Performance Trend Classification

Abhishek Gupta

agupta001@regis.edu

PROBLEM STATEMENT

Problem Statement

Analyze the 30 days performance of thousands of wireless network mobile sites and identify any recent performance impact: degradation/improvement/neutral.

Existing Methodology

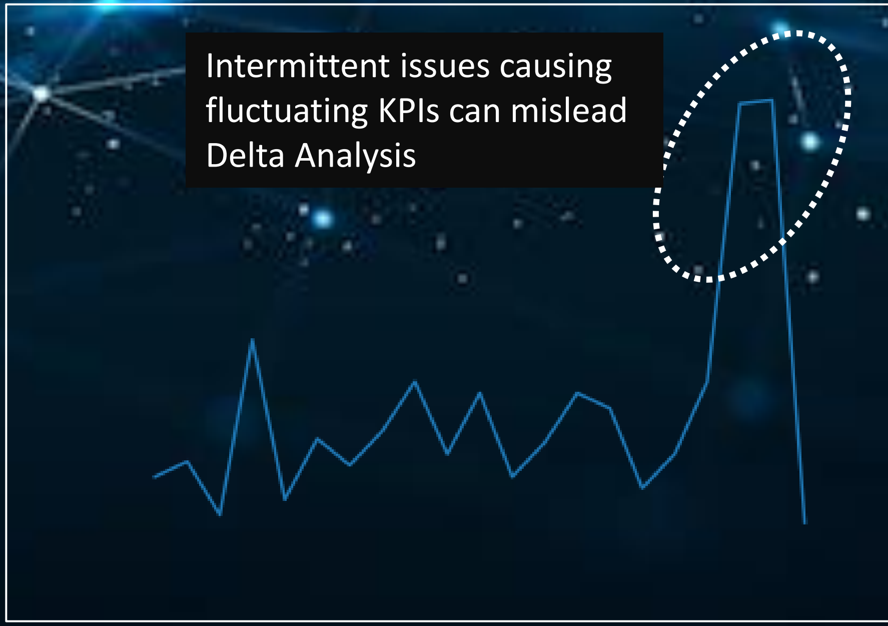
Perform an Excel based Pre and Post delta analysis on the targeted KPI and identify the degraded sites

Challenge

Intermittent/fluctuating issues can mislead average based delta analysis and hence the result may contain lot of unwanted noise.

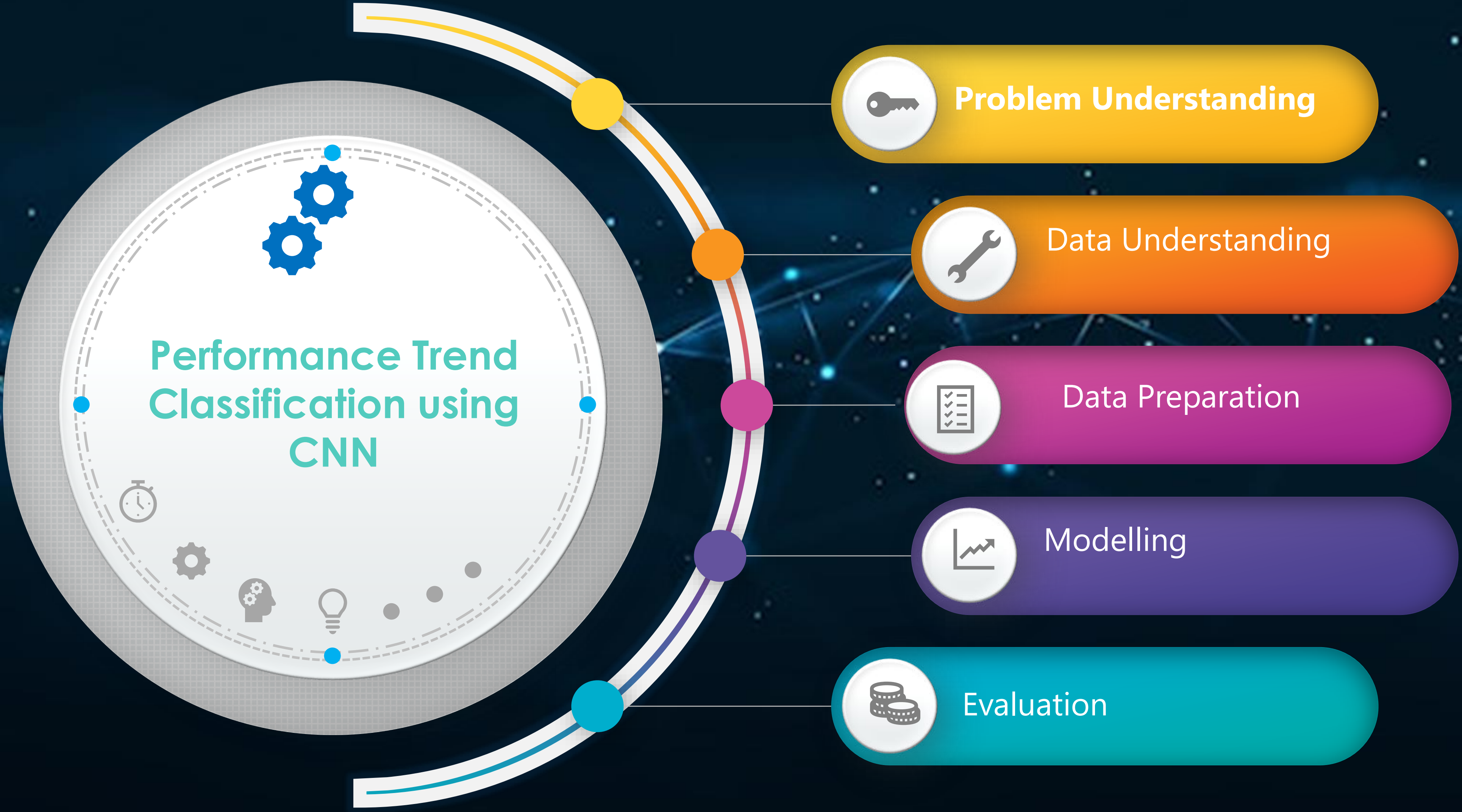
Proposed Solution

Instead of a pre/post delta analysis, build an image classification model using Convolutional Neural Network which can classify performance trends into Improved, Neutral, Degraded Category



Intermittent issues causing fluctuating KPIs can mislead Delta Analysis

PROJECT METHODOLOGY



Performance Trend Classification using CNN

Problem Understanding

Data Understanding

Data Preparation

Modelling

Evaluation

DATA UNDERSTANDING

- For my Project, I collected 30 days Call Drop performance data from thousands of wireless network mobile sites.

