ML_Caltech256_single_VGG16-Data_Agumentation

November 24, 2019

1 Final project Caltech256

-- Network used VGG16

- **Pretrained VGG model (using weights of imageNet)**

Student Name/ID : Aloukik aditya(115290) , Sarthak Rawat(1101124)

[1]:
```
from __future__ import absolute_import, division, print_function,␣
 ↪unicode_literals

import tensorflow as tf
import glob
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense,␣
 ↪Dropout,concatenate,Activation
import numpy as np
from PIL import Image
from os import listdir
import sys
import matplotlib.pyplot as plt
from tensorflow.keras import optimizers
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import CSVLogger
from sklearn.preprocessing import OneHotEncoder
import timeit
import gc
import random
from tensorflow.python.keras.callbacks import TensorBoard
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.regularizers import l1
from tensorflow.keras.callbacks import LearningRateScheduler
from time import time

print("Setup Done..")
```

```
Setup Done..
```

1.1 Loading test and train data

- we have split the data into 30 images of each category for Training
- And the rest for testing

[2]:
```
path = './256_ObjectCategories/'
random.seed(28)
img_size = 224
test_dict = {}
train_dict = {}

train_imgs = []
test_imgs = []
train_labels = []
test_labels = []

ctr=0

start = timeit.default_timer()
categories = listdir(path)
for category in categories:
    image_files_list = glob.glob(path+category+ '/*.jpg')
    train_list = random.sample(image_files_list, k=30)
    test_list = [x for x in image_files_list if x not in train_list]
    test_dict[category] = test_list
    train_dict[category] = train_list


for key in test_dict:
    for img_name in test_dict[key]:
        file =  img_name
        test_imgs.append(np.array(image.load_img(file, target_size=(img_size,␣
 ↪img_size))))
        test_labels.append(key)


for key in train_dict:
    for img_name in train_dict[key]:
        file =  img_name
        train_imgs.append(np.array(image.load_img(file, target_size=(img_size,␣
 ↪img_size))))
        train_labels.append(key)


stop = timeit.default_timer()
print("Loading dataset Done")
print('Time taken to load data : ', stop - start)
test_imgs = np.array(test_imgs)
```

```
train_imgs = np.array(train_imgs)
test_labels = np.array(test_labels)
train_labels = np.array(train_labels)
```

```
Loading dataset Done
Time taken to load data :  270.3415339139992
```

1.2 Checking data shape and size

- We can see Train shape is of 7710 images (257*30)
- And rest for testing

[3]:
```
gc.collect()
print("Shape of train data: ",train_imgs.shape)
print("Shape of test data: ",test_imgs.shape)
print("Shape of train labels: ",train_labels.shape)
print("Shape of test labels: ",test_labels.shape)
print("Total instances: ",train_labels.shape[0]+test_labels.shape[0])
```

```
Shape of train data:  (7710, 224, 224, 3)
Shape of test data:  (22897, 224, 224, 3)
Shape of train labels:  (7710,)
Shape of test labels:  (22897,)
Total instances:  30607
```

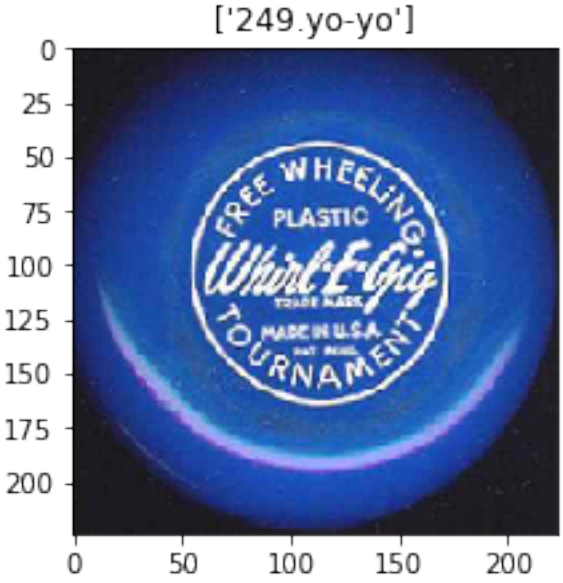
1.3 Flipping images for image augmentation

