Project Development Phase

Sprint 2

Date	05 November 2022
Team ID	PNT2022TMID35899
Project Name	Digital Naturalist - AI Enabled Tool For Biodiversity Researchers
Maximum Marks	4 Marks

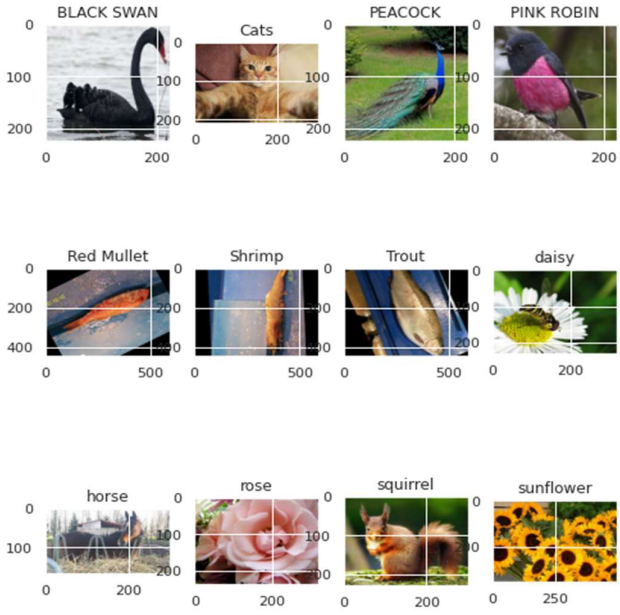
1. Indexing species classes

```
train_generator.class_indices
```

```
]: {'BLACK SWAN': 0,
    'Cats': 1,
    'PEACOCK': 2,
    'PINK ROBIN': 3,
    'Red Mullet': 4,
    'Shrimp': 5,
    'Trout': 6,
    'daisy': 7,
    'horse': 8,
    'rose': 9,
    'squirrel': 10,
    'sunflower': 11}
```

2. Sample Plot for each classes

```
from skimage import io
samples = ['/home/wsuser/work/re-rev/Training data/BLACK SWAN/001.jpg','/home/w
sample_names = list(train_generator.class_indices.keys())
x,axarr = plt.subplots(3,4,figsize=(8,10))
for i in range(3):
    for j in range(4):
        axarr[i][j].imshow(io.imread(samples[4*i+j]))
        axarr[i][j].title.set_text(sample_names[4*i+j])
```



3. CNN Model architecture and Compilation using Adam Optimizer

```
model = Sequential()
```

```
model.add(Convolution2D(16,kernel_size=(3,3),input_shape=(224,224,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(32,kernel_size=(3,3),input_shape=(224,224,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Convolution2D(64,kernel_size=(3,3),input_shape=(224,224,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Convolution2D(32,kernel_size=(3,3),input_shape=(224,224,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Flatten())

model.add(Dense(units=256,kernel_initializer="random_uniform",activation="relu"))
model.add(Dropout(0.4))

model.add(Dense(units=nb_classes,activation="softmax"))
```

```
model.compile(loss='categorical_crossentropy',optimizer = 'adam',metrics=[accuracy])
```

```
model.summary()
```

4. Summary of Model

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 222, 222, 16)      448

 max_pooling2d_12 (MaxPoolin  (None, 111, 111, 16)     0
 g2D)

 conv2d_13 (Conv2D)          (None, 109, 109, 32)      4640

 max_pooling2d_13 (MaxPoolin  (None, 54, 54, 32)       0
 g2D)

 dropout_12 (Dropout)        (None, 54, 54, 32)        0

 conv2d_14 (Conv2D)          (None, 52, 52, 64)        18496

 max_pooling2d_14 (MaxPoolin  (None, 26, 26, 64)       0
 g2D)

 dropout_13 (Dropout)        (None, 26, 26, 64)        0

 conv2d_15 (Conv2D)          (None, 24, 24, 32)        18464

 max_pooling2d_15 (MaxPoolin  (None, 12, 12, 32)       0
 g2D)

 dropout_14 (Dropout)        (None, 12, 12, 32)        0

 flatten_3 (Flatten)         (None, 4608)              0

 dense_5 (Dense)             (None, 256)               1179904

 dropout_15 (Dropout)        (None, 256)               0

 dense_6 (Dense)             (None, 12)                3084

=================================================================
Total params: 1,225,036
Trainable params: 1,225,036
Non-trainable params: 0
_________________________________________________________________
```

5. Training and Validation

