Deep Learning Techniques - CS6005

Project 2 - Transfer Learning For Image Classification

Jerrick Gerald
2018103031
N- Batch

Dataset Details:

Data was collected by a group of students at Atılım University from food courts of several shopping malls in Ankara, Turkey. 5 Brand logos were recorded and then extracted still images from these videos. Furthermore, there is a class called "None" for images without any logos. Data is already splitted into two main directories as train and test for the user convenience.

Dataset Name: **logos-bk-kfc-mcdonald-starbucks-subway-none**

Modules:

1. Model Summary
2. Train.Test,Val Generator
3. Training and Evaluating Models
4. Predictions

Model Summary

Inception V3

```
base_model = InceptionV3(weights='imagenet',
                         include_top=False,
                         input_shape=(224, 224,3))
base_model.trainable = False
x = base_model.output
x = keras.layers.GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
predictions = Dense(len(selectedClasses), activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
model.summary()
```

```
__________________________________________________________________________________________________
activation_939 (Activation)     (None, 5, 5, 192)    0           batch_normalization_939[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 5, 5, 2048)   0           activation_931[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_19[0][0]
                                                                 activation_939[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_9 (Glo (None, 2048)         0           mixed10[0][0]
__________________________________________________________________________________________________
dropout_9 (Dropout)             (None, 2048)         0           global_average_pooling2d_9[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 6)            12294       dropout_9[0][0]
==================================================================================================
Total params: 21,815,078
Trainable params: 12,294
Non-trainable params: 21,802,784
__________________________________________________________________________________________________
```

CODING SNAPSHOTS:

```
import numpy as np
import keras
from keras import backend as K
from keras.models import Sequential
from keras.models import Model
from keras.layers import Activation
from keras.layers.core import Dense, Flatten
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.normalization import BatchNormalization
from keras.layers.core import Dropout
from keras.layers.convolutional import *
from keras.callbacks import ModelCheckpoint
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_v3 import preprocess_input
from keras.applications.inception_v3 import decode_predictions
from sklearn.metrics import confusion_matrix
from sklearn.metrics import average_precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from keras.models import model_from_json
import itertools
import matplotlib.pyplot as plt
import time
import pandas as pd
%matplotlib inline
```

```
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(224, 224),
    batch_size=batchSize,
    classes=selectedClasses,
    subset='training') # set as training data

validation_generator = train_datagen.flow_from_directory(
    train_path, # same directory as training data
    target_size=(224, 224),
    batch_size=batchSize,
    classes=selectedClasses,
    subset='validation') # set as validation data

test_generator = ImageDataGenerator().flow_from_directory(
    test_path,
    target_size=(224,224),
    classes=selectedClasses,
    shuffle= False,
    batch_size = batchSize)# set as test data
```

```
Found 1393 images belonging to 6 classes.
Found 345 images belonging to 6 classes.
Found 560 images belonging to 6 classes.
```

```
train_datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        fill_mode='nearest',
    validation_split=0.2)
```

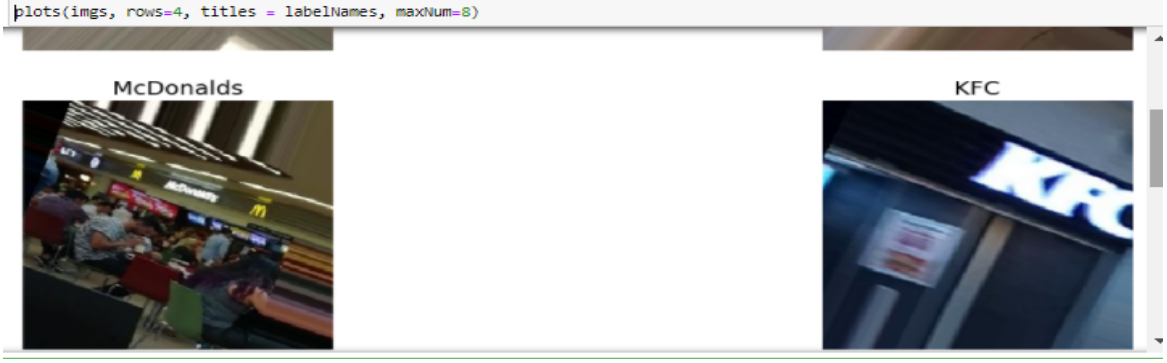
```
print ("In train_generator ")
for cls in range(len (train_generator.class_indices)):
    print(selectedClasses[cls],":\t",list(train_generator.classes).count(cls))
print ("")

print ("In validation_generator ")
for cls in range(len (validation_generator.class_indices)):
    print(selectedClasses[cls],":\t",list(validation_generator.classes).count(cls))
print ("")

print ("In test_generator ")
for cls in range(len (test_generator.class_indices)):
    print(selectedClasses[cls],":\t",list(test_generator.classes).count(cls))
```

```
In train_generator
Burger King :    238
KFC :    56
McDonalds :      152
Other :  660
Starbucks :      187
Subway :         100

In validation_generator
Burger King :    59
KFC :    14
McDonalds :      37
Other :  165
Starbucks :      46
Subway :         24
```

