Mounting Drive

```
from google.colab import drive

drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Second part of the NB

https://colab.research.google.com/drive/1riHzh22WI-Z0GJ2cEbkPcB9CND4_ksk9#scrollTo=Mz_8IHW-jbUA

Notebook Imports

```
#-------------------------------------------------------------------------
# VGG16 ON CIFAR_10
#-------------------------------------------------------------------------
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.applications.vgg16 import VGG16
import keras as k
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense, Flatten, Dropout
from keras.utils.np_utils import to_categorical
from keras import optimizers
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split


from sklearn import metrics


from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from itertools import cycle
from sklearn import svm, datasets
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier


from matplotlib import pyplot
import matplotlib.pyplot as plt
from keras.utils.vis_utils import plot_model
```

Loading the Data

```
batch1 = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/batch1.npy')
batch2 = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/batch2.npy')
batch3 = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/batch3.npy')
batch4 = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/batch4.npy')
batch5 = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/batch5.npy')
test_batch = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/converted numpy files/test_batch.npy')


batch1_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/batch1_labels.npy')
batch2_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/batch2_labels.npy')
batch3_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/batch3_labels.npy')
batch4_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/batch4_labels.npy')
batch5_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/batch5_labels.npy')


test_batch_labels = np.load('/content/gdrive/MyDrive/DL ASSIGNMENT/cifar 10/og/numpy files/test_batch_labels.npy')


test_batch_labels_ohe = to_categorical(test_batch_labels, num_classes = 10)
```

Merging the Original and Transformed Data

```
batch1_labels_ohe = to_categorical(batch1_labels, num_classes = 10)
batch2_labels_ohe = to_categorical(batch2_labels, num_classes = 10)
batch3_labels_ohe = to_categorical(batch3_labels, num_classes = 10)
batch4_labels_ohe = to_categorical(batch4_labels, num_classes = 10)
batch5_labels_ohe = to_categorical(batch5_labels, num_classes = 10)
```

Preparing the data

Batch 1

```
X_train1, X_val1, y_train1, y_val1 = train_test_split(batch1, batch1_labels, test_size=0.2, random_state=42)

# Convert class vectors to binary class matrices using one hot encoding
y_train1_ohe = to_categorical(y_train1, num_classes = 10)
y_val1_ohe = to_categorical(y_val1, num_classes = 10)

# Data normalization
X_train1 = X_train1.astype('float32')
X_val1 = X_val1.astype('float32')
X_train1 /= 255
X_val1 /= 255


test_batch_labels_ohe = to_categorical(test_batch_labels, num_classes=10)
```

Debug Batches

```
batch1_n = batch1.astype('float32')
batch1_n /= 255


batch2_n = batch2.astype('float32')
batch2_n /= 255


batch3_n = batch3.astype('float32')
batch3_n /= 255


batch4_n = batch4.astype('float32')
batch4_n /= 255


batch5_n = batch5.astype('float32')
batch5_n /= 255


test_batch_labels_1 = test_batch_labels_ohe[:5000]
test_batch_labels_2 = test_batch_labels_ohe[5000:]

test_batch_f = test_batch.astype('float32')
test_batch_f /= 255

test_batch_1 = test_batch_f[:5000]
test_batch_2 = test_batch_f[5000:]
```

Batch 2

```
X_train2, X_val2, y_train2, y_val2 = train_test_split(batch2, batch2_labels, test_size=0.2, random_state=42)

# Convert class vectors to binary class matrices using one hot encoding
y_train2_ohe = to_categorical(y_train2, num_classes = 10)
y_val2_ohe = to_categorical(y_val2, num_classes = 10)

# Data normalization
X_train2 = X_train2.astype('float32')
X_val2 = X_val2.astype('float32')
X_train2 /= 255
X_val2 /= 255
```

Batch 3

```
X_train3, X_val3, y_train3, y_val3 = train_test_split(batch3, batch3_labels, test_size=0.2, random_state=42)

# Convert class vectors to binary class matrices using one hot encoding
y_train3_ohe = to_categorical(y_train3, num_classes = 10)
y_val3_ohe = to_categorical(y_val3, num_classes = 10)

# Data normalization
X_train3 = X_train3.astype('float32')
X_val3 = X_val3.astype('float32')
X_train3 /= 255
X_val3 /= 255
```

Batch 4

```
X_train4, X_val4, y_train4, y_val4 = train_test_split(batch4, batch4_labels, test_size=0.2, random_state=42)

# Convert class vectors to binary class matrices using one hot encoding
y_train4_ohe = to_categorical(y_train4, num_classes = 10)
y_val4_ohe = to_categorical(y_val4, num_classes = 10)

# Data normalization
X_train4 = X_train4.astype('float32')
X_val4 = X_val4.astype('float32')
X_train4 /= 255
X_val4 /= 255
```

Batch 5

```
X_train5, X_val5, y_train5, y_val5 = train_test_split(batch5, batch5_labels, test_size=0.2, random_state=42)

# Convert class vectors to binary class matrices using one hot encoding
y_train5_ohe = to_categorical(y_train5, num_classes = 10)
y_val5_ohe = to_categorical(y_val5, num_classes = 10)