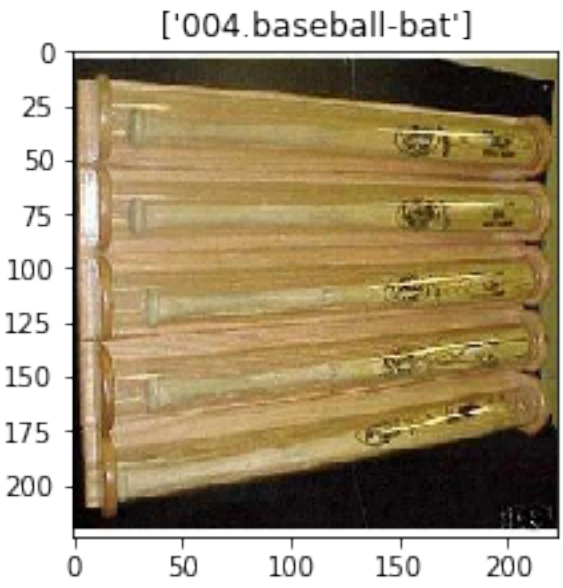
[4]:
```
train_imgs_flip_horizontal = np.flip(train_imgs,axis=2)

train_imgs = np.concatenate((train_imgs,train_imgs_flip_horizontal),axis=0)
train_labels = np.concatenate((train_labels,train_labels),axis=0)
```

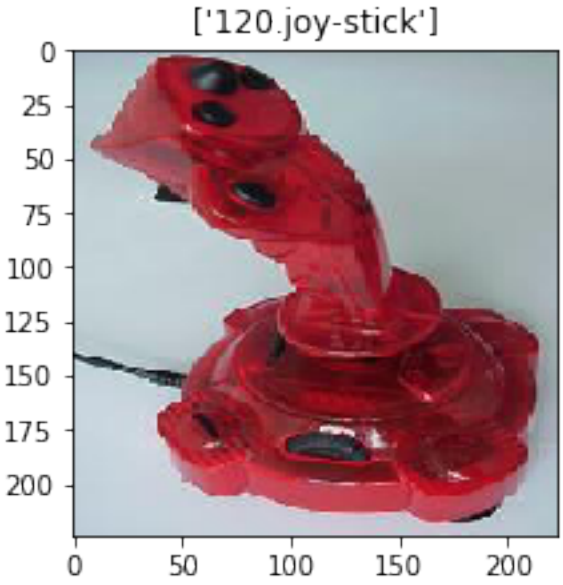
1.4 Plotting some random images

[6]:
```
val_index = []
random.seed(22)
for i in range(4):
    value = random.randint(0,3582)
    val_index.append(value)
    plt.figure( figsize=(15, 15))
    plt.subplot(1, 4, i+1)
    plt.title([train_labels[value]])
    plt.imshow(train_imgs[value])

# plt.tight_layout()
plt.show()
```


['055.dice']
0
25
50
75
100
125
150
175
200
0
50
100
150
200


['249.yo-yo']
0
25
50
75
100
125
150
175
200
0
50
100
150
200
FREE WHEELING
PLASTIC
Whirl-E-Gig
MADE IN U.S.A.
TOURNAMENT





2 Preprocess image using VGG function image

[7]:
```
from tensorflow.keras.applications.vgg16 import preprocess_input
start = timeit.default_timer()
train_imgs = preprocess_input(train_imgs)
test_imgs = preprocess_input(test_imgs)
```

```
stop = timeit.default_timer()
print('Time taken to preprocess/normalize data : ', stop - start)
gc.collect()
```

```
Time taken to preprocess/normalize data :  22.64659117500014
```

[7]:
```
9125
```

[8]:
```
#check shape of  image
#number of train images are now doubled due to image augmentation(we have only␣
 ↪used flip image for augmentation)
print(train_imgs.shape)
test_imgs.shape
```

```
(15420, 224, 224, 3)
```