```
history = model.fit_generator(train_generator,epochs=20, validation_data=valid_generator)
```

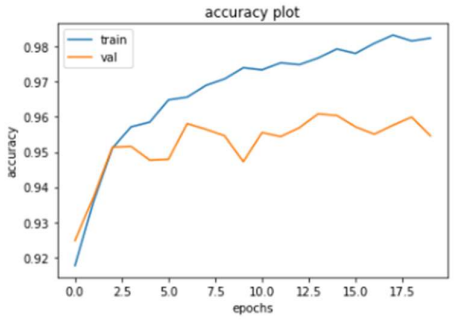
```
Epoch 1/20
180/180 [==============================] - 66s 361ms/step - loss: 2.2439 - accuracy: 0.9176 - val_loss: 1.7700 - val_accuracy: 0.9248
Epoch 2/20
180/180 [==============================] - 65s 360ms/step - loss: 1.3980 - accuracy: 0.9360 - val_loss: 1.2727 - val_accuracy: 0.9373
Epoch 3/20
180/180 [==============================] - 65s 360ms/step - loss: 1.0426 - accuracy: 0.9511 - val_loss: 0.9816 - val_accuracy: 0.9514
Epoch 4/20
180/180 [==============================] - 65s 361ms/step - loss: 0.8965 - accuracy: 0.9572 - val_loss: 0.9574 - val_accuracy: 0.9516
Epoch 5/20
180/180 [==============================] - 65s 360ms/step - loss: 0.8380 - accuracy: 0.9585 - val_loss: 0.9537 - val_accuracy: 0.9477
Epoch 6/20
180/180 [==============================] - 65s 359ms/step - loss: 0.6971 - accuracy: 0.9649 - val_loss: 0.8772 - val_accuracy: 0.9479
Epoch 7/20
180/180 [==============================] - 64s 357ms/step - loss: 0.6713 - accuracy: 0.9656 - val_loss: 0.7969 - val_accuracy: 0.9581
Epoch 8/20
180/180 [==============================] - 65s 360ms/step - loss: 0.6216 - accuracy: 0.9690 - val_loss: 0.7577 - val_accuracy: 0.9565
Epoch 9/20
180/180 [==============================] - 65s 362ms/step - loss: 0.5675 - accuracy: 0.9708 - val_loss: 0.8827 - val_accuracy: 0.9546
Epoch 10/20
180/180 [==============================] - 65s 359ms/step - loss: 0.5080 - accuracy: 0.9740 - val_loss: 0.9970 - val_accuracy: 0.9472
Epoch 11/20
180/180 [==============================] - 64s 356ms/step - loss: 0.4885 - accuracy: 0.9734 - val_loss: 0.8772 - val_accuracy: 0.9556
Epoch 12/20
180/180 [==============================] - 62s 345ms/step - loss: 0.4608 - accuracy: 0.9754 - val_loss: 0.8464 - val_accuracy: 0.9544
Epoch 13/20
180/180 [==============================] - 62s 346ms/step - loss: 0.4508 - accuracy: 0.9749 - val_loss: 0.8298 - val_accuracy: 0.9569
Epoch 14/20
180/180 [==============================] - 63s 348ms/step - loss: 0.4175 - accuracy: 0.9768 - val_loss: 0.7616 - val_accuracy: 0.9609
Epoch 15/20
180/180 [==============================] - 63s 347ms/step - loss: 0.3726 - accuracy: 0.9794 - val_loss: 0.7456 - val_accuracy: 0.9604
Epoch 16/20
180/180 [==============================] - 62s 346ms/step - loss: 0.4158 - accuracy: 0.9781 - val_loss: 0.8010 - val_accuracy: 0.9572
Epoch 17/20
180/180 [==============================] - 63s 347ms/step - loss: 0.3328 - accuracy: 0.9810 - val_loss: 0.8820 - val_accuracy: 0.9551
Epoch 18/20
180/180 [==============================] - 63s 347ms/step - loss: 0.3219 - accuracy: 0.9833 - val_loss: 0.8546 - val_accuracy: 0.9576
Epoch 19/20
180/180 [==============================] - 63s 348ms/step - loss: 0.3157 - accuracy: 0.9817 - val_loss: 0.8109 - val_accuracy: 0.9600
Epoch 20/20
180/180 [==============================] - 62s 346ms/step - loss: 0.3273 - accuracy: 0.9825 - val_loss: 1.0102 - val_accuracy: 0.9546
```

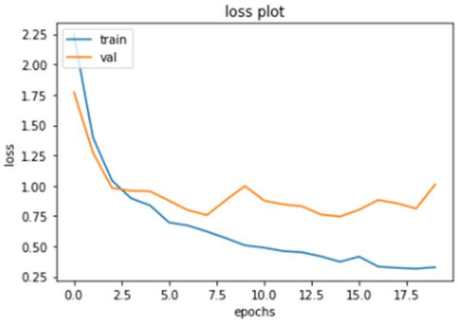
6. Saving the model:

