

**Project Development Phase  
Model Performance Test**

Date	17 November 2022
Team ID	PNT2022TMID25758
Project Name	Project - Digital Naturalist - AI Enabled tool for Biodiversity Researchers
Maximum Marks	10 Marks

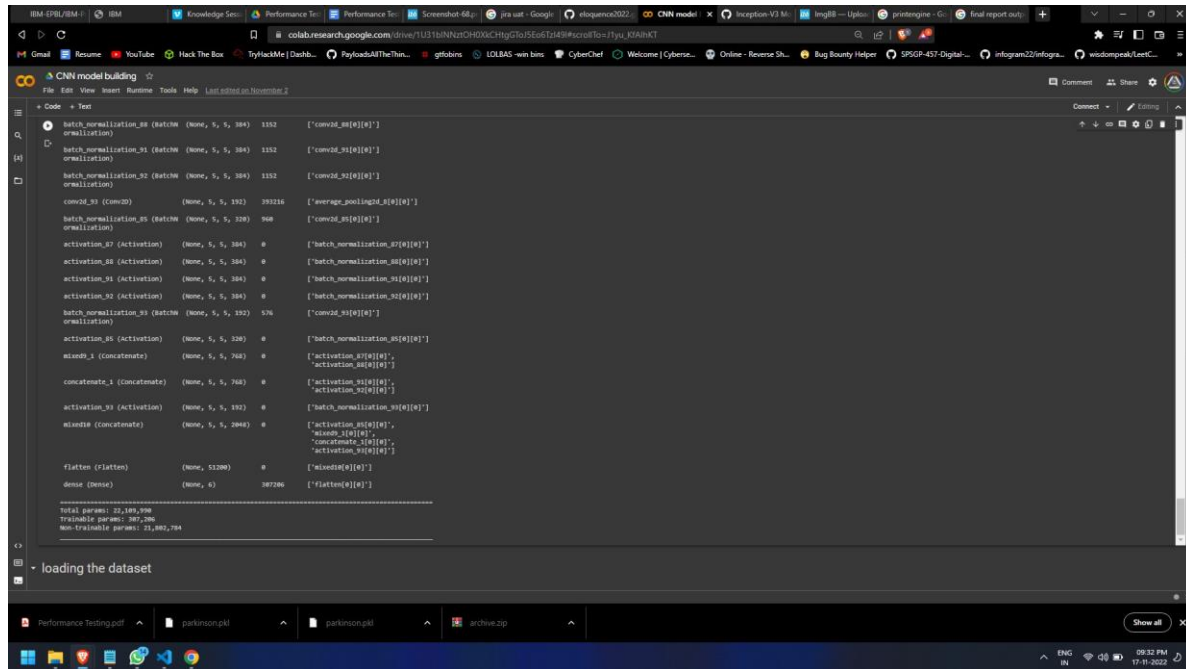
**Model Performance Testing:**

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	<b>Total params: 22,109,990</b> <b>Trainable params: 307,206</b> <b>Non-trainable params: 21,802,784</b>	Screenshot 1
2.	Accuracy	Training Accuracy - 92.8%  Validation Accuracy - 85.6%	Screenshot 2

Screenshots - Please refer to the next page:

Screenshot 1 :