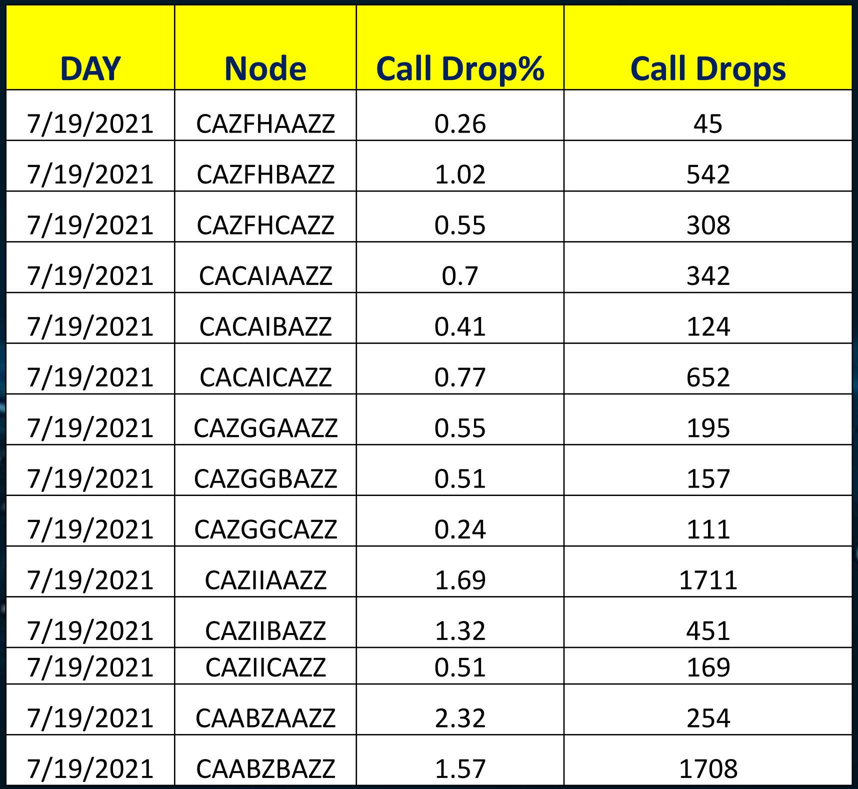
- The Call drop% (Drops/Total Calls) and Call drops data was collected

- The project was divided into 2 parts. For Part 1, I used only 1 feature(Call Drop%) and part 2 I used both the features.

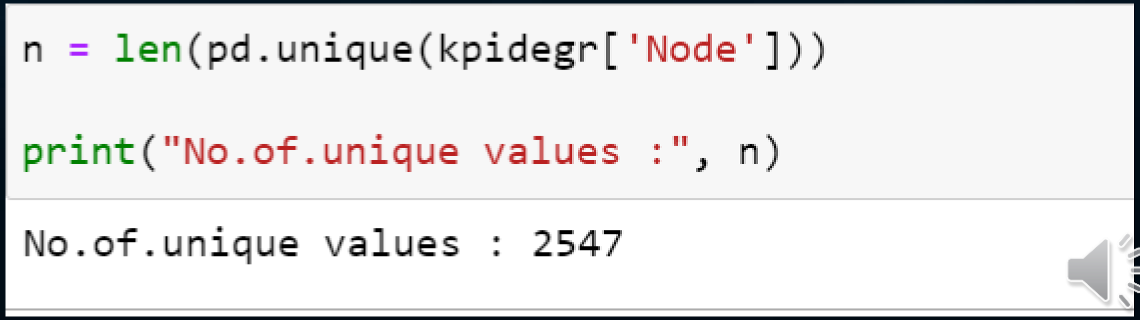
- Raw data has 4 columns : Date, Anonymized name of the Nodes, Call Drop%, Call Drops.

- Anonymization was done using Excel (Column 2)

- Performance data collected for 2547 Nodes



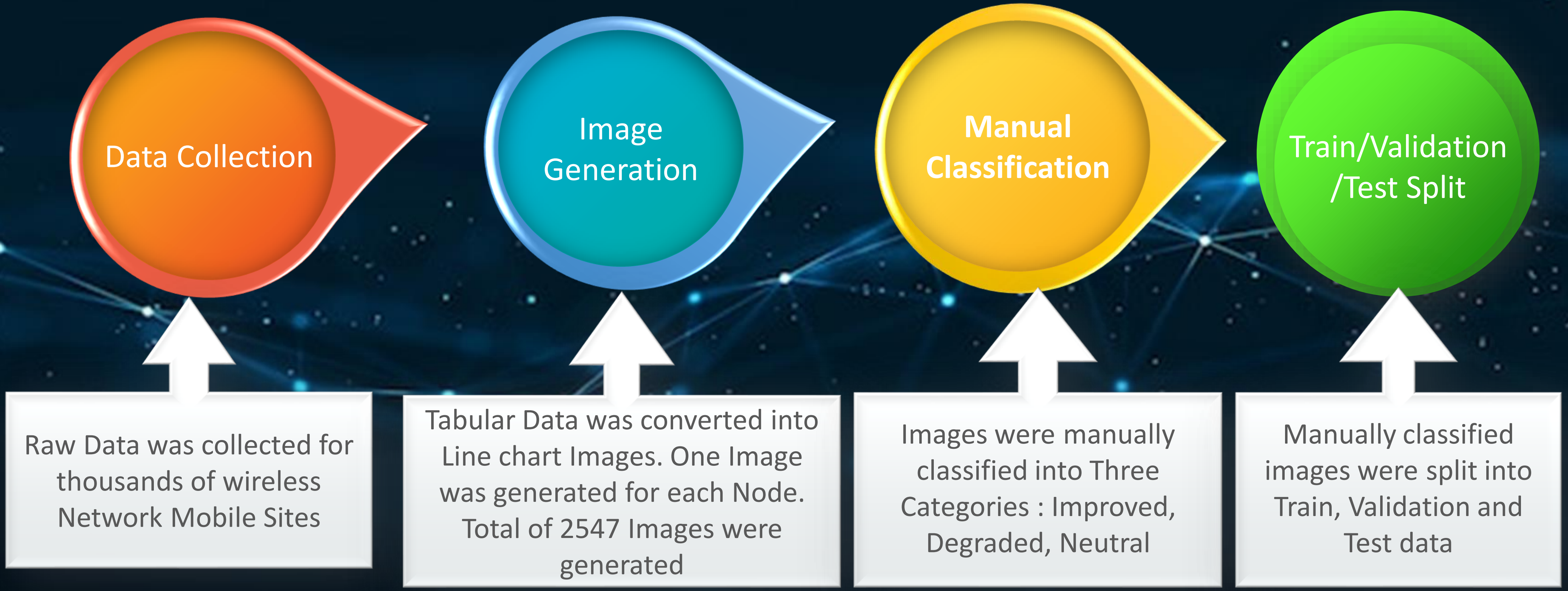
DAY	Node	Call Drop%	Call Drops
7/19/2021	CAZFHAAZZ	0.26	45
7/19/2021	CAZFHBAZZ	1.02	542
7/19/2021	CAZFHCAZZ	0.55	308
7/19/2021	CACAIAAZZ	0.7	342
7/19/2021	CACAIBAZZ	0.41	124
7/19/2021	CACAICAZZ	0.77	652
7/19/2021	CAZGGAAZZ	0.55	195
7/19/2021	CAZGGBAZZ	0.51	157
7/19/2021	CAZGGCAZZ	0.24	111
7/19/2021	CAZIIAAZZ	1.69	1711
7/19/2021	CAZIIBAZZ	1.32	451
7/19/2021	CAZIICAZZ	0.51	169
7/19/2021	CAABZAAZZ	2.32	254
7/19/2021	CAABZBAZZ	1.57	1708



```
n = len(pd.unique(kpidegr['Node']))

print("No.of.unique values :", n)
```

```
No.of.unique values : 2547
```