[8]:
```
(22897, 224, 224, 3)
```

2.1 Applying one hot encoding on our labels

[9]:
```
onehot_encoder = OneHotEncoder(sparse=False)

train_labels = train_labels.reshape(-1,1)
test_labels = test_labels.reshape(-1,1)

train_labels = onehot_encoder.fit_transform(train_labels)
test_labels = onehot_encoder.transform(test_labels)
```

[9]:
```
# test_labels1 = onehot_encoder.inverse_transform(test_labels)
```

[10]:
```
print("Shape of train labels: ",train_labels.shape)
print("Shape of test labels: ",test_labels.shape)
```

```
Shape of train labels:  (15420, 257)
Shape of test labels:  (22897, 257)
```

[11]:
```
# Function to free up keras memory

from keras.backend.tensorflow_backend import set_session
from keras.backend.tensorflow_backend import clear_session
from keras.backend.tensorflow_backend import get_session
import tensorflow

# Reset Keras Session
def reset_keras():
    sess = get_session()
    clear_session()
```

```
    sess.close()
    sess = get_session()

    try:
        del classifier # this is from global space - change this as you need
    except:
        pass

    print(gc.collect()) # if it's done something you should see a number being␣
↪outputted

    # use the same config as you used to create the session
    config = tensorflow.ConfigProto()
    config.gpu_options.per_process_gpu_memory_fraction = 1
    config.gpu_options.visible_device_list = "0"
    set_session(tensorflow.Session(config=config))
```

```
Using TensorFlow backend.
```

2.2 Using model from keras--> VGG16

- we have used pretrained weights for imagenet and removed the top fully connected layers.

[22]:
```
#define variable learning rates

def learning_rate_schedule(epoch):
    if epoch <= 6:
        return 1e-5 # 0.00001
    elif epoch <= 8:
        return 1e-5
    elif epoch <= 10:
        return 1e-6
    else:
        return 1e-7
    return LR
```

[14]:
```
#Initilize vgg net
vgg_net = VGG16(include_top=True, weights='imagenet', input_shape=(224,224,3))

#Define our model
x = vgg_net.get_layer('fc2').output
x = Dense(900, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(257, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)
```

```
## save our model weights
model_updated.save_weights('CALTECH256_VGG16_model_updated_initial.h5')
print("Model_Updated saved")

history_fc_list = []
history_full_list = []
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-
packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling
BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model_Updated saved
```