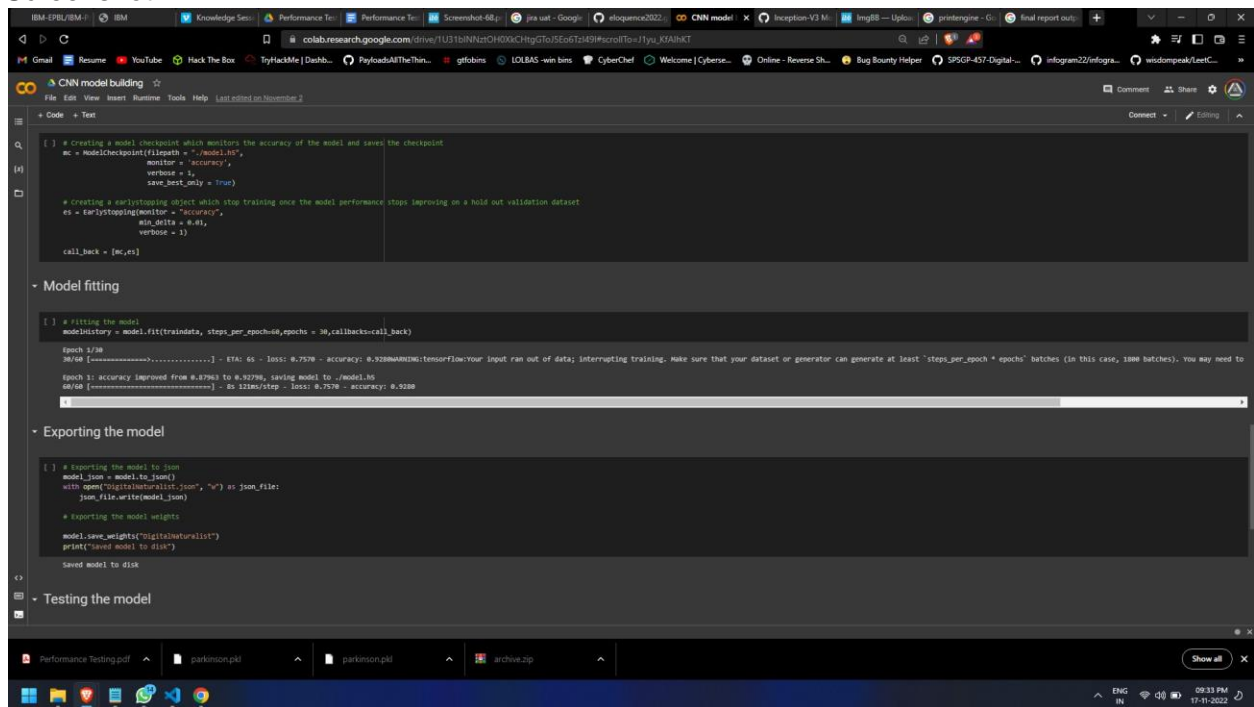


The screenshot shows a Jupyter Notebook titled "CNN model building". The code cell displays a detailed summary of a CNN model architecture. The summary lists layers such as batch\_normalization, conv2d, max\_pooling2d, and activation layers, along with their shapes and output dimensions. At the bottom, it provides a total parameter count of 22,145,794, with 147,248 trainable parameters and 21,998,546 non-trainable parameters.

```
batch_normalization_0 (Batch Normalization) (None, 5, 5, 384) 1332 ['conv2d_0[0][0]']
batch_normalization_1 (Batch Normalization) (None, 5, 5, 384) 1332 ['conv2d_1[0][0]']
batch_normalization_2 (Batch Normalization) (None, 5, 5, 384) 1332 ['conv2d_2[0][0]']
conv2d_3 (Conv2D) (None, 5, 5, 192) 39216 ['average_pooling2d_0[0][0]']
batch_normalization_3 (Batch Normalization) (None, 5, 5, 192) 608 ['conv2d_3[0][0]']
activation_0 (Activation) (None, 5, 5, 384) 0 ['batch_normalization_0[0][0]']
activation_1 (Activation) (None, 5, 5, 384) 0 ['batch_normalization_1[0][0]']
activation_2 (Activation) (None, 5, 5, 384) 0 ['batch_normalization_2[0][0]']
activation_3 (Activation) (None, 5, 5, 384) 0 ['batch_normalization_3[0][0]']
batch_normalization_4 (Batch Normalization) (None, 5, 5, 192) 608 ['conv2d_4[0][0]']
activation_4 (Activation) (None, 5, 5, 192) 0 ['batch_normalization_4[0][0]']
mixed_1 (Concatenate) (None, 5, 5, 768) 0 ['activation_4[0][0]', 'activation_5[0][0]']
concatenate_1 (Concatenate) (None, 5, 5, 768) 0 ['activation_6[0][0]', 'activation_7[0][0]']
activation_5 (Activation) (None, 5, 5, 192) 0 ['batch_normalization_5[0][0]']
mixed_2 (Concatenate) (None, 5, 5, 2048) 0 ['activation_8[0][0]', 'mixed_1[0][0]', 'concatenate_1[0][0]', 'activation_9[0][0]']
flatten (Flatten) (None, 51200) 0 ['mixed_2[0][0]']
dense (Dense) (None, 4) 167208 ['flatten[0][0]']

Total params: 22,145,794
Trainable params: 147,248
Non-trainable params: 21,998,546
```

Screenshot 2:



The screenshot shows a Jupyter Notebook titled "CNN model building" with three code cells. The first cell defines a model checkpoint and an early stopping monitor. The second cell fits the model and displays a progress bar. The third cell exports the model and weights to disk.

```
# Creating a model checkpoint which monitors the accuracy of the model and saves the checkpoint
mc = ModelCheckpoint(filepath = ".model.h5",
                    monitor = 'accuracy',
                    verbose = 1,
                    save_best_only = True)

# Creating a earlystopping object which stop training once the model performance stops improving on a hold out validation dataset
es = EarlyStoppingMonitor = "accuracy",
min_delta = 0.01,
verbose = 1)

call_back = [mc, es]
```

```
# Fitting the model
model_history = model.fit(traindata, steps_per_epoch=10, epochs = 30, callbacks=call_back)

Epoch 1/30
30/0 [-----] - ETA: 6s - loss: 0.7578 - accuracy: 0.9280WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least 'steps_per_epoch * epochs' batches (in this case, 1800 batches). You may need to
Epoch 1: accuracy improved from 0.87663 to 0.92798, saving model to .model.h5
00/0 [-----] - 8s 12ms/step - loss: 0.7578 - accuracy: 0.9280
```

```
# Exporting the model
model_json = model.to_json()
with open("digit recognizer.json", "w") as json_file:
    json_file.write(model_json)

# Exporting the model weights
model.save_weights("digit recognizer.h5")
print("Saved model to disk")

Saved model to disk
```