DATA PREPARATION



Data Collection

Raw Data was collected for thousands of wireless Network Mobile Sites

Image Generation

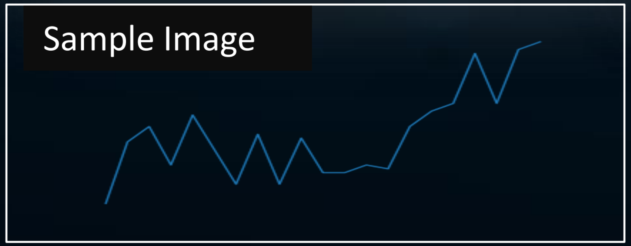
Tabular Data was converted into Line chart Images. One Image was generated for each Node. Total of 2547 Images were generated

Manual Classification

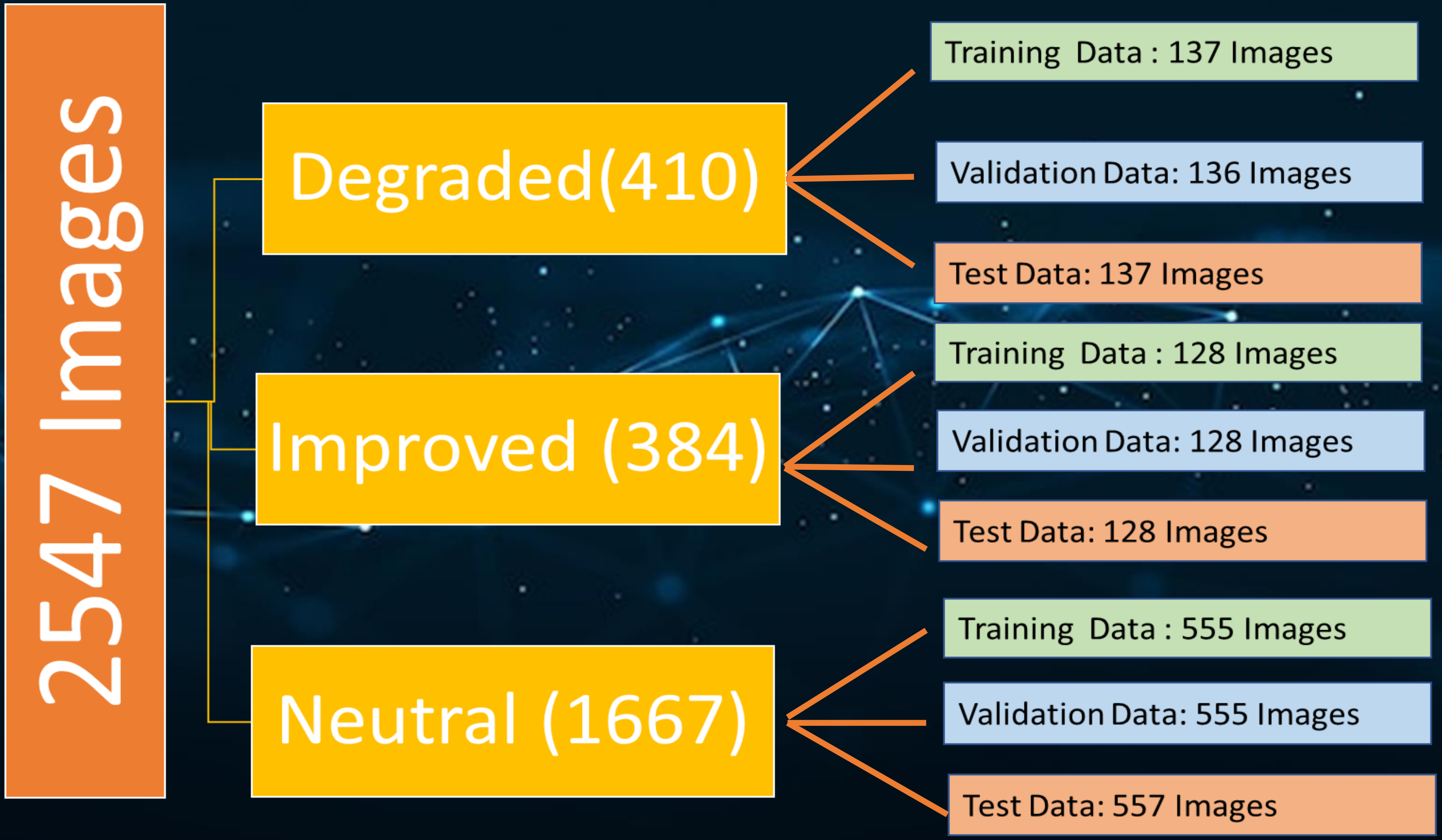
Images were manually classified into Three Categories : Improved, Degraded, Neutral

Train/Validation /Test Split

Manually classified images were split into Train, Validation and Test data



DATA PREPARATION



2547 Images

Degraded(410)

- Training Data : 137 Images
- Validation Data: 136 Images
- Test Data: 137 Images

Improved (384)

- Training Data : 128 Images
- Validation Data: 128 Images
- Test Data: 128 Images

Neutral (1667)

- Training Data : 555 Images
- Validation Data: 555 Images
- Test Data: 557 Images

MODELING

- I used Convolutional Neural Network to build a classification model to learn and classify images into three categories : Improved, Degraded, Neutral.

- CNN model was trained on training data and validation was done using Validation Data

- Model was tested on the Test Data and accuracy was recorded

- Fine tuning of the model was done to improve accuracy

- Transfer learning and Functional API was used to achieve further improvement in accuracy

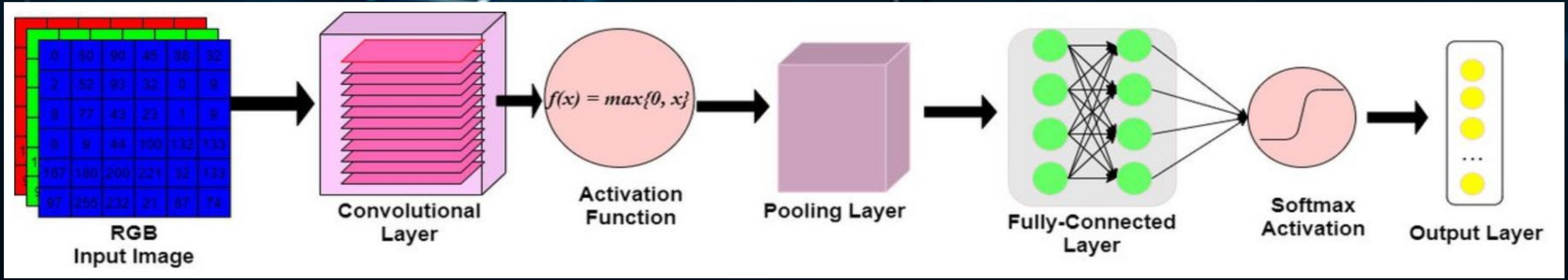


Image retrieved from https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/

EVALUATION

- Once the model was built , we checked the testing accuracy and validation accuracy of our model

- The accuracy metric used is classification accuracy which tells Correctly predicted images out of the total Images

- 4 Model were built for classifying the images with one feature

- For Model 1, I used the Keras Sequential API, where you have just to add one layer at a time, starting from the input. The model gave an accuracy of 67%
- For Model2 , Functional API was used and not much accuracy improvement observed(Accuracy still around 67%).
- For Model3, Transfer learning model InceptionV3 was used and an accuracy of 82% was achieved.
- For Model4 ,Transfer learning model VGG16 was used and an accuracy of 86.6% was achieved.

