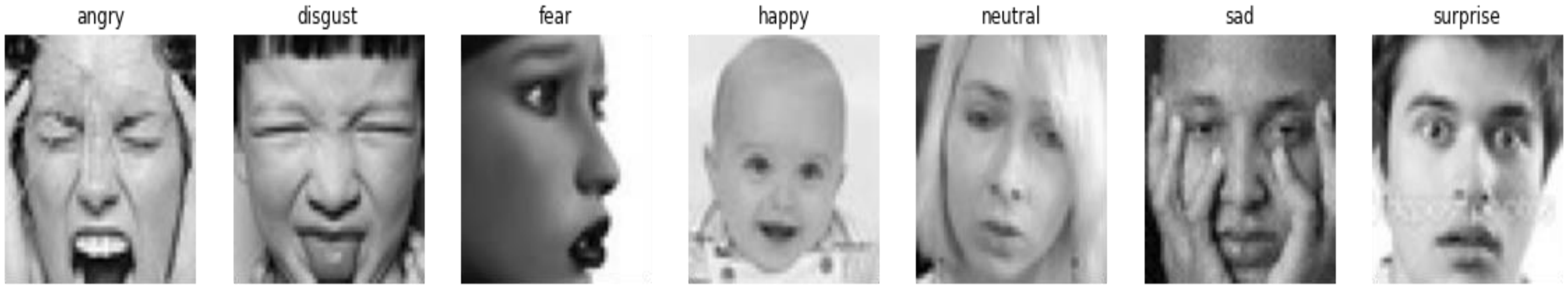
Training Dataset Inspection



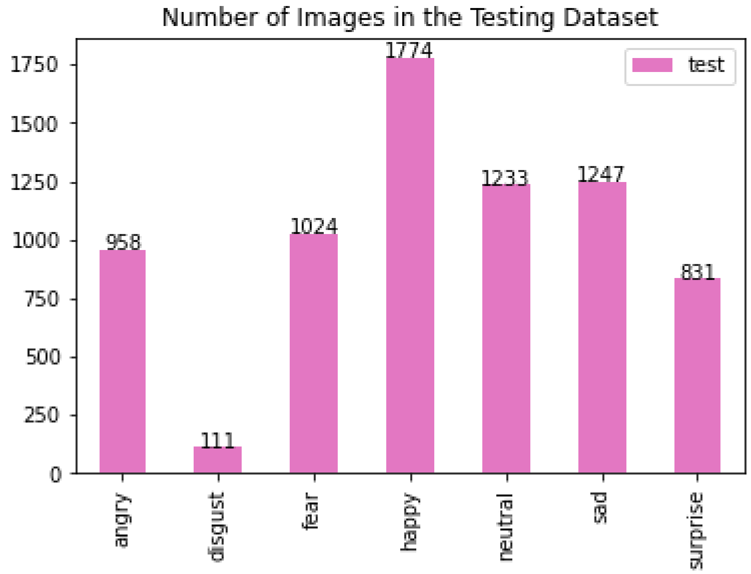
Looking at the distribution of datas among all the seven emotions and its examples on training dataset.



Testing Dataset Inspection



Looking at the distribution of datas among all the seven emotions and its examples on testing dataset.



Data Augmentation

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a normalizer and helps reduce overfitting when training a deep learning model. It is closely related to oversampling in data analysis.

Data Augmentation Modelling

```python
# Applying ImageDataGenerator method for data augumentation on train set
train_datagen = ImageDataGenerator(rescale=1./255,
                                   featurewise_center=False,
                                   featurewise_std_normalization=False,
                                   rotation_range=30,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   horizontal_flip=True)
```

```python
# Applying ImageDataGenerator method for data augumentation on test set
validation_datagen = ImageDataGenerator(rescale=1./255)
```

Data Augmentation Implementation

```python
# Implementing flow from directory method on training set
train_gen = train_datagen.flow_from_directory(train_dir,color_mode='rgb',
                                              target_size=(224,224),
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=True)
```