```
model.save("83_per_cnn_dig_nat.h5")
```

```
model_json=model.to_json()
with open("model-bw.json","w") as json_file:
  json_file.write(model_json)
```

7. Plots for training vs validation accuracies and losses





8. Testing the model with the Test dataset

```
test_datagen = ImageDataGenerator(rescale=1. / 255)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=180, # The number of test images
    class_mode='categorical')
```

```
x_test, y_test = test_generator.__getitem__(0)
```

```
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred,axis=1)
```

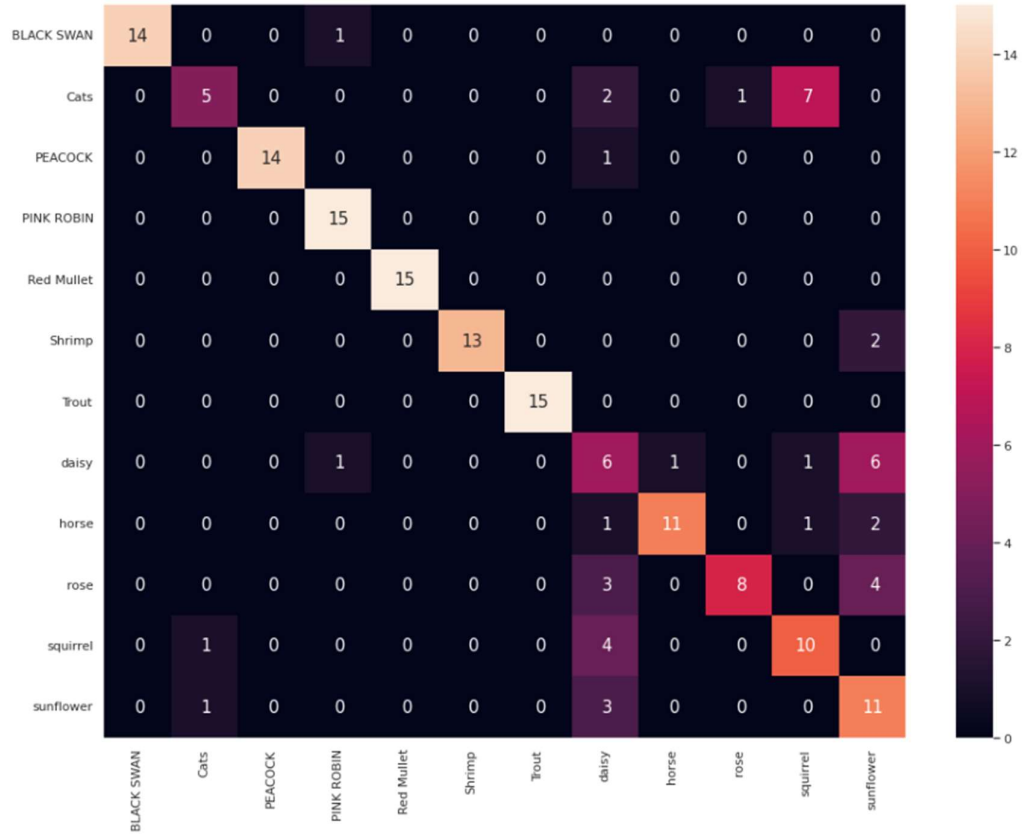
```
y_test = np.argmax(y_test, axis=1)
```

9. Classification Report with F1 score

CNN Model Accuracy on test set: 0.7611

	precision	recall	f1-score	support
0	1.00	0.93	0.97	15
1	0.71	0.33	0.45	15
2	1.00	0.93	0.97	15
3	0.88	1.00	0.94	15
4	1.00	1.00	1.00	15
5	1.00	0.87	0.93	15
6	1.00	1.00	1.00	15
7	0.30	0.40	0.34	15
8	0.92	0.73	0.81	15
9	0.89	0.53	0.67	15
10	0.53	0.67	0.59	15
11	0.44	0.73	0.55	15
accuracy			0.76	180
macro avg	0.81	0.76	0.77	180
weighted avg	0.81	0.76	0.77	180

10. Confusion Matrix



11. **Model Accuracy after Testing the model**

```
acc = np.count_nonzero(np.equal(y_pred,y_test))/x_test.shape[0]
print(acc)
```

```
0.7611111111111111
```

Notebook link: https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/d8cf79ed-e0b3-44dd-b150-cf2ececc025d?projectid=9f99f93d-6e5b-44eb-ad3d-f2133b037f06&context=cpdaas#