- 1 Model was built for classifying images with 2 features to see if further improvement can be achieved

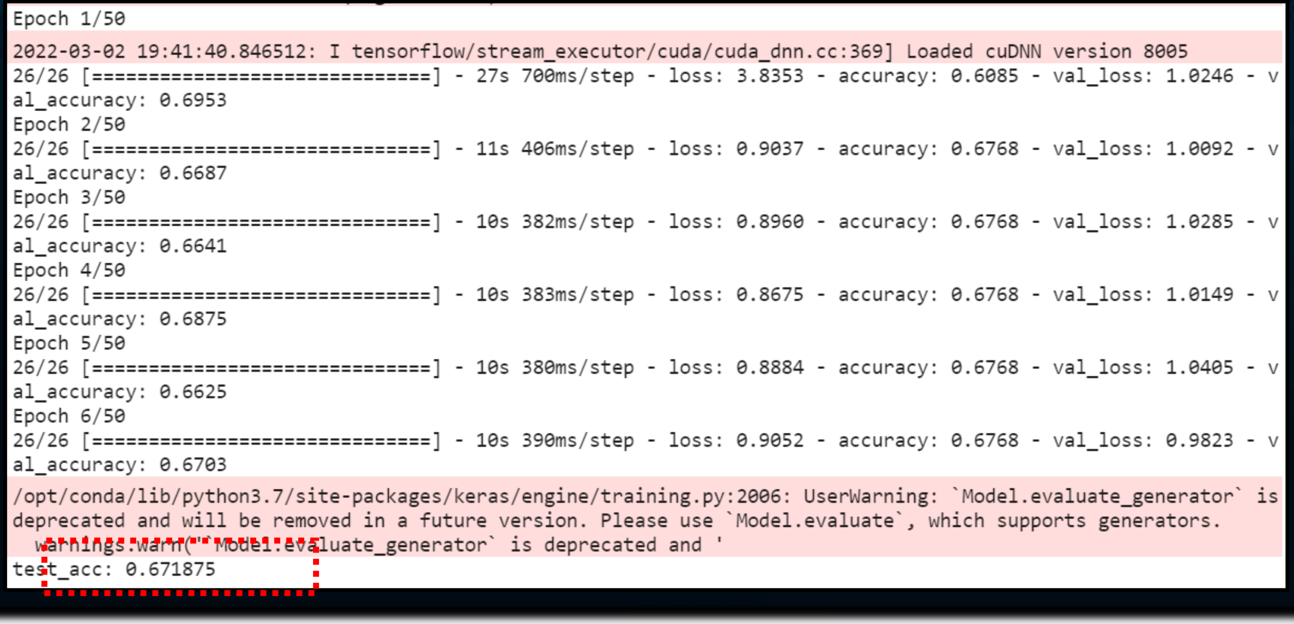
- VGG16 with RMSprop Optimizer was used and an accuracy of 73.9% was achieved.

EVALUATION MODEL 1

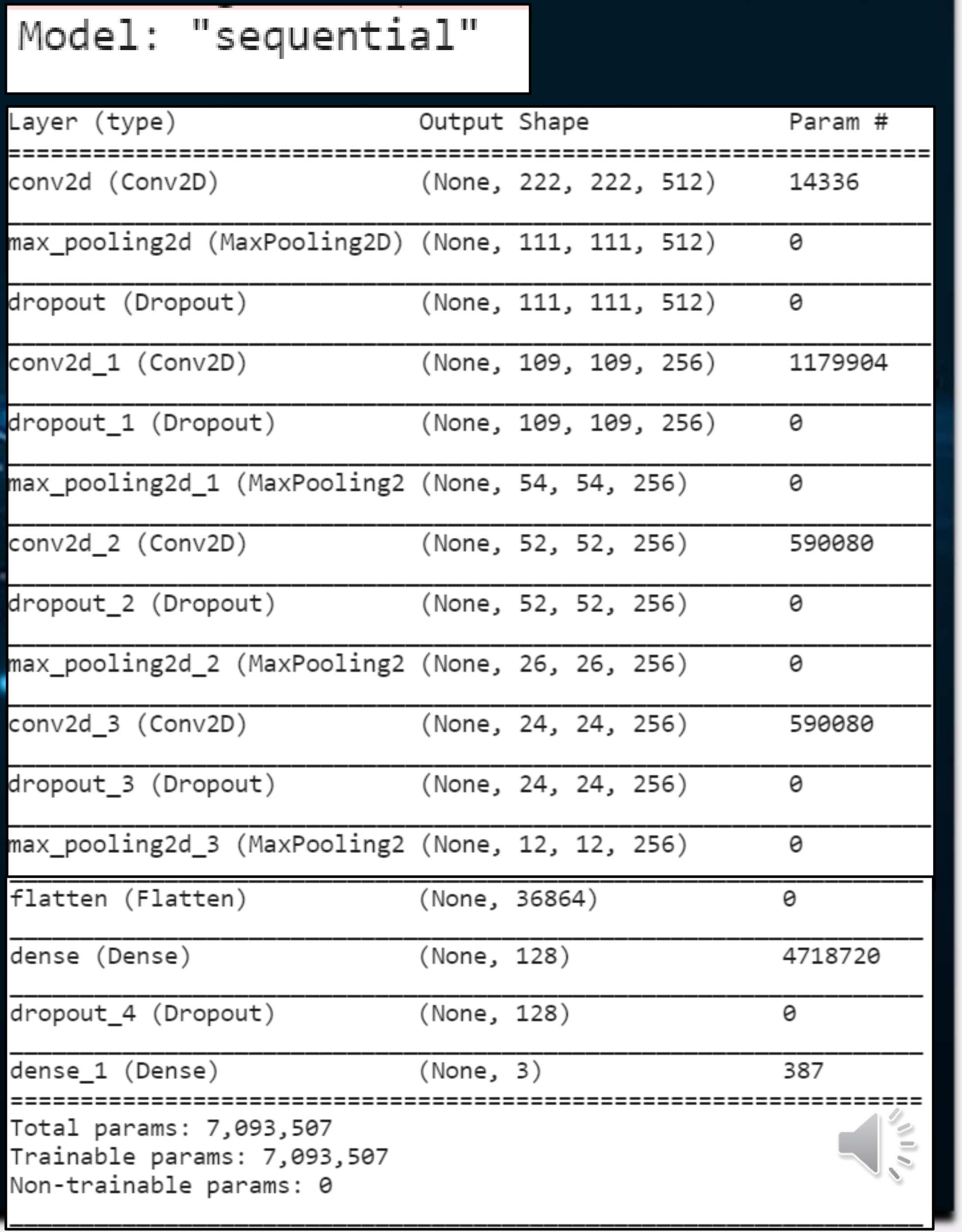
- A CNN basic architecture contains Convolutional layers , ReLU layers, Pooling Layers and a Fully connected layer.

- The sequential model was built with 4 Convolutional layers + ReLU, 4 Pooling layers and a fully connected layer. Dropout layer was used to prevent any overfitting. The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.