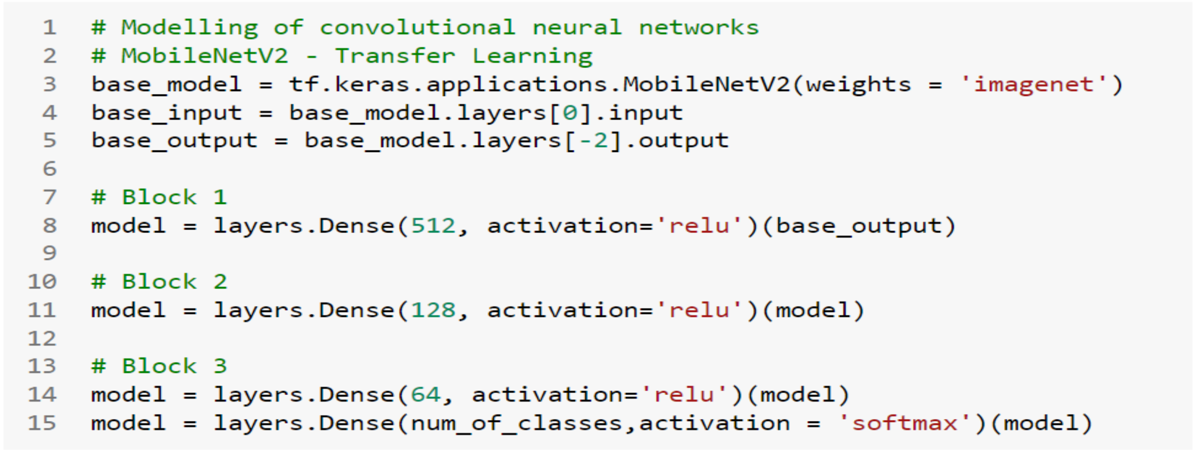
```
Found 28709 images belonging to 7 classes.
```

```python
# Implementing flow from directory method on testing set
valid_gen = validation_datagen.flow_from_directory(test_dir,color_mode='rgb',
                                                   target_size=(224,224),
                                                   batch_size=batch_size,
                                                   class_mode='categorical',
                                                   shuffle=True)
```

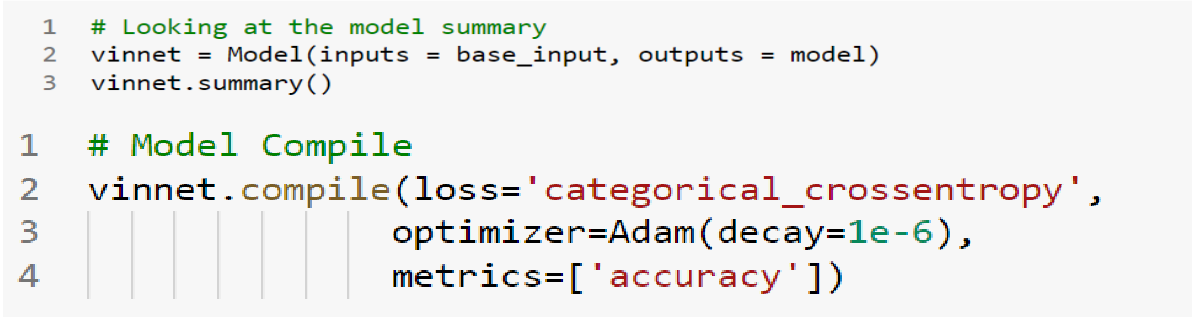
```
Found 7178 images belonging to 7 classes.
```

Vinnet - Modelling

Vinnet Model is developed by transfer learning. MobileNetV2 which is efficient on running on the smaller devices. With that I have created my own neural network – Vinnet.



```python
# Modelling of convolutional neural networks
# MobileNetV2 - Transfer Learning
base_model = tf.keras.applications.MobileNetV2(weights = 'imagenet')
base_input = base_model.layers[0].input
base_output = base_model.layers[-2].output

# Block 1
model = layers.Dense(512, activation='relu')(base_output)

# Block 2
model = layers.Dense(128, activation='relu')(model)

# Block 3
model = layers.Dense(64, activation='relu')(model)
model = layers.Dense(num_of_classes,activation = 'softmax')(model)
```

