Model Development Phase Template

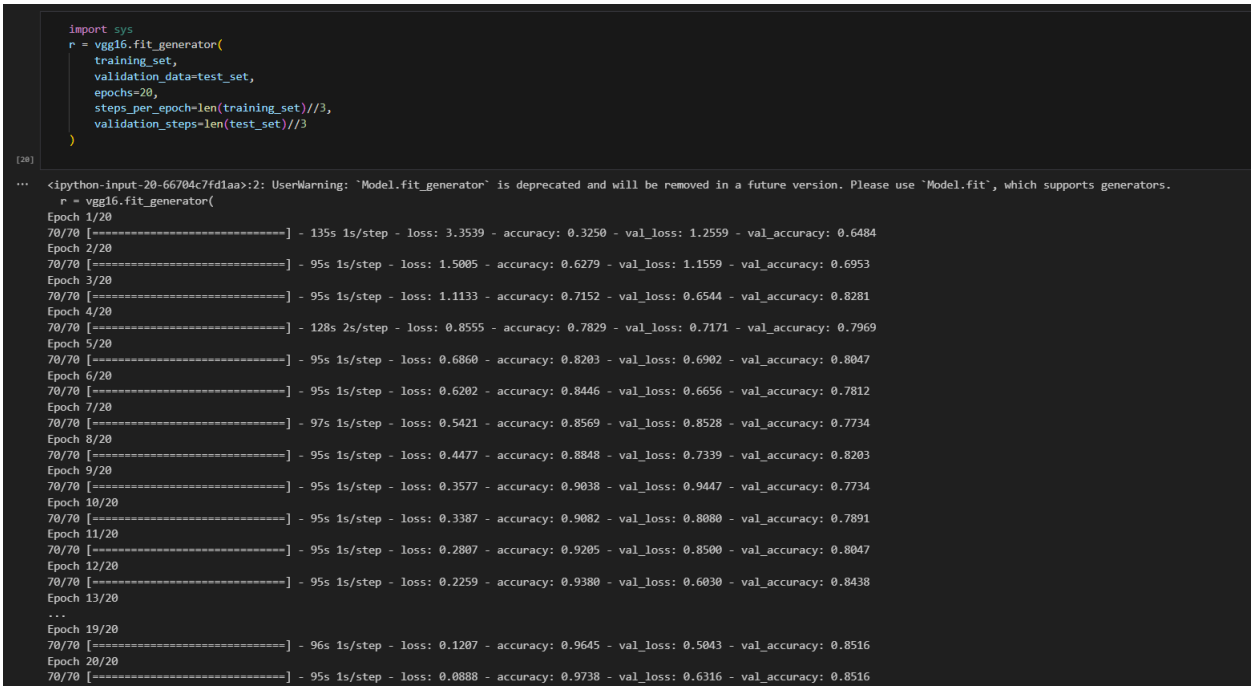
Date	10 July 2024
Team ID	SWTID1720158677
Project Title	SportSpecs: Unraveling Athletic Prowess With Advanced Transfer Learning For Sports.
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Model 1:- (VGG16)