- **Test accuracy of 67% was achieved with this model**



```
Epoch 1/50
2022-03-02 19:41:40.846512: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
26/26 [==============================] - 27s 700ms/step - loss: 3.8353 - accuracy: 0.6085 - val_loss: 1.0246 - v
al_accuracy: 0.6953
Epoch 2/50
26/26 [==============================] - 11s 406ms/step - loss: 0.9037 - accuracy: 0.6768 - val_loss: 1.0092 - v
al_accuracy: 0.6687
Epoch 3/50
26/26 [==============================] - 10s 382ms/step - loss: 0.8960 - accuracy: 0.6768 - val_loss: 1.0285 - v
al_accuracy: 0.6641
Epoch 4/50
26/26 [==============================] - 10s 383ms/step - loss: 0.8675 - accuracy: 0.6768 - val_loss: 1.0149 - v
al_accuracy: 0.6875
Epoch 5/50
26/26 [==============================] - 10s 380ms/step - loss: 0.8884 - accuracy: 0.6768 - val_loss: 1.0405 - v
al_accuracy: 0.6625
Epoch 6/50
26/26 [==============================] - 10s 390ms/step - loss: 0.9052 - accuracy: 0.6768 - val_loss: 0.9823 - v
al_accuracy: 0.6703
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:2006: UserWarning: `Model.evaluate_generator` is
deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
test_acc: 0.671875
```



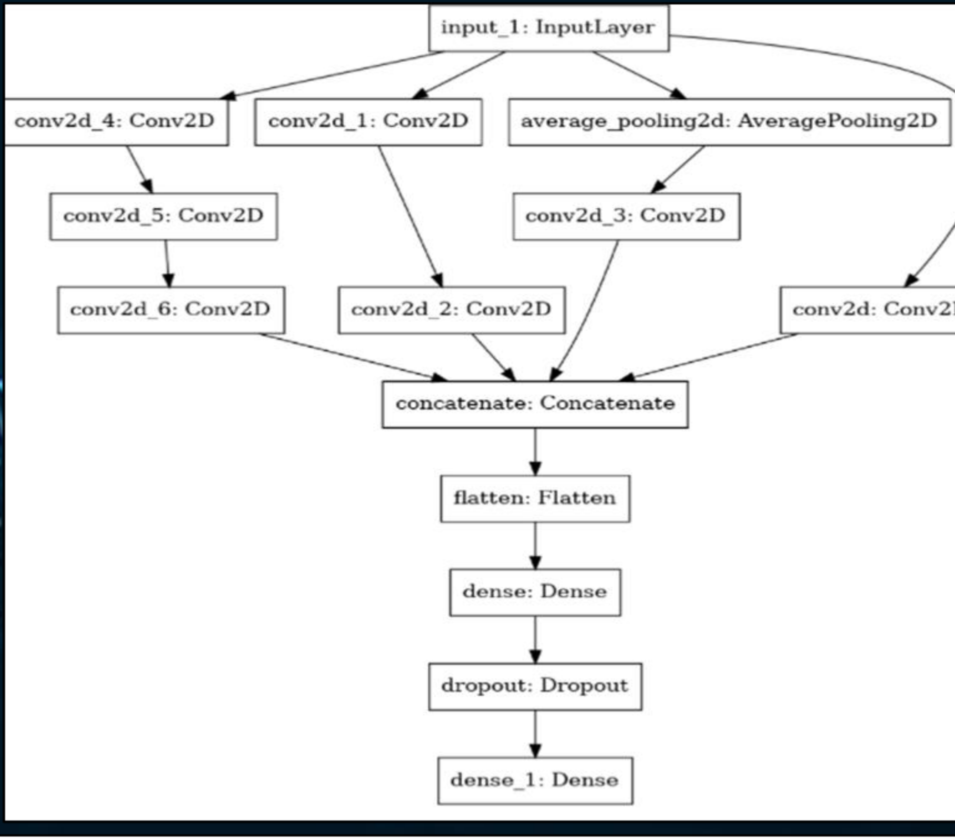
```
Model: "sequential"

Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 512)     14336
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 512)     0
_________________________________________________________________
dropout (Dropout)            (None, 111, 111, 512)     0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 256)     1179904
_________________________________________________________________
dropout_1 (Dropout)          (None, 109, 109, 256)     0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 256)       590080
_________________________________________________________________
dropout_2 (Dropout)          (None, 52, 52, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 256)       590080
_________________________________________________________________
dropout_3 (Dropout)          (None, 24, 24, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 36864)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4718720
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 7,093,507
Trainable params: 7,093,507
Non-trainable params: 0
_________________________________________________________________
```

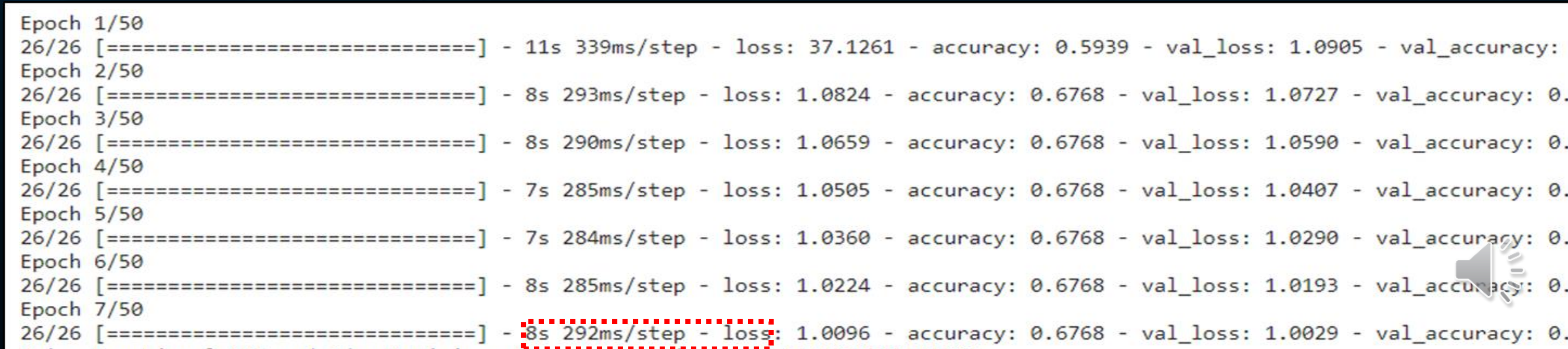
EVALUATION MODEL 2

- **Model-2** The functional API in Keras is an alternate way of creating models that offers a lot more flexibility, including creating more complex models.

- The model on the right was built using functional API to build classification model
- Not much improvement in accuracy obtained compared to sequential Model

- 67% improvement achieved with Functional API model





```
Epoch 1/50
26/26 [==============================] - 11s 339ms/step - loss: 37.1261 - accuracy: 0.5939 - val_loss: 1.0905 - val_accuracy:
Epoch 2/50
26/26 [==============================] - 8s 293ms/step - loss: 1.0824 - accuracy: 0.6768 - val_loss: 1.0727 - val_accuracy: 0.
Epoch 3/50
26/26 [==============================] - 8s 290ms/step - loss: 1.0659 - accuracy: 0.6768 - val_loss: 1.0590 - val_accuracy: 0.
Epoch 4/50
26/26 [==============================] - 7s 285ms/step - loss: 1.0505 - accuracy: 0.6768 - val_loss: 1.0407 - val_accuracy: 0.
Epoch 5/50
26/26 [==============================] - 7s 284ms/step - loss: 1.0360 - accuracy: 0.6768 - val_loss: 1.0290 - val_accuracy: 0.
Epoch 6/50
26/26 [==============================] - 8s 285ms/step - loss: 1.0224 - accuracy: 0.6768 - val_loss: 1.0193 - val_accuracy: 0.
Epoch 7/50
26/26 [==============================] - 8s 292ms/step - loss: 1.0096 - accuracy: 0.6768 - val_loss: 1.0029 - val_accuracy: 0.
```