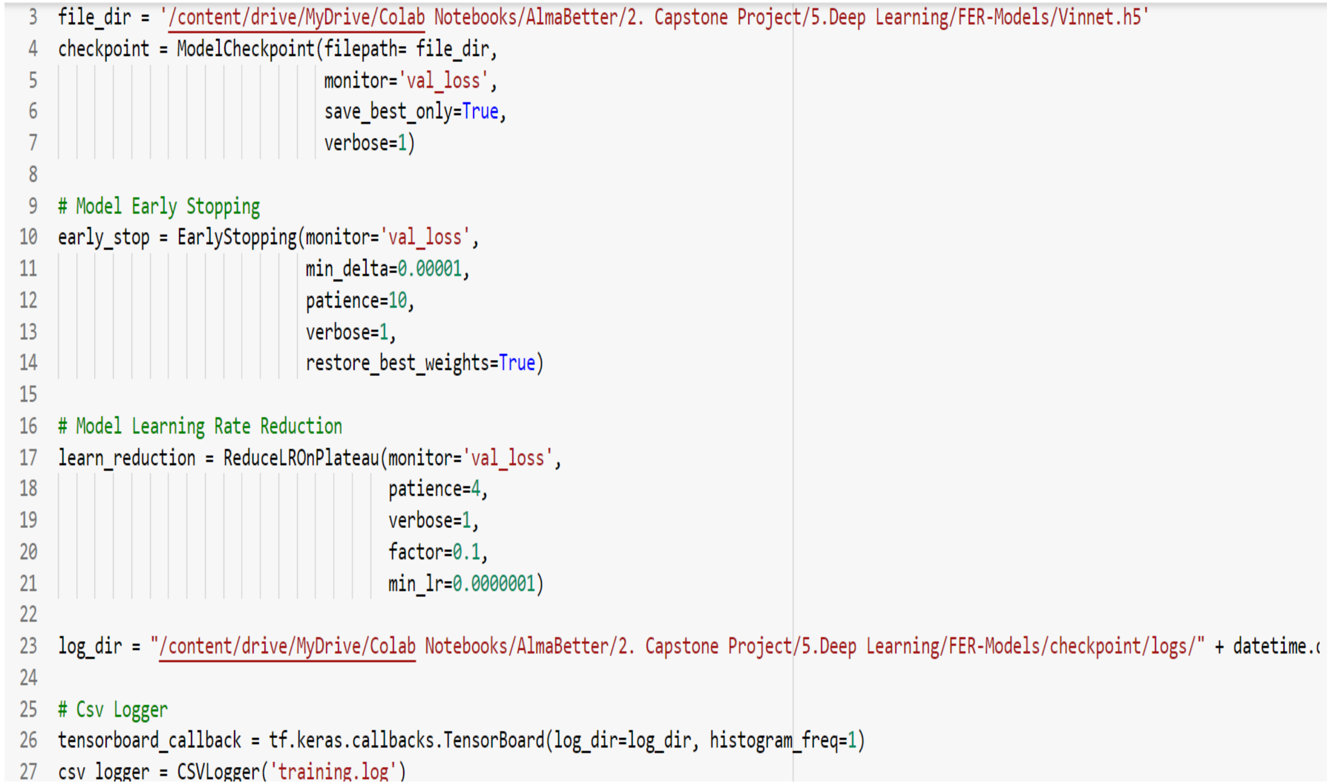


```python
# Looking at the model summary
vinnet = Model(inputs = base_input, outputs = model)
vinnet.summary()
```

```python
# Model Compile
vinnet.compile(loss='categorical_crossentropy',
              optimizer=Adam(decay=1e-6),
              metrics=['accuracy'])
```

