Model1_FT - Fine Tuning using InceptionNetV2

by Kaushik Srivatsan - CDS - kaushik.s-25@scds.saiuniversity.edu.in

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import tensorflow as tf
print(tf.__version__)

from tensorflow import keras
tf.random.set_seed(42)

import numpy as np
np.random.seed(42)

import matplotlib.pyplot as plt
%matplotlib inline

import glob
import PIL
from PIL import Image
```

```
2.15.0
```

Loading the preprocessed dataset

```
# load numpy array from npy file
from numpy import load

X_train_std = load('/content/drive/MyDrive/Models/X_train_std_model1.npy')
X_test_std = load('/content/drive/MyDrive/Models/X_test_std_model1.npy')

y_train = load('/content/drive/MyDrive/Models/y_train_model1.npy')
y_test = load('/content/drive/MyDrive/Models/y_test_model1.npy')
```

```
print("X_train_std_shape: {}".format(X_train_std.shape))
print("X_test_std_shape: {}".format(X_test_std.shape))
```

```
X_train_std_shape: (373, 299, 299, 3)
X_test_std_shape: (125, 299, 299, 3)
```

Loading the Transfer-learning Model

```
model1_FT = keras.models.load_model('/content/drive/MyDrive/Models/Model1_TL.h5')
model1_FT.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 299, 299, 3)]     0         []

 conv2d (Conv2D)             (None, 149, 149, 32)      864       ['input_1[0][0]']

 batch_normalization (Batch  (None, 149, 149, 32)      96        ['conv2d[0][0]']
 Normalization)

 activation (Activation)     (None, 149, 149, 32)      0         ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)           (None, 147, 147, 32)      9216      ['activation[0][0]']

 batch_normalization_1 (Bat  (None, 147, 147, 32)      96        ['conv2d_1[0][0]']
 chNormalization)

 activation_1 (Activation)   (None, 147, 147, 32)      0         ['batch_normalization_1[0][0]'
                                                                 ]

 conv2d_2 (Conv2D)           (None, 147, 147, 64)      18432     ['activation_1[0][0]']

 batch_normalization_2 (Bat  (None, 147, 147, 64)      192       ['conv2d_2[0][0]']
 chNormalization)
```

```
activation_2 (Activation)   (None, 147, 147, 64)         0         ['batch_normalization_2[0][0]'
                                                                    ]

max_pooling2d (MaxPooling2  (None, 73, 73, 64)          0         ['activation_2[0][0]']
D)

conv2d_3 (Conv2D)           (None, 73, 73, 80)          5120      ['max_pooling2d[0][0]']

batch_normalization_3 (Bat  (None, 73, 73, 80)          240       ['conv2d_3[0][0]']
chNormalization)

activation_3 (Activation)   (None, 73, 73, 80)          0         ['batch_normalization_3[0][0]'
                                                                    ]

conv2d_4 (Conv2D)           (None, 71, 71, 192)         138240    ['activation_3[0][0]']

batch_normalization_4 (Bat  (None, 71, 71, 192)         576       ['conv2d_4[0][0]']
chNormalization)

activation_4 (Activation)   (None, 71, 71, 192)         0         ['batch_normalization_4[0][0]'
                                                                    ]

max_pooling2d_1 (MaxPoolin  (None, 35, 35, 192)         0         ['activation_4[0][0]']
g2D)

conv2d_8 (Conv2D)           (None, 35, 35, 64)          12288     ['max_pooling2d_1[0][0]']

batch_normalization_8 (Bat  (None, 35, 35, 64)          192       ['conv2d_8[0][0]']
chNormalization)

activation_8 (Activation)   (None, 35, 35, 64)          0         ['batch_normalization_8[0][0]'
                                                                    ]
```

Modifyng and Fine tuning the layers to be trained

```
total_layers = len(model1_FT.layers)
split_index = int(0.25 * total_layers)

for layer in model1_FT.layers[:split_index]:
    layer.trainable = False

for layer in model1_FT.layers[split_index:]:
    layer.trainable = True
```

Compiling and Training the Model