Plot Some Train Data

```
train_generator.reset()
imgs, labels = train_generator.next()
labelNames=[]
labelIndices=[np.where(r==1)[0][0] for r in labels]
for ind in labelIndices:
    for labelName,labelIndex in train_generator.class_indices.items():
        if labelIndex == ind:

            labelNames.append(labelName)
```

```
plots(imgs, rows=4, titles = labelNames, maxNum=8)
```



Early Stopping:

```
modelName= "InceptionTutorial"
filepath=modelName+"_bestweights.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
```

Compile the model

```
model.compile(Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Training the model:

Train

Run more epochs for increasing the accuracy. For example:

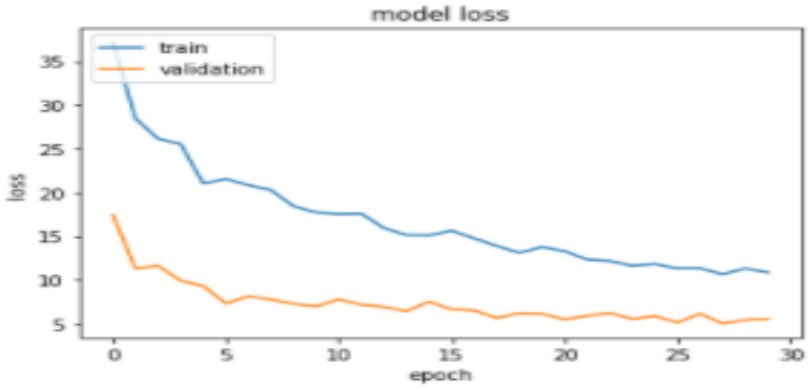
epochs = 30

```
train_generator.reset()
validation_generator.reset()

history = model.fit_generator(
    train_generator,
    validation_data = validation_generator,
    epochs = 30,
    steps_per_epoch = stepsPerEpoch,
    validation_steps= validationSteps,
    callbacks=callbacks_list,
    verbose=1)
```

Epoch:30

```
Epoch 24/30
44/44 [==============================] - 25s 571ms/step - loss: 11.1414 - accuracy: 0.5383 - val_loss: 5.5614 - val_accur
acy: 0.5681
Epoch 25/30
44/44 [==============================] - 25s 564ms/step - loss: 11.2385 - accuracy: 0.5255 - val_loss: 5.8767 - val_accur
acy: 0.5739
Epoch 26/30
44/44 [==============================] - 24s 550ms/step - loss: 11.2597 - accuracy: 0.5224 - val_loss: 5.1521 - val_accur
acy: 0.5507
Epoch 27/30
44/44 [==============================] - 25s 562ms/step - loss: 11.6828 - accuracy: 0.4858 - val_loss: 6.1482 - val_accur
acy: 0.5681
Epoch 28/30
44/44 [==============================] - 25s 567ms/step - loss: 11.3718 - accuracy: 0.5354 - val_loss: 5.0413 - val_accur
acy: 0.6087
Epoch 29/30
44/44 [==============================] - 24s 549ms/step - loss: 11.4750 - accuracy: 0.5124 - val_loss: 5.4308 - val_accur
acy: 0.5739
Epoch 30/30
44/44 [==============================] - 25s 559ms/step - loss: 11.0646 - accuracy: 0.5246 - val_loss: 5.5523 - val_accur
acy: 0.5681
```

Plotting:



Evaluating Models

```
validation_generator.reset()
score = model.evaluate_generator(validation_generator, (validation_generator.samples + (batchSize-1)) //batchSize)
print("For validation data set; Loss: ",score[0]," Accuracy: ", score[1])
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1877: 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 '
```

```
For validation data set; Loss:  5.207617282867432  Accuracy:  0.5826086997985840
```

```
test_generator.reset()
score = model.evaluate_generator(test_generator, (test_generator.samples + (batchSize-1)) // batchSize)
print("For test data set; Loss: ",score[0]," Accuracy: ", score[1])
```

```
For test data set; Loss:  9.788841247558594  Accuracy:  0.5321428775787354
```

Predictions

```
predicted_class_indices=np.argmax(predictions,axis=1)
print(predicted_class_indices)
len(predicted_class_indices)
```