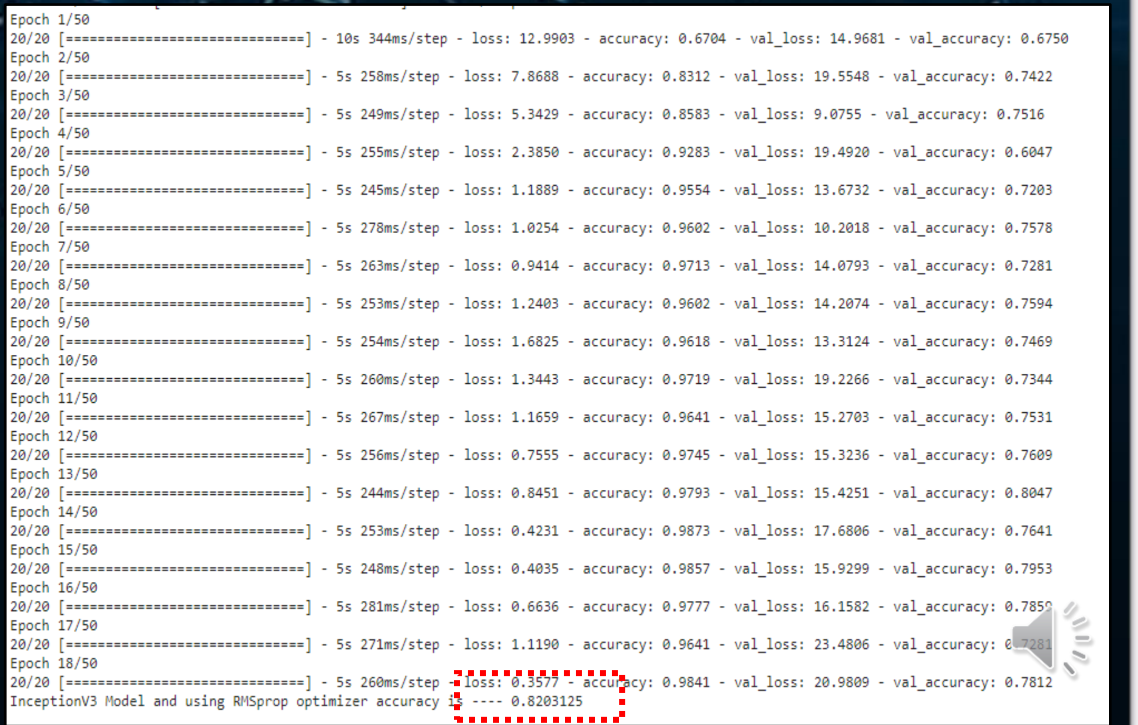
EVALUATION MODEL 3

- For Model-3, I used Transfer learning InceptionV3 model.

- Transfer learning for machine learning is when existing models are reused to solve a new challenge or problem.

- Inception is a convolutional neural network architecture introduced by Google which achieved top results in ImageNet Large Scale Visual Recognition Challenge 2014.

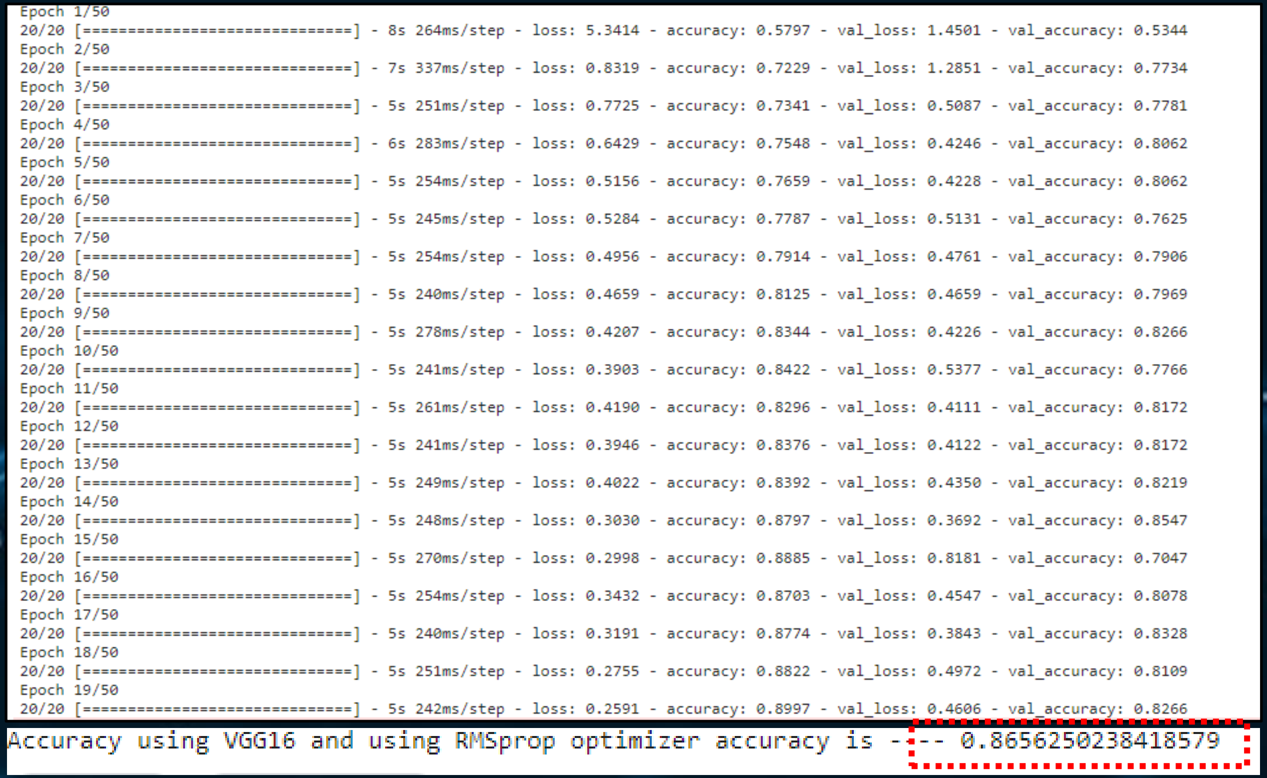
- Using InceptionV3 model an accuracy of 82% was achieved



```
Epoch 1/50
20/20 [==============================] - 10s 344ms/step - loss: 12.9903 - accuracy: 0.6704 - val_loss: 14.9681 - val_accuracy: 0.6750
Epoch 2/50
20/20 [==============================] - 5s 258ms/step - loss: 7.8688 - accuracy: 0.8312 - val_loss: 19.5548 - val_accuracy: 0.7422
Epoch 3/50
20/20 [==============================] - 5s 249ms/step - loss: 5.3429 - accuracy: 0.8583 - val_loss: 9.0755 - val_accuracy: 0.7516
Epoch 4/50
20/20 [==============================] - 5s 255ms/step - loss: 2.3850 - accuracy: 0.9283 - val_loss: 19.4920 - val_accuracy: 0.6047
Epoch 5/50
20/20 [==============================] - 5s 245ms/step - loss: 1.1889 - accuracy: 0.9554 - val_loss: 13.6732 - val_accuracy: 0.7203
Epoch 6/50
20/20 [==============================] - 5s 278ms/step - loss: 1.0254 - accuracy: 0.9602 - val_loss: 10.2018 - val_accuracy: 0.7578
Epoch 7/50
20/20 [==============================] - 5s 263ms/step - loss: 0.9414 - accuracy: 0.9713 - val_loss: 14.0793 - val_accuracy: 0.7281
Epoch 8/50
20/20 [==============================] - 5s 253ms/step - loss: 1.2403 - accuracy: 0.9602 - val_loss: 14.2074 - val_accuracy: 0.7594
Epoch 9/50
20/20 [==============================] - 5s 254ms/step - loss: 1.6825 - accuracy: 0.9618 - val_loss: 13.3124 - val_accuracy: 0.7469
Epoch 10/50
20/20 [==============================] - 5s 260ms/step - loss: 1.3443 - accuracy: 0.9719 - val_loss: 19.2266 - val_accuracy: 0.7344
Epoch 11/50
20/20 [==============================] - 5s 267ms/step - loss: 1.1659 - accuracy: 0.9641 - val_loss: 15.2703 - val_accuracy: 0.7531
Epoch 12/50
20/20 [==============================] - 5s 256ms/step - loss: 0.7555 - accuracy: 0.9745 - val_loss: 15.3236 - val_accuracy: 0.7609
Epoch 13/50
20/20 [==============================] - 5s 244ms/step - loss: 0.8451 - accuracy: 0.9793 - val_loss: 15.4251 - val_accuracy: 0.8047
Epoch 14/50
20/20 [==============================] - 5s 253ms/step - loss: 0.4231 - accuracy: 0.9873 - val_loss: 17.6806 - val_accuracy: 0.7641
Epoch 15/50
20/20 [==============================] - 5s 248ms/step - loss: 0.4035 - accuracy: 0.9857 - val_loss: 15.9299 - val_accuracy: 0.7953
Epoch 16/50
20/20 [==============================] - 5s 281ms/step - loss: 0.6636 - accuracy: 0.9777 - val_loss: 16.1582 - val_accuracy: 0.7859
Epoch 17/50
20/20 [==============================] - 5s 271ms/step - loss: 1.1190 - accuracy: 0.9641 - val_loss: 23.4806 - val_accuracy: 0.7281
Epoch 18/50
20/20 [==============================] - 5s 260ms/step - loss: 0.3577 - accuracy: 0.9841 - val_loss: 20.9809 - val_accuracy: 0.7812
InceptionV3 Model and using RMSprop optimizer accuracy is ---- 0.8203125
```