Vinnet - Model Monitoring

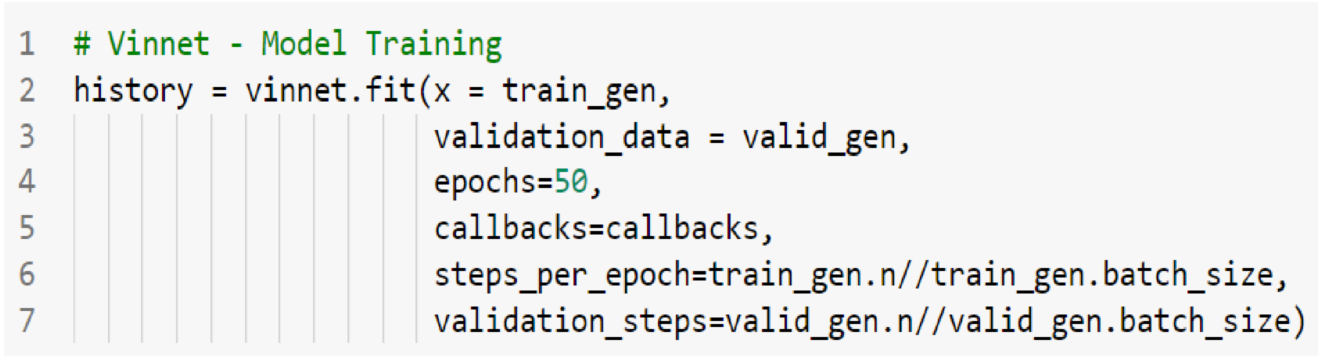
Vinnet Model Checkpoint are used to monitor the model during training phase.



```
file_dir = '/content/drive/MyDrive/Colab Notebooks/AlmaBetter/2. Capstone Project/5.Deep Learning/FER-Models/Vinnet.h5'
checkpoint = ModelCheckpoint(filepath= file_dir,
                              monitor='val_loss',
                              save_best_only=True,
                              verbose=1)

# Model Early Stopping
early_stop = EarlyStopping(monitor='val_loss',
                           min_delta=0.00001,
                           patience=10,
                           verbose=1,
                           restore_best_weights=True)

# Model Learning Rate Reduction
learn_reduction = ReduceLROnPlateau(monitor='val_loss',
                                    patience=4,
                                    verbose=1,
                                    factor=0.1,
                                    min_lr=0.0000001)

log_dir = "/content/drive/MyDrive/Colab Notebooks/AlmaBetter/2. Capstone Project/5.Deep Learning/FER-Models/checkpoint/logs/" + datetime.

# Csv Logger
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
csv_logger = CSVLogger('training.log')
```

Vinnet - Model Training

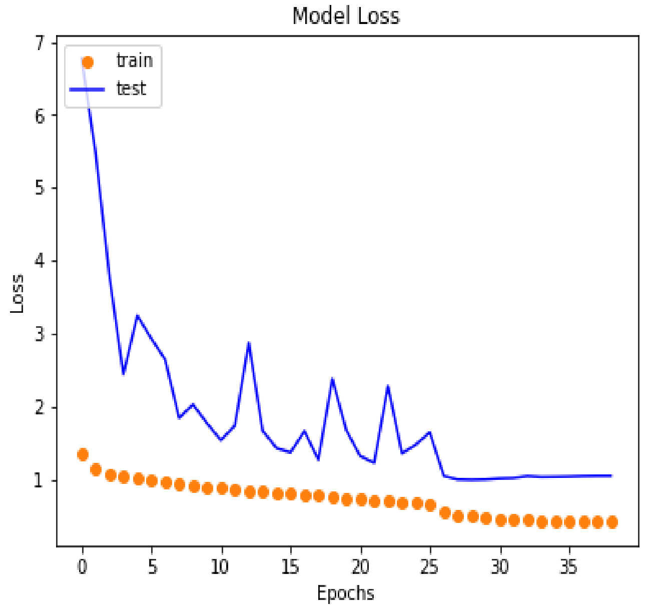
Vinnet Model Training done by initiating vinnet.fit to history and passing necessary parameters to the fit method.

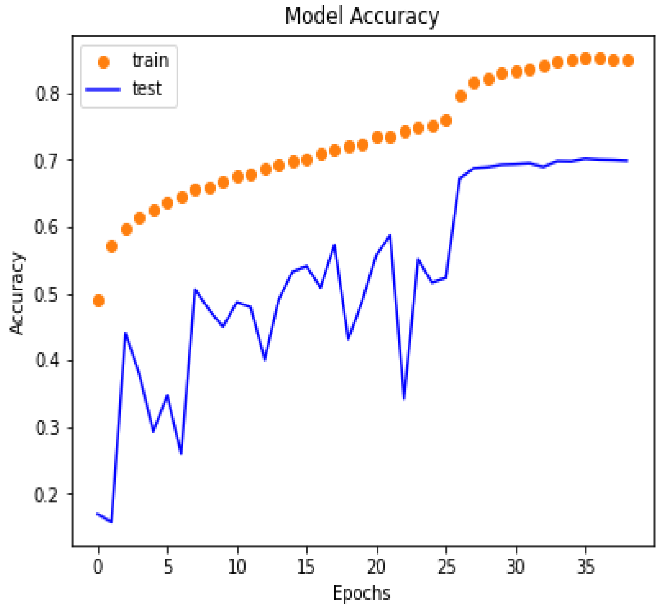


```python
# Vinnet - Model Training
history = vinnet.fit(x = train_gen,
                     validation_data = valid_gen,
                     epochs=50,
                     callbacks=callbacks,
                     steps_per_epoch=train_gen.n//train_gen.batch_size,
                     validation_steps=valid_gen.n//valid_gen.batch_size)
```