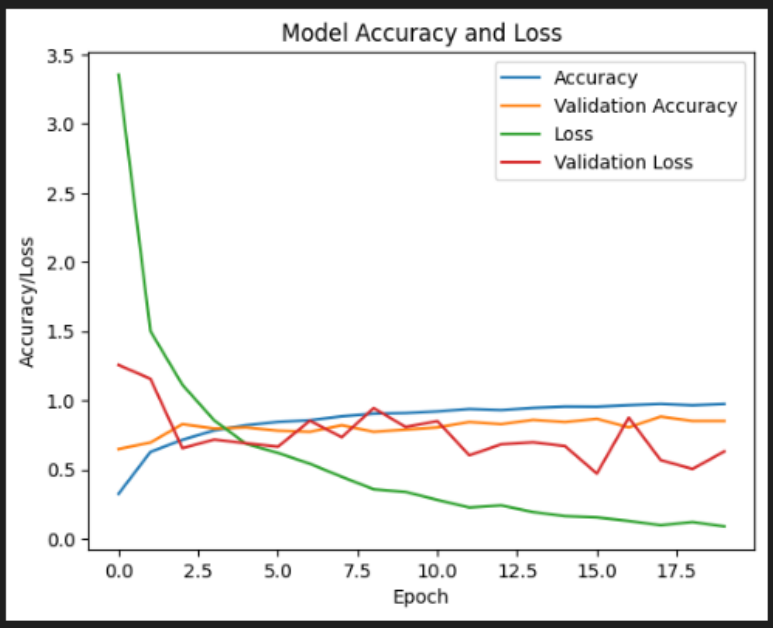
Initial Model Training Code (5 marks) :



```
import sys
r = vgg16.fit_generator(
    training_set,
    validation_data=test_set,
    epochs=20,
    steps_per_epoch=len(training_set)//3,
    validation_steps=len(test_set)//3
)
```

```
[20]
...
<ipython-input-20-66704c7fd1aa>:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  r = vgg16.fit_generator(
Epoch 1/20
70/70 [==============================] - 135s 1s/step - loss: 3.3539 - accuracy: 0.3250 - val_loss: 1.2559 - val_accuracy: 0.6484
Epoch 2/20
70/70 [==============================] - 95s 1s/step - loss: 1.5005 - accuracy: 0.6279 - val_loss: 1.1559 - val_accuracy: 0.6953
Epoch 3/20
70/70 [==============================] - 95s 1s/step - loss: 1.1133 - accuracy: 0.7152 - val_loss: 0.6544 - val_accuracy: 0.8281
Epoch 4/20
70/70 [==============================] - 128s 2s/step - loss: 0.8555 - accuracy: 0.7829 - val_loss: 0.7171 - val_accuracy: 0.7969
Epoch 5/20
70/70 [==============================] - 95s 1s/step - loss: 0.6860 - accuracy: 0.8203 - val_loss: 0.6902 - val_accuracy: 0.8047
Epoch 6/20
70/70 [==============================] - 95s 1s/step - loss: 0.6202 - accuracy: 0.8446 - val_loss: 0.6656 - val_accuracy: 0.7812
Epoch 7/20
70/70 [==============================] - 97s 1s/step - loss: 0.5421 - accuracy: 0.8569 - val_loss: 0.8528 - val_accuracy: 0.7734
Epoch 8/20
70/70 [==============================] - 95s 1s/step - loss: 0.4477 - accuracy: 0.8848 - val_loss: 0.7339 - val_accuracy: 0.8203
Epoch 9/20
70/70 [==============================] - 95s 1s/step - loss: 0.3577 - accuracy: 0.9038 - val_loss: 0.9447 - val_accuracy: 0.7734
Epoch 10/20
70/70 [==============================] - 95s 1s/step - loss: 0.3387 - accuracy: 0.9082 - val_loss: 0.8080 - val_accuracy: 0.7891
Epoch 11/20
70/70 [==============================] - 95s 1s/step - loss: 0.2807 - accuracy: 0.9205 - val_loss: 0.8500 - val_accuracy: 0.8047
Epoch 12/20
70/70 [==============================] - 95s 1s/step - loss: 0.2259 - accuracy: 0.9380 - val_loss: 0.6030 - val_accuracy: 0.8438
Epoch 13/20
...
Epoch 19/20
70/70 [==============================] - 96s 1s/step - loss: 0.1207 - accuracy: 0.9645 - val_loss: 0.5043 - val_accuracy: 0.8516
Epoch 20/20
70/70 [==============================] - 95s 1s/step - loss: 0.0888 - accuracy: 0.9738 - val_loss: 0.6316 - val_accuracy: 0.8516
```

```python
import matplotlib.pyplot as plt

# Plotting accuracy
plt.plot(r.history["accuracy"])
plt.plot(r.history['val_accuracy'])

# Plotting loss
plt.plot(r.history['loss'])
plt.plot(r.history['val_loss'])

# Adding title and labels
plt.title("Model Accuracy and Loss")
plt.ylabel("Accuracy/Loss")
plt.xlabel("Epoch")

# Adding legend
plt.legend(["Accuracy", "Validation Accuracy", "Loss", "Validation Loss"])

# Displaying plot
plt.show()
```




```