```
model1_FT.compile(loss='sparse_categorical_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

callbacks_FineTune = [
            keras.callbacks.ModelCheckpoint("bestFT1.h5",
                                            monitor='val_accuracy',
                                            save_weights_only=True,
                                            save_best_only=True)
]

history_FineTune = model1_FT.fit(x = X_train_std, y = y_train, epochs=10,
                                 validation_split=0.1, batch_size=16, callbacks=callbacks_FineTune)
```

```
Epoch 1/10
21/21 [==============================] - 115s 1s/step - loss: 2.0020 - accuracy: 0.5403 - val_loss: 91.9486 - val_accuracy: 0.2105
Epoch 2/10
21/21 [==============================] - 11s 515ms/step - loss: 0.8413 - accuracy: 0.7134 - val_loss: 448.7038 - val_accuracy: 0.289
Epoch 3/10
21/21 [==============================] - 10s 459ms/step - loss: 0.6594 - accuracy: 0.7731 - val_loss: 58.6408 - val_accuracy: 0.210
Epoch 4/10
21/21 [==============================] - 11s 515ms/step - loss: 0.5184 - accuracy: 0.8328 - val_loss: 94.6049 - val_accuracy: 0.342
Epoch 5/10
21/21 [==============================] - 9s 449ms/step - loss: 0.5477 - accuracy: 0.8269 - val_loss: 10469.3779 - val_accuracy: 0.28
Epoch 6/10
21/21 [==============================] - 9s 450ms/step - loss: 0.3422 - accuracy: 0.8746 - val_loss: 2586.4138 - val_accuracy: 0.315
Epoch 7/10
21/21 [==============================] - 11s 517ms/step - loss: 0.2396 - accuracy: 0.8985 - val_loss: 3.8631 - val_accuracy: 0.3947
Epoch 8/10
21/21 [==============================] - 11s 537ms/step - loss: 0.2028 - accuracy: 0.9254 - val_loss: 2.0081 - val_accuracy: 0.4211
Epoch 9/10
21/21 [==============================] - 11s 517ms/step - loss: 0.1190 - accuracy: 0.9522 - val_loss: 4.1281 - val_accuracy: 0.5263
Epoch 10/10
21/21 [==============================] - 10s 459ms/step - loss: 0.1709 - accuracy: 0.9254 - val_loss: 16.6901 - val_accuracy: 0.447
```

```
keys = ['accuracy', 'val_accuracy']
progress = {k:v for k,v in history_FineTune.history.items() if k in keys}

import pandas as pd
pd.DataFrame(progress).plot()

plt.xlabel("epochs")
plt.ylabel("accuracy")

plt.grid(True)
plt.show()
```

Evaluating the Model with Best weights

```
model1_FT.load_weights("bestFT1.h5")

testLoss_FineTune, testAccuracy_FineTune = model1_FT.evaluate(x = X_test_std, y = y_test)

print("Test-loss: %f, Test-accuracy: %f" % (testLoss_FineTune, testAccuracy_FineTune))
```

```
4/4 [==============================] - 12s 2s/step - loss: 1.7353 - accuracy: 0.6320
Test-loss: 1.735336, Test-accuracy: 0.632000
```

Checking Model Performance

```
y_proba = model1_FT.predict(X_test_std)
y_predict = np.argmax(y_proba, axis=-1)
print(y_predict)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true = y_test, y_pred = y_predict)

fig, ax = plt.subplots(figsize=(6, 6))
ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)

for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i, s=cm[i, j], va='center', ha='center')

ax.title.set_text('CNN\n')
plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.tight_layout()
plt.savefig("ConfusionMatrix.png", dpi=300, format='png', pad_inches=0.3)
plt.show()
```

```
from sklearn.metrics import precision_score, recall_score, f1_score

pScore = precision_score(y_true= y_test, y_pred = y_predict, average = 'weighted')
print("Precision: ", pScore)

rScore = recall_score(y_true= y_test, y_pred = y_predict, average = 'weighted')
print("Recall: ", rScore)

fScore = f1_score(y_true= y_test, y_pred = y_predict, average = 'weighted')
print("F1-score: ", fScore)

print("\n\n\n")
```

Saving the Fine-Tuned Model

```
model1_FT.save('/content/drive/MyDrive/Models/model1_FT.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file vi
  saving_api.save_model(
```