Vinnet – Model Performance Visualization

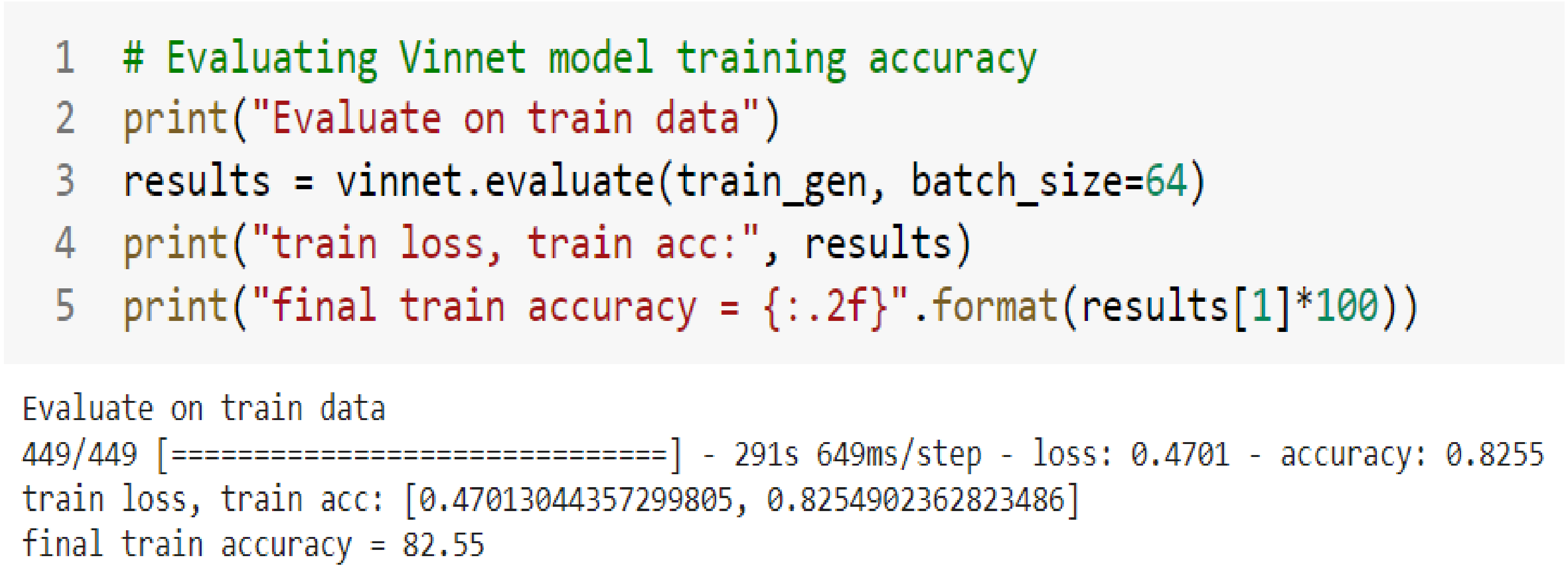
Vinnet Model Performance Visualization graphs are plotted to analyze how the model performed such as model loss vs epochs and model accuracy vs epochs.





Vinnet – Model Training set Evaluation

After successful training. Model is evaluated using train dataset. To find out final train accuracy.