2.3 Run 1

[23]:
```
i=0
#Initilize vgg net
vgg_net = VGG16(include_top=True, weights='imagenet', input_shape=(224,224,3))

#Define our model
x = vgg_net.get_layer('fc2').output
x = Dense(900, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(257, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH256_VGG16_model_updated_initial.h5')

print('-----------------------New run started--------------------------------')
print('Initial model weights loaded , Started training run number ', i+1)
print('----------------------------------------------------------------------')

# Make the last Fully connected layers trainable and freezing the rest of VGG␣
 ↪model
for layer in model_updated.layers:
    layer.trainable = False
for layer in model_updated.layers[-4:]:
    layer.trainable = True
```

```
tensorboard = TensorBoard(log_dir="logs\{}".format('CALTECH256\FC_Layer'+str(i)))

model_updated.compile(loss='categorical_crossentropy', optimizer=optimizers.
 ↪Adam(lr=0.0001), metrics=['accuracy'])


print('-------------------------Starting Fully connected layer␣
 ↪training-----------------------------------')

history_fc = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=2, validation_data=\
                    (test_imgs, test_labels),callbacks =[tensorboard])


gc.collect()
##Unfreeze weights ------ Now full model will be trained
for layer in model_updated.layers:
    layer.trainable = True


no_epochs = 10
tensorboard = TensorBoard(log_dir="logs\{}".
 ↪format('CALTECH256\FULL_Model'+str(i)))

opt2 = optimizers.Adam(lr=0.00001)

### Variable learning rate -----------------------
lrate = LearningRateScheduler(learning_rate_schedule)
callbacks_list = [lrate,tensorboard]




model_updated.compile(loss='categorical_crossentropy', optimizer=opt2,␣
 ↪metrics=['accuracy'])


print('-------------------------Starting Full model training -->after␣
 ↪unfreez-------------------------------------')

## Training full model
history = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=no_epochs, validation_data=\
                    (test_imgs, test_labels),callbacks =callbacks_list)
```

```
## Timer for checking time taken
stop = timeit.default_timer()
print('Time taken for one run is : ', stop - start)

model_updated.save_weights('CALTECH256_VGG16_model_updated_fin.h5')
print("Model_Updated weights saved")

## saving history for plots
history_fc_list.append(history_fc)
history_full_list.append(history)
```

```
------------------------New run started----------------------------------
Initial model weights loaded , Started training run number  1
-------------------------------------------------------------------------
-------------------------Starting Fully connected layer
training-----------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 5.4572 -
acc: 0.2312 - val_loss: 3.2471 - val_acc: 0.6002
Epoch 2/2
15420/15420 [==============================] - 99s 6ms/sample - loss: 2.8660 -
acc: 0.6231 - val_loss: 2.6142 - val_acc: 0.6837
-------------------------Starting Full model training -->after
unfreez-------------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 2.0365 -
acc: 0.7683 - val_loss: 2.1525 - val_acc: 0.7335
Epoch 2/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 1.4026 -
acc: 0.9133 - val_loss: 2.0244 - val_acc: 0.7546
Epoch 3/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 1.1111 -
acc: 0.9714 - val_loss: 1.9311 - val_acc: 0.7646
Epoch 4/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.9460 -
acc: 0.9898 - val_loss: 1.8819 - val_acc: 0.7702
Epoch 5/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.8458 -
acc: 0.9969 - val_loss: 1.8424 - val_acc: 0.7764
Epoch 6/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.7688 -
acc: 0.9988 - val_loss: 1.8147 - val_acc: 0.7805
Epoch 7/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.7125 -
acc: 0.9998 - val_loss: 1.7847 - val_acc: 0.7833
```

```
Epoch 8/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.6666 -
acc: 0.9999 - val_loss: 1.7817 - val_acc: 0.7867
Epoch 9/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.6288 -
acc: 1.0000 - val_loss: 1.7669 - val_acc: 0.7880
Epoch 10/10
15420/15420 [==============================] - 186s 12ms/sample - loss: 0.6020 -
acc: 1.0000 - val_loss: 1.7753 - val_acc: 0.7891
Time taken for one run is :  2970.6186460989975
Model_Updated weights saved
```