[23]  vgg16.save("project1.h5")                                                                    Python

...   /usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered lega
        saving_api.save_model(
```

```
[25]
#import load_model class for loading h5 file
from tensorflow.keras.models import load_model
#import image class to process the images
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
import numpy as np
```

```
[26]
#load saved vgg 16 model file
model=load_model("project1.h5")
```

```
[27]
# test accuracy
print('Test Score',model.evaluate(test_set))
```

```
8/8 [==============================] - 26s 2s/step - loss: 0.6175 - accuracy: 0.8500
Test Score [0.6175491809844971, 0.8500000238418579]
```

```
[28]
# train accuracy
print('Train Score',model.evaluate(training_set))
```

```
211/211 [==============================] - 199s 941ms/step - loss: 0.0676 - accuracy: 0.9819
Train Score [0.06758040934801102, 0.9819152355194092]
```

```
[39]
img=image.load_img("/content/test/sky surfing/4.jpg", target_size=(224,224))
#convert image to array format
x=image.img_to_array(img)
import numpy as np
x=np.expand_dims(x,axis=0)
img_data=preprocess_input(x)
output=np.argmax(model.predict(img_data), axis=1)
index=['air hockey', 'ampute football', 'archery', 'arm wrestling', 'axe throwing',
    'balance beam', 'barell racing', 'baseball', 'basketball', 'baton twirling',
    'bike polo', 'billiards', 'bmx', 'bobsled', 'bowling', 'boxing', 'bull riding',
    'bungee jumping', 'canoe slamon', 'cheerleading', 'chuckwagon racing', 'cricket',
    'croquet', 'curling', 'disc golf', 'fencing', 'field hockey', 'figure skating men',
    'figure skating pairs', 'figure skating women', 'fly fishing', 'football',
    'formula 1 racing', 'frisbee', 'gaga', 'giant slalom', 'golf', 'hammer throw',
    'hang gliding', 'harness racing', 'high jump', 'hockey', 'horse jumping',
    'horse racing', 'horseshoe pitching', 'hurdles', 'hydroplane racing', 'ice climbing',
    'ice yachting', 'jai alai', 'javelin', 'jousting', 'judo', 'lacrosse', 'log rolling',
    'luge', 'motorcycle racing', 'mushing', 'nascar racing', 'olympic wrestling',
    'parallel bar', 'pole climbing', 'pole dancing', 'pole vault', 'polo', 'pommel horse',
    'rings', 'rock climbing', 'roller derby', 'rollerblade racing', 'rowing', 'rugby',
    'sailboat racing', 'shot put', 'shuffleboard', 'sidecar racing', 'ski jumping',
    'sky surfing', 'skydiving', 'snow boarding', 'snowmobile racing', 'speed skating',
    'steer wrestling', 'sumo wrestling', 'surfing', 'swimming', 'table tennis', 'tennis',
    'track bicycle', 'trapeze', 'tug of war', 'ultimate', 'uneven bars', 'volleyball',
    'water cycling', 'water polo', 'weightlifting', 'wheelchair basketball',
    'wheelchair racing', 'wingsuit flying']
result = str(index[output[0]])
result
```

```
1/1 [==============================] - 0s 19ms/step
```

```
'sky surfing'
```