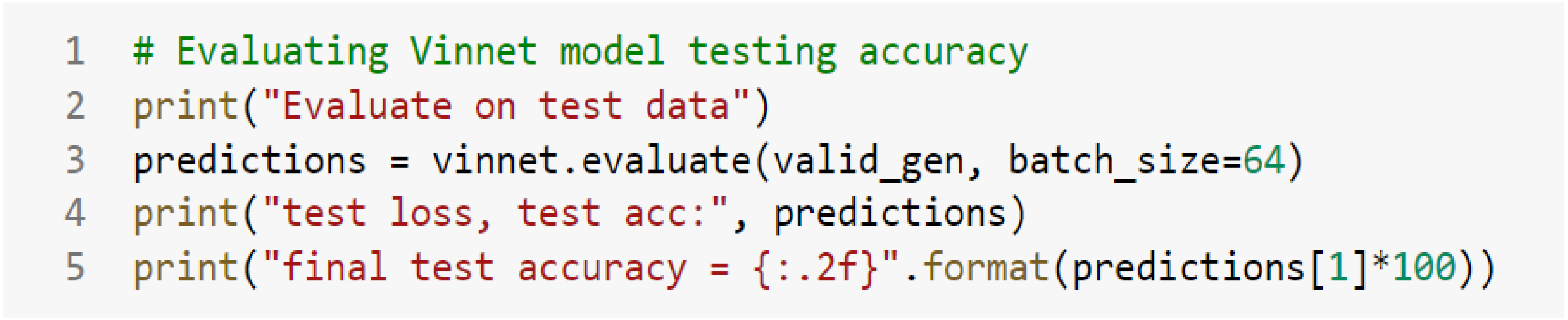


```python
# Evaluating Vinnet model training accuracy
print("Evaluate on train data")
results = vinnet.evaluate(train_gen, batch_size=64)
print("train loss, train acc:", results)
print("final train accuracy = {:.2f}".format(results[1]*100))
```

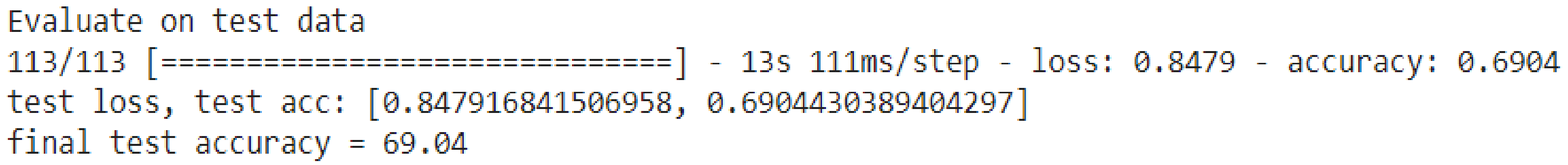
```
Evaluate on train data
449/449 [==============================] - 291s 649ms/step - loss: 0.4701 - accuracy: 0.8255
train loss, train acc: [0.47013044357299805, 0.8254902362823486]
final train accuracy = 82.55
```

Vinnet – Model Testing set Evaluation

After successful testing. Model is evaluated using test dataset. To find out final test accuracy.



```python
# Evaluating Vinnet model testing accuracy
print("Evaluate on test data")
predictions = vinnet.evaluate(valid_gen, batch_size=64)
print("test loss, test acc:", predictions)
print("final test accuracy = {:.2f}".format(predictions[1]*100))
```



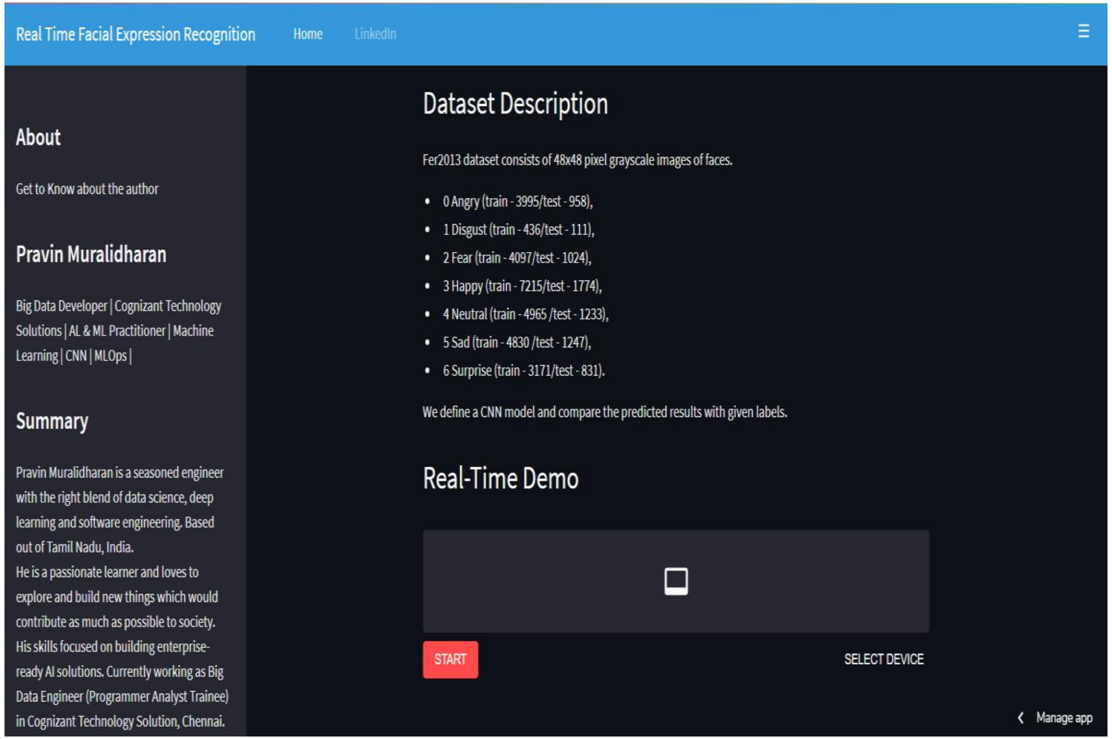
```
Evaluate on test data
113/113 [==============================] - 13s 111ms/step - loss: 0.8479 - accuracy: 0.6904
test loss, test acc: [0.847916841506958, 0.6904430389404297]
final test accuracy = 69.04
```