# Data normalization
X_train5 = X_train5.astype('float32')
X_val5 = X_val5.astype('float32')
X_train5 /= 255
X_val5 /= 255


vgg16_model = VGG16(weights='imagenet',
                    include_top=False,
                    classes=10,
                    input_shape=(32,32,3)# input: 32x32 images with 3 channels -> (32, 32, 3) tensors.
                   )

print(vgg16_model.layers)
```

```
[<keras.engine.input_layer.InputLayer object at 0x7f7bff5c4b90>, <keras.layers.convolutional.Conv2D object at 0x7f7bff5a
```

Creating the Model

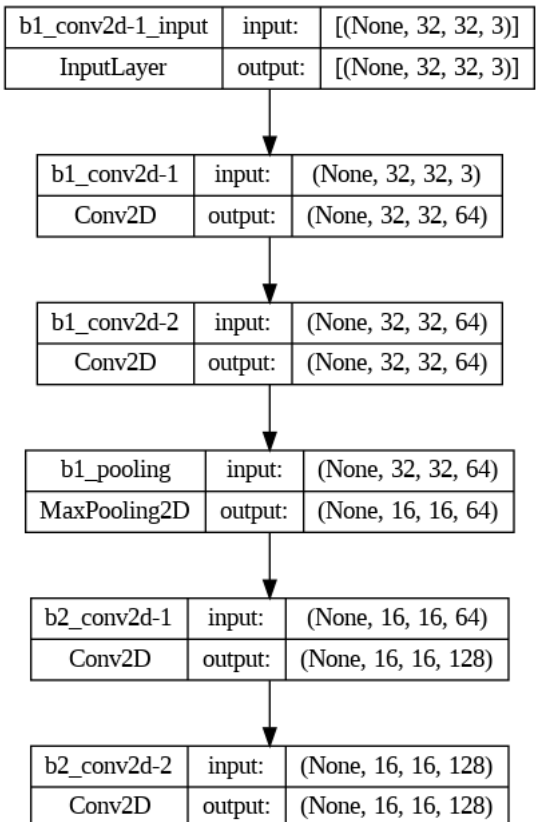
```
model = tf.keras.Sequential()

#BLOCK-1
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3), nam
```

```
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b1_conv2d-2"))
model.add(MaxPooling2D((2, 2), name="b1_pooling"))

#BLOCK-2
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b2_conv2d-1"))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b2_conv2d-2"))
model.add(MaxPooling2D((2, 2), name="b2_pooling"))

#BLOCK-3
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b3_conv2d-1"))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b3_conv2d-2"))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b3_conv2d-3"))
model.add(MaxPooling2D((2, 2), name="b3_pooling"))

# #BLOCK-4
# model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b4_conv2d-1"))
# model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b4_conv2d-2"))
# model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b4_conv2d-3"))
# model.add(MaxPooling2D((2, 2), name="b4_pooling"))

#BLOCK-5
model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b5_conv2d-1"))
model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b5_conv2d-2"))
model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', name="b5_conv2d-3"))
model.add(MaxPooling2D((2, 2), name="b4_pooling"))


#--------------------------------------------------------------------------------
# Adding hiddens  and output layer to our model
#--------------------------------------------------------------------------------

from keras.layers import Dense, Flatten, Dropout
model.add(Flatten())
model.add(Dense(512, activation='relu', name='hidden1'))
# model.add(Dropout(0.4))
model.add(Dense(256, activation='relu', name='hidden2'))
# model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax', name='predictions'))



model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 b1_conv2d-1 (Conv2D)        (None, 32, 32, 64)        1792

 b1_conv2d-2 (Conv2D)        (None, 32, 32, 64)        36928

 b1_pooling (MaxPooling2D)   (None, 16, 16, 64)        0

 b2_conv2d-1 (Conv2D)        (None, 16, 16, 128)       73856

 b2_conv2d-2 (Conv2D)        (None, 16, 16, 128)       147584

 b2_pooling (MaxPooling2D)   (None, 8, 8, 128)         0

 b3_conv2d-1 (Conv2D)        (None, 8, 8, 256)         295168

 b3_conv2d-2 (Conv2D)        (None, 8, 8, 256)         590080

 b3_conv2d-3 (Conv2D)        (None, 8, 8, 256)         590080

 b3_pooling (MaxPooling2D)   (None, 4, 4, 256)         0

 b5_conv2d-1 (Conv2D)        (None, 4, 4, 512)         1180160

 b5_conv2d-2 (Conv2D)        (None, 4, 4, 512)         2359808

 b5_conv2d-3 (Conv2D)        (None, 4, 4, 512)         2359808

 b4_pooling (MaxPooling2D)   (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 hidden1 (Dense)             (None, 512)               1049088

 hidden2 (Dense)             (None, 256)               131328

 predictions (Dense)         (None, 10)                2570

=================================================================
Total params: 8,818,250
Trainable params: 8,818,250
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
plot_model(model, to_file = 'model_plot.png', show_shapes = True, show_layer_names=True)
```




```
sess = tf.compat.v1.Session()
```

```
# initiate sgd optimizer
sgd = tf.keras.optimizers.SGD(learning_rate=0.001)

def roc_auc_score_ovo(y_true, y_pred):
  y_true, y_pred = y_true.eval(session=sess), y_pred.eval(session=sess)
  return roc_auc_score(y_true, y_pred, multi_class = 'ovo')

# For a multi-class classification problem
model.compile(loss='categorical_crossentropy', optimizer = sgd, metrics=['accuracy'])
```

```
# initialize the number of epochs and batch size
EPOCHS = 60
BS = 64
```

Training

[ ] ↳ *18 cells hidden*