Model 2:- (VGG19)

Initial Model Training Code :

```
[23]
import sys
r = vgg19.fit_generator(
    training_set,
    validation_data=test_set,
    epochs=20,
    steps_per_epoch=len(training_set)//3,
    validation_steps=len(test_set)//3
)
```

```
<ipython-input-23-a33db2704ae1>:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  r = vgg19.fit_generator(
Epoch 1/20
70/70 [==============================] - 136s 1s/step - loss: 3.8137 - accuracy: 0.2828 - val_loss: 1.7766 - val_accuracy: 0.5859
Epoch 2/20
70/70 [==============================] - 98s 1s/step - loss: 1.7551 - accuracy: 0.5824 - val_loss: 1.2456 - val_accuracy: 0.6562
Epoch 3/20
70/70 [==============================] - 130s 2s/step - loss: 1.2835 - accuracy: 0.6791 - val_loss: 1.2765 - val_accuracy: 0.6719
Epoch 4/20
70/70 [==============================] - 98s 1s/step - loss: 0.9734 - accuracy: 0.7484 - val_loss: 1.1409 - val_accuracy: 0.6719
Epoch 5/20
70/70 [==============================] - 98s 1s/step - loss: 0.7942 - accuracy: 0.7980 - val_loss: 0.7637 - val_accuracy: 0.7500
Epoch 6/20
70/70 [==============================] - 98s 1s/step - loss: 0.6965 - accuracy: 0.8125 - val_loss: 0.9722 - val_accuracy: 0.7344
Epoch 7/20
70/70 [==============================] - 97s 1s/step - loss: 0.6761 - accuracy: 0.8254 - val_loss: 0.9852 - val_accuracy: 0.7500
Epoch 8/20
70/70 [==============================] - 98s 1s/step - loss: 0.5491 - accuracy: 0.8556 - val_loss: 0.8887 - val_accuracy: 0.7656
Epoch 9/20
70/70 [==============================] - 97s 1s/step - loss: 0.5386 - accuracy: 0.8550 - val_loss: 0.7471 - val_accuracy: 0.8203
Epoch 10/20
70/70 [==============================] - 98s 1s/step - loss: 0.4497 - accuracy: 0.8767 - val_loss: 0.6678 - val_accuracy: 0.7969
Epoch 11/20
70/70 [==============================] - 98s 1s/step - loss: 0.3739 - accuracy: 0.8913 - val_loss: 1.0400 - val_accuracy: 0.7812
Epoch 12/20
70/70 [==============================] - 98s 1s/step - loss: 0.3241 - accuracy: 0.9096 - val_loss: 0.4723 - val_accuracy: 0.9062
Epoch 13/20
...
Epoch 19/20
70/70 [==============================] - 98s 1s/step - loss: 0.1747 - accuracy: 0.9496 - val_loss: 0.7033 - val_accuracy: 0.8047
Epoch 20/20
70/70 [==============================] - 98s 1s/step - loss: 0.1597 - accuracy: 0.9569 - val_loss: 0.8032 - val_accuracy: 0.7500
```

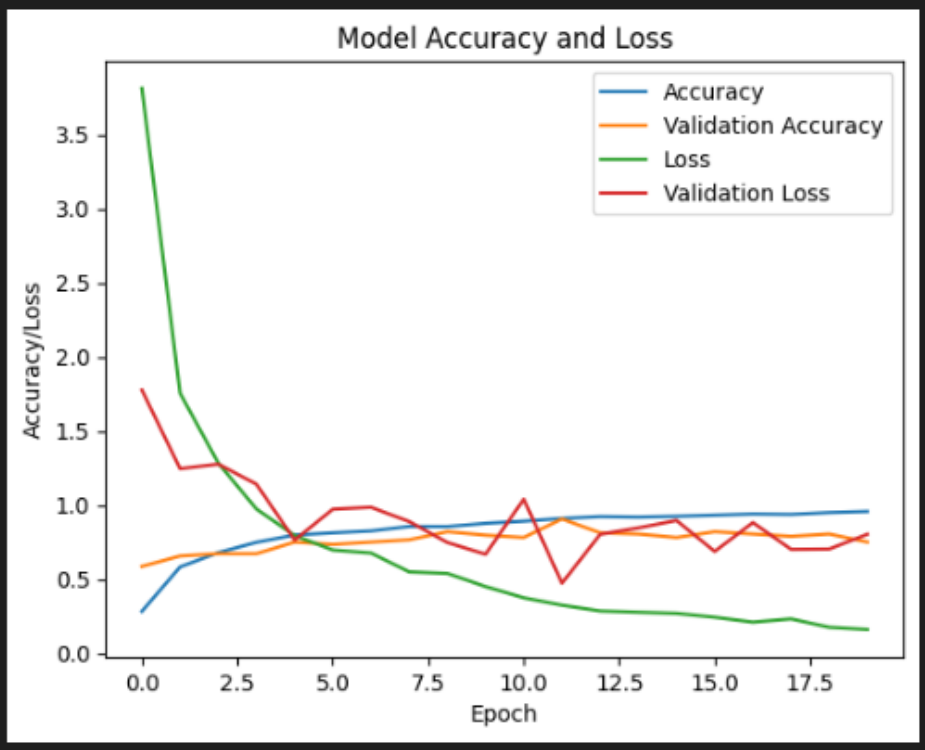
```
[25]
vgg19.save("project_vgg19.h5")
```