EVALUATION MODEL 4

- For Model-4 VGG16 model, which is a convolutional neural network trained on 1.2 million images to classify 1000 different categories was used.
- The best accuracy was seen using VGG16 model
- Accuracy of 86.5% was seen with VGG16



```
Epoch 1/50
20/20 [==============================] - 8s 264ms/step - loss: 5.3414 - accuracy: 0.5797 - val_loss: 1.4501 - val_accuracy: 0.5344
Epoch 2/50
20/20 [==============================] - 7s 337ms/step - loss: 0.8319 - accuracy: 0.7229 - val_loss: 1.2851 - val_accuracy: 0.7734
Epoch 3/50
20/20 [==============================] - 5s 251ms/step - loss: 0.7725 - accuracy: 0.7341 - val_loss: 0.5087 - val_accuracy: 0.7781
Epoch 4/50
20/20 [==============================] - 6s 283ms/step - loss: 0.6429 - accuracy: 0.7548 - val_loss: 0.4246 - val_accuracy: 0.8062
Epoch 5/50
20/20 [==============================] - 5s 254ms/step - loss: 0.5156 - accuracy: 0.7659 - val_loss: 0.4228 - val_accuracy: 0.8062
Epoch 6/50
20/20 [==============================] - 5s 245ms/step - loss: 0.5284 - accuracy: 0.7787 - val_loss: 0.5131 - val_accuracy: 0.7625
Epoch 7/50
20/20 [==============================] - 5s 254ms/step - loss: 0.4956 - accuracy: 0.7914 - val_loss: 0.4761 - val_accuracy: 0.7906
Epoch 8/50
20/20 [==============================] - 5s 240ms/step - loss: 0.4659 - accuracy: 0.8125 - val_loss: 0.4659 - val_accuracy: 0.7969
Epoch 9/50
20/20 [==============================] - 5s 278ms/step - loss: 0.4207 - accuracy: 0.8344 - val_loss: 0.4226 - val_accuracy: 0.8266
Epoch 10/50
20/20 [==============================] - 5s 241ms/step - loss: 0.3903 - accuracy: 0.8422 - val_loss: 0.5377 - val_accuracy: 0.7766
Epoch 11/50
20/20 [==============================] - 5s 261ms/step - loss: 0.4190 - accuracy: 0.8296 - val_loss: 0.4111 - val_accuracy: 0.8172
Epoch 12/50
20/20 [==============================] - 5s 241ms/step - loss: 0.3946 - accuracy: 0.8376 - val_loss: 0.4122 - val_accuracy: 0.8172
Epoch 13/50
20/20 [==============================] - 5s 249ms/step - loss: 0.4022 - accuracy: 0.8392 - val_loss: 0.4350 - val_accuracy: 0.8219
Epoch 14/50
20/20 [==============================] - 5s 248ms/step - loss: 0.3030 - accuracy: 0.8797 - val_loss: 0.3692 - val_accuracy: 0.8547
Epoch 15/50
20/20 [==============================] - 5s 270ms/step - loss: 0.2998 - accuracy: 0.8885 - val_loss: 0.8181 - val_accuracy: 0.7047
Epoch 16/50
20/20 [==============================] - 5s 254ms/step - loss: 0.3432 - accuracy: 0.8703 - val_loss: 0.4547 - val_accuracy: 0.8078
Epoch 17/50
20/20 [==============================] - 5s 240ms/step - loss: 0.3191 - accuracy: 0.8774 - val_loss: 0.3843 - val_accuracy: 0.8328
Epoch 18/50
20/20 [==============================] - 5s 251ms/step - loss: 0.2755 - accuracy: 0.8822 - val_loss: 0.4972 - val_accuracy: 0.8109
Epoch 19/50
20/20 [==============================] - 5s 242ms/step - loss: 0.2591 - accuracy: 0.8997 - val_loss: 0.4606 - val_accuracy: 0.8266
Accuracy using VGG16 and using RMSprop optimizer accuracy is ---- 0.865625023841857
```