Vinnet Model – Random Image Testing

Before deploying directly into cloud as web-app. I’ve tested my model with some random new labeled image in the internet. And my model have performed well on those image predictions also. Out of 10 images 7 were correctly predicted by my model.

Web App – Development

Web app was developed with the help of streamlit library. Streamlit support web development through python. For face detection haarcasacade frontalface.xml and for predictions pre-trained model was loaded into the web-app and used for real time demo. And the web-app works based on webrtc protocol.



About

Get to Know about the author

Pravin Muralidharan

Big Data Developer | Cognizant Technology Solutions | AL & ML Practitioner | Machine Learning | CNN | MLOps |

Summary

Pravin Muralidharan is a seasoned engineer with the right blend of data science, deep learning and software engineering. Based out of Tamil Nadu, India.
He is a passionate learner and loves to explore and build new things which would contribute as much as possible to society.
His skills focused on building enterprise-ready AI solutions. Currently working as Big Data Engineer (Programmer Analyst Trainee) in Cognizant Technology Solution, Chennai.

Dataset Description

Fer2013 dataset consists of 48x48 pixel grayscale images of faces.

- 0 Angry (train - 3995/test - 958),
- 1 Disgust (train - 436/test - 111),
- 2 Fear (train - 4097/test - 1024),
- 3 Happy (train - 7215/test - 1774),
- 4 Neutral (train - 4965 /test - 1233),
- 5 Sad (train - 4830 /test - 1247),
- 6 Surprise (train - 3171/test - 831).

We define a CNN model and compare the predicted results with given labels.

Real-Time Demo

Cloud Deployment

Web app was developed initially in the local host as .py file. For cloud space streamlit cloud was used. Before deploying the whole project into the cloud. I had implemented both CI & CD pipeline.

Continuous Integration Pipeline:

For continuous integration pipeline GitHub is used as central repository for code. GitHub project repo was cloned into the streamlit cloud space. And thus making connection between central repository and streamlit cloud.

Continuous Delivery Pipeline:

For continuous delivery pipeline again GitHub is used. Whenever I made changes in my web-app and push the same to GitHub repo. It automatically deliver the same to streamlit cloud. And changes will reflects on the web-app.

Real Time Demo

For real time demo I've build an web-app and hosted on cloud. To avail those web-applications. You need to open your web browser it may be anyone. And navigate to the web-application url:https://share.streamlit.io/pravinmuralidharan/real_time_facial_expression_detection/main/Src/app.py . Wait until the whole web-application is loaded once in the browser. And then scroll to find **Real-Time Demo** header below there will be a **Start** button. Press the start button pop-up will ask for permission to access webcam and microphone. Select **Allow Access** then it will starts webcam and first it will detect the presence of face in the frame only then it will predict the emotion in the face. Thus it will demonstrate the application of the deep learning model which I've build on.

Conclusion

After successful training of vinnet model using train dataset. It is tested using valid dataset.

Then model is evaluated with train dataset to find final train accuracy which was around = **82.55**.

Similarly the model was also evaluated with valid dataset to find final test accuracy which was around = **69.04**.

Further model was tested with new sets of data. Those datas are not in both train and valid dataset.

Out of 10 data **7** were correctly Predicted by our vinnet model. Only 3 were wrongly predicted. Based on the above evaluation and testing our vinnet model is ready for deployment to the real world.