Python

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered leg
  saving_api.save_model(
```

```
import matplotlib.pyplot as plt

# Plotting accuracy
plt.plot(r.history["accuracy"])
plt.plot(r.history['val_accuracy'])

# Plotting loss
plt.plot(r.history['loss'])
plt.plot(r.history['val_loss'])

# Adding title and labels
plt.title("Model Accuracy and Loss")
plt.ylabel("Accuracy/Loss")
plt.xlabel("Epoch")

# Adding legend
plt.legend(["Accuracy", "Validation Accuracy", "Loss", "Validation Loss"])

# Displaying plot
plt.show()
```

[24]



```
[27]
#import load_model class for loading h5 file
from tensorflow.keras.models import load_model
#import image class to process the images
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
import numpy as np
```

```
[28]
#load saved vgg 16 model file
model=load_model("project_vgg19.h5")
```

```
[29]
# test accuracy
print('Test Score',model.evaluate(test_set))
```

```
8/8 [==============================] - 23s 2s/step - loss: 0.7823 - accuracy: 0.7980
Test Score [0.7822853922843933, 0.7979999780654907]
```

```
[30]
# train accuracy
print('Train Score',model.evaluate(training_set))
```

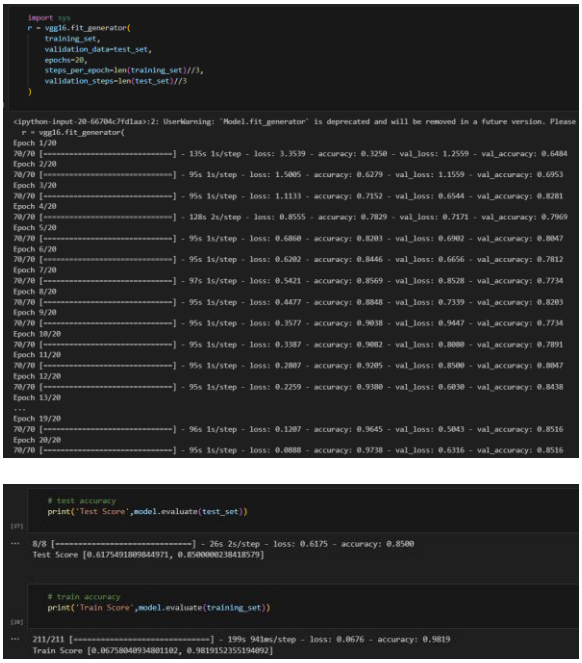
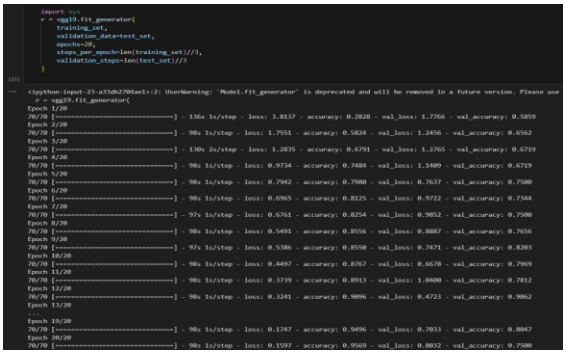
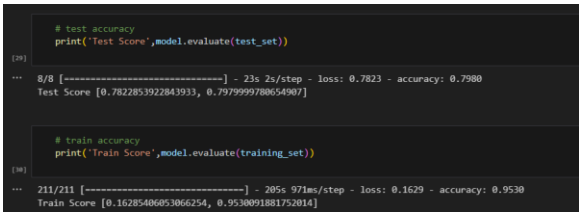
```
211/211 [==============================] - 205s 971ms/step - loss: 0.1629 - accuracy: 0.9530
Train Score [0.16285406053066254, 0.9530091881752014]
```