```
[5 0 3 0 0 3 2 4 2 2 2 3 2 0 4 0 0 4 4 4 0 3 0 2 3 0 5 4 0 0 0 0 0 0 0 0 0
 0 0 4 3 0 0 4 4 2 2 2 4 0 4 4 4 4 0 4 4 0 0 2 2 0 2 2 4 0 4 4 4 0 0 0 3 3
 0 4 4 2 3 0 2 4 3 4 0 4 4 0 2 0 2 4 0 4 2 2 2 0 4 0 0 4 0 2 0 0 0 2 2 3 3
 4 4 0 2 4 4 4 0 4 4 4 4 4 0 2 2 4 2 3 2 2 0 4 2 0 5 0 2 4 5 0 4 3 0 2 0 4
 3 2 0 5 5 4 3 3 3 4 4 4 3 4 4 4 4 4 4 3 3 3 4 3 4 0 0 2 3 3 2 3 4 3 3 0 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 0 0 0 0 4 0 0 0 5 0 5 0 0 5 0 0 5 5 4
 0 0 0 0 5 0 0 0 0 5 3 5 4 0 4 0 3 0 0 0 0 5 4 0 0 0 0 0 0 5 5 4 4 0 4 0 4
 0 0 4 4 4 4 3 3 5 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 4 3 3 3 3 3 3 3 3 3 4 4 4 4 4 0 0 0 4 0 4 0 0 0 0 0 0 4 4 4 4 0 2 0 4
 4 4 4 4 0 4 4 4 4 4 4 4 4 4 3 3 4 3 4 4 3 3 3 3 0 4 4 4 4 4 4 3 4 3 4 4 0
 0 4 4 4 3]
```

39]: 560

*predicted_class_indices has the predicted labels, but you can't simply tell what the predictions are, because all you can see is numbers like 0,1,4,1,0,6... and most importantly you need to map the predicted labels with their unique ids such as filenames to find out what you predicted for which image.

```
labels = (test_generator.class_indices)
print(labels)
```

```
{'Burger King': 0, 'KFC': 1, 'McDonalds': 2, 'Other': 3, 'Starbucks': 4, 'Subway': 5}
```

```
labels = dict((v,k) for k,v in labels.items())
print(labels)
```

```
{0: 'Burger King', 1: 'KFC', 2: 'McDonalds', 3: 'Other', 4: 'Starbucks', 5: 'Subway'}
```

```
predictedLables= [labels[k] for k in predicted_class_indices]
print(predictedLables)
```

```
test_generator.reset()
testStep = (test_generator.samples + (batchSize-1)) // batchSize
print("testStep: ", testStep)
predictions = model.predict_generator(test_generator, steps = testStep ,  verbose = 1)
len(predictions)
```

```
testStep:  18
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1905: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
```

```
18/18 [==============================] - 3s 104ms/step
```

]: 560

```
len(predictions)
```

]: 560

Accuracy

```
accuracy_score(actualLables, predictedLables)
```

```
4]: 0.5321428571428571
```

```
matrix = confusion_matrix(actualLables, predictedLables)
print(labels)
matrix
```

```
{0: 'Burger King', 1: 'KFC', 2: 'McDonalds', 3: 'Other', 4: 'Starbucks', 5: 'Subway'}

5]: array([[ 54,   0,  33,  15,  45,   6],
       [  0,   0,   0,   8,  12,   0],
       [ 41,   0,   2,   9,  32,  12],
       [  1,   0,   3, 218,   2,   0],
       [ 13,   0,   1,   0,  24,   0],
       [  3,   0,   0,  10,  16,   0]])
```

Classification Report

```
print(classification_report(actualLables, predictedLables))
```

```
              precision    recall  f1-score   support

  Burger King       0.48      0.35      0.41       153
          KFC       0.00      0.00      0.00        20
    McDonalds       0.05      0.02      0.03        96
        Other       0.84      0.97      0.90       224
    Starbucks       0.18      0.63      0.28        38
       Subway       0.00      0.00      0.00        29

     accuracy                           0.53       560
    macro avg       0.26      0.33      0.27       560
 weighted avg       0.49      0.53      0.50       560
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classifi
re are ill-defined and being set to 0.0 in labels with no predic
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classifi
re are ill-defined and being set to 0.0 in labels with no predic
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classifi
re are ill-defined and being set to 0.0 in labels with no predic
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
recall_score( actualLables, predictedLables,average='weighted')
```

```
0.5321428571428571
```

```
precision_score( actualLables, predictedLables,average='weighted
```

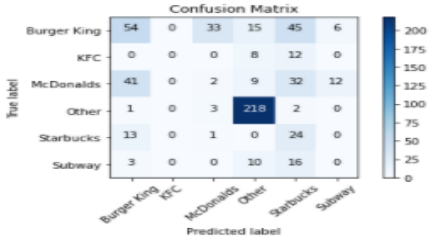
```
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classifi
fined and being set to 0.0 in labels with no predicted samples.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
0.48833598346854895
```

Confusion Matrix & Sample Prediction

```
cm_plot_labels = selectedClasses
plot_confusion_matrix(matrix,cm_plot_labels, normalize=False
                      , title = 'Confusion Matrix')
```

```
Confusion matrix, without normalization
[[ 54   0  33  15  45   6]
 [  0   0   0   8  12   0]
 [ 41   0   2   9  32  12]
 [  1   0   3 218   2   0]
 [ 13   0   1   0  24   0]
 [  3   0   0  10  16   0]]
```




```
import matplotlib.image as mpimg
%matplotlib inline

res = results[260:280]

images = []

for img_path in "./"+res['Directory']+"/"+res['Filename']:
    images.append(mpimg.imread(img_path))

plt.figure(figsize=(80,80))
columns = 4
for i, image in enumerate(images):
    ax= plt.subplot(len(images) / columns + 1, columns, i + 1)
    ax.set_title(res['Actuals'].iloc[i]+" "+res['Predictions'].iloc[i], fontsize=
    plt.imshow(image)
```