VGG16 Architecture

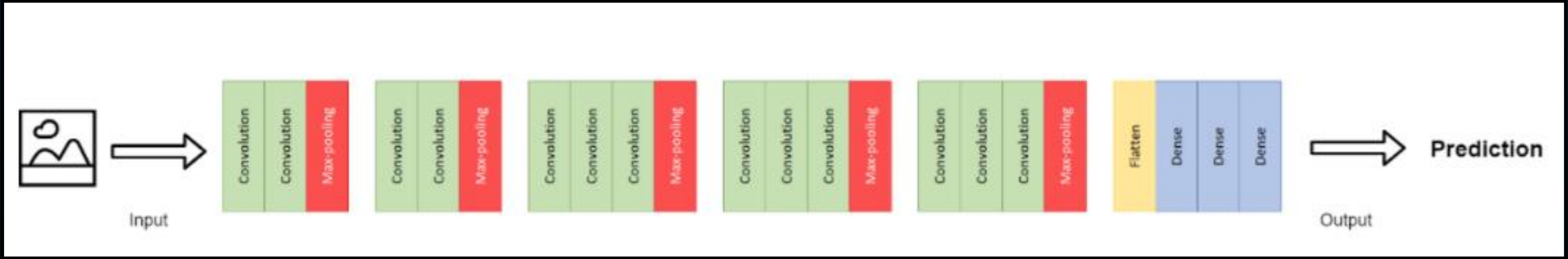
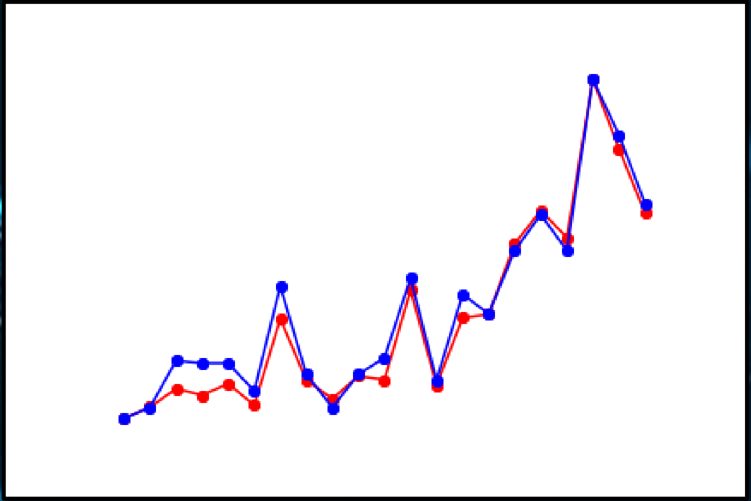


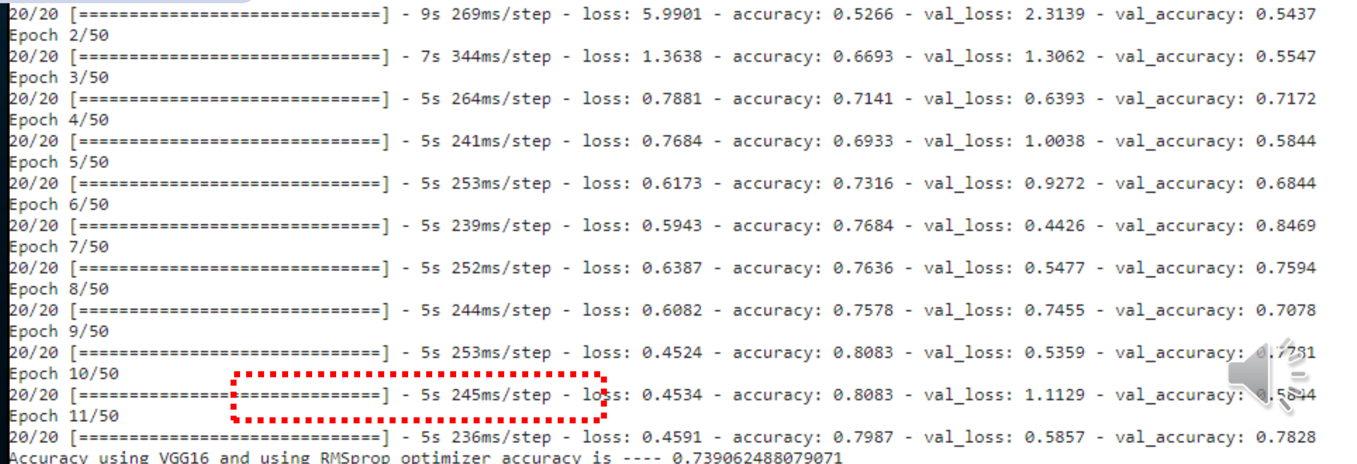
Image retrieved from https://towardsdatascience.com/transfer-learning-with-vgg16-and-keras-50ea161580b4

PROJECT PART-2 EVALUATION MODEL 1

- In Part-2 of the project, I generated images with line charts representing 2 features(Call Drop% and Call Drops) . Sample Image shown on the right
- Images were manually classified into Degraded, Improvement , Neutral Category and further divided into Train , Validation and Test Data.
- Transfer learning Model VGG16 was used for classification since it gave the best accuracy in part 1 of the project
- The accuracy achieved was 73% which is less than the accuracy seen in part 1 of the project which was classifying images with one feature(Call Drop%)

Sample Image(Category Degraded)





```
20/20 [==============================] - 9s 269ms/step - loss: 5.9901 - accuracy: 0.5266 - val_loss: 2.3139 - val_accuracy: 0.5437
Epoch 2/50
20/20 [==============================] - 7s 344ms/step - loss: 1.3638 - accuracy: 0.6693 - val_loss: 1.3062 - val_accuracy: 0.5547
Epoch 3/50
20/20 [==============================] - 5s 264ms/step - loss: 0.7881 - accuracy: 0.7141 - val_loss: 0.6393 - val_accuracy: 0.7172
Epoch 4/50
20/20 [==============================] - 5s 241ms/step - loss: 0.7684 - accuracy: 0.6933 - val_loss: 1.0038 - val_accuracy: 0.5844
Epoch 5/50
20/20 [==============================] - 5s 253ms/step - loss: 0.6173 - accuracy: 0.7316 - val_loss: 0.9272 - val_accuracy: 0.6844
Epoch 6/50
20/20 [==============================] - 5s 239ms/step - loss: 0.5943 - accuracy: 0.7684 - val_loss: 0.4426 - val_accuracy: 0.8469
Epoch 7/50
20/20 [==============================] - 5s 252ms/step - loss: 0.6387 - accuracy: 0.7636 - val_loss: 0.5477 - val_accuracy: 0.7594
Epoch 8/50
20/20 [==============================] - 5s 244ms/step - loss: 0.6082 - accuracy: 0.7578 - val_loss: 0.7455 - val_accuracy: 0.7078
Epoch 9/50
20/20 [==============================] - 5s 253ms/step - loss: 0.4524 - accuracy: 0.8083 - val_loss: 0.5359 - val_accuracy: 0.7781
Epoch 10/50
20/20 [==============================] - 5s 245ms/step - loss: 0.4534 - accuracy: 0.8083 - val_loss: 1.1129 - val_accuracy: 0.5844
Epoch 11/50
20/20 [==============================] - 5s 236ms/step - loss: 0.4591 - accuracy: 0.7987 - val_loss: 0.5857 - val_accuracy: 0.7828
Accuracy using VGG16 and using RMSprop optimizer accuracy is ---- 0.739062488079071
```

CONCLUSION

- Multiple Models were tried to classify images into Improved, Degraded, Neutral Category
- VGG16 Model was able to classify images into 3 categories with 86.5% accuracy
- VGG16 Model was used to classify images with 2 features into 3 categories but could only achieve 73% accuracy
- Most of the time was spent in Manual Classification of images for preparing Train/Validation/Test data and possible human error is expected that can impact accuracy negatively.
- Further efforts to be put in to improve accuracy for classifying images with 2 features
- Once further improvement in accuracy in observed, the deployment can be considered so that existing methodology can be replaced with current methodology

THANK YOU