```
[32]
img=image.load_img("/content/test/sky surfing/4.jpg", target_size=(224,224))
#convert image to array format
x=image.img_to_array(img)
import numpy as np
x=np.expand_dims(x,axis=0)
img_data=preprocess_input(x)
output=np.argmax(model.predict(img_data), axis=1)
index=['air hockey', 'ampute football', 'archery', 'arm wrestling', 'axe throwing',
    'balance beam', 'barell racing', 'baseball', 'basketball', 'baton twirling',
    'bike polo', 'billiards', 'bmx', 'bobsled', 'bowling', 'boxing', 'bull riding',
    'bungee jumping', 'canoe slamon', 'cheerleading', 'chuckwagon racing', 'cricket',
    'croquet', 'curling', 'disc golf', 'fencing', 'field hockey', 'figure skating men',
    'figure skating pairs', 'figure skating women', 'fly fishing', 'football',
    'formula 1 racing', 'frisbee', 'gaga', 'giant slalom', 'golf', 'hammer throw',
    'hang gliding', 'harness racing', 'high jump', 'hockey', 'horse jumping',
    'horse racing', 'horseshoe pitching', 'hurdles', 'hydroplane racing', 'ice climbing',
    'ice yachting', 'jai alai', 'javelin', 'jousting', 'judo', 'lacrosse', 'log rolling',
    'luge', 'motorcycle racing', 'mushing', 'nascar racing', 'olympic wrestling',
    'parallel bar', 'pole climbing', 'pole dancing', 'pole vault', 'polo', 'pommel horse',
    'rings', 'rock climbing', 'roller derby', 'rollerblade racing', 'rowing', 'rugby',
    'sailboat racing', 'shot put', 'shuffleboard', 'sidecar racing', 'ski jumping',
    'sky surfing', 'skydiving', 'snow boarding', 'snowmobile racing', 'speed skating',
    'steer wrestling', 'sumo wrestling', 'surfing', 'swimming', 'table tennis', 'tennis',
    'track bicycle', 'trapeze', 'tug of war', 'ultimate', 'uneven bars', 'volleyball',
    'water cycling', 'water polo', 'weightlifting', 'wheelchair basketball',
    'wheelchair racing', 'wingsuit flying']
result = str(index[output[0]])
result
```

```
1/1 [==============================] - 1s 1s/step
```

```
'sky surfing'
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1 (VGG16)	Model 1 (VGG16) Summary :- ♦ VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford. It has 16 layers with weights, consisting of 13 convolutional layers and 3 fully connected layers. ♦ The model is pre-trained on the ImageNet dataset and is known for its simplicity and effectiveness in image classification tasks.	
Model 2 (VGG19)	Model 2 (VGG19) Summary :- ♦ VGG19 is an extension of VGG16 with 19 layers, including 16 convolutional layers and 3 fully connected layers. ♦ Like VGG16, it is pre-trained on the ImageNet dataset and is used for image classification tasks.	 

…	…	…