2.4 Run 2

[24]:
```
#clearing keras memory to free gpu
reset_keras()
i=1
#Initilize vgg net
vgg_net = VGG16(include_top=True, weights='imagenet', input_shape=(224,224,3))

#Define our model
x = vgg_net.get_layer('fc2').output
x = Dense(900, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(257, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH256_VGG16_model_updated_initial.h5')

print('-----------------------New run started--------------------------------')
print('Initial model weights loaded , Started training run number ', i+1)
print('----------------------------------------------------------------------')

# Make the last Fully connected layers trainable and freezing the rest of VGG␣
 ↪model
for layer in model_updated.layers:
    layer.trainable = False
for layer in model_updated.layers[-4:]:
    layer.trainable = True

tensorboard = TensorBoard(log_dir="logs\{}".format('CALTECH256\FC_Layer'+str(i)))
```

```
model_updated.compile(loss='categorical_crossentropy', optimizer=optimizers.
 ↪Adam(lr=0.0001), metrics=['accuracy'])


print('-------------------------Starting Fully connected layer␣
 ↪training-------------------------------------')

history_fc = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=2, validation_data=\
                    (test_imgs, test_labels),callbacks =[tensorboard])


gc.collect()
##Unfreeze weights ------ Now full model will be trained
for layer in model_updated.layers:
    layer.trainable = True


no_epochs = 10
tensorboard = TensorBoard(log_dir="logs\{}".
 ↪format('CALTECH256\FULL_Model'+str(i)))

opt2 = optimizers.Adam(lr=0.00001)

### Variable learning rate ----------------------
lrate = LearningRateScheduler(learning_rate_schedule)
callbacks_list = [lrate,tensorboard]




model_updated.compile(loss='categorical_crossentropy', optimizer=opt2,␣
 ↪metrics=['accuracy'])


print('-------------------------Starting Full model training -->after␣
 ↪unfreez-------------------------------------')

## Training full model
history = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=no_epochs, validation_data=\
                    (test_imgs, test_labels),callbacks =callbacks_list)



## Timer for checking time taken
stop = timeit.default_timer()
```

```
print('Time taken for one run is : ', stop - start)

model_updated.save_weights('CALTECH256_VGG16_model_updated_fin.h5')
print("Model_Updated weights saved")

## saving history for plots
history_fc_list.append(history_fc)
history_full_list.append(history)
```

```
14
------------------------New run started---------------------------------
Initial model weights loaded , Started training run number  2
------------------------------------------------------------------------
--------------------------Starting Fully connected layer
training-----------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 5.4489 -
acc: 0.2297 - val_loss: 3.2304 - val_acc: 0.6023
Epoch 2/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 2.8594 -
acc: 0.6245 - val_loss: 2.5989 - val_acc: 0.6893
--------------------------Starting Full model training -->after
unfreez-------------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 2.0344 -
acc: 0.7654 - val_loss: 2.1425 - val_acc: 0.7392
Epoch 2/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 1.4046 -
acc: 0.9134 - val_loss: 2.0208 - val_acc: 0.7568
Epoch 3/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 1.1060 -
acc: 0.9723 - val_loss: 1.9331 - val_acc: 0.7677
Epoch 4/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.9481 -
acc: 0.9907 - val_loss: 1.8773 - val_acc: 0.7720
Epoch 5/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.8423 -
acc: 0.9972 - val_loss: 1.8450 - val_acc: 0.7745
Epoch 6/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.7689 -
acc: 0.9988 - val_loss: 1.8139 - val_acc: 0.7802
Epoch 7/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.7136 -
acc: 0.9994 - val_loss: 1.8032 - val_acc: 0.7823
Epoch 8/10
```

```
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.6684 -
acc: 0.9997 - val_loss: 1.7799 - val_acc: 0.7856
Epoch 9/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.6301 -
acc: 0.9999 - val_loss: 1.7696 - val_acc: 0.7883
Epoch 10/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.6010 -
acc: 1.0000 - val_loss: 1.7751 - val_acc: 0.7875
Time taken for one run is :  4983.058034358999
Model_Updated weights saved
```

2.5 Run 3

[25]:
```
#clearing keras memory to free gpu
reset_keras()
i=2
#Initilize vgg net
vgg_net = VGG16(include_top=True, weights='imagenet', input_shape=(224,224,3))

#Define our model
x = vgg_net.get_layer('fc2').output
x = Dense(900, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(257, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH256_VGG16_model_updated_initial.h5')

print('-----------------------New run started--------------------------------')
print('Initial model weights loaded , Started training run number ', i+1)
print('----------------------------------------------------------------------')

# Make the last Fully connected layers trainable and freezing the rest of VGG␣
 ↪model
for layer in model_updated.layers:
    layer.trainable = False
for layer in model_updated.layers[-4:]:
    layer.trainable = True

tensorboard = TensorBoard(log_dir="logs\{}".format('CALTECH256\FC_Layer'+str(i)))

model_updated.compile(loss='categorical_crossentropy', optimizer=optimizers.
 ↪Adam(lr=0.0001), metrics=['accuracy'])
```

```
print('-------------------------Starting Fully connected layer␣
 ↪training-------------------------------------')
start = timeit.default_timer()
history_fc = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=2, validation_data=\
                    (test_imgs, test_labels),callbacks =[tensorboard])


gc.collect()
##Unfreeze weights ------ Now full model will be trained
for layer in model_updated.layers:
    layer.trainable = True


no_epochs = 10
tensorboard = TensorBoard(log_dir="logs\{}".
 ↪format('CALTECH256\FULL_Model'+str(i)))

opt2 = optimizers.Adam(lr=0.00001)

### Variable learning rate -----------------------
lrate = LearningRateScheduler(learning_rate_schedule)
callbacks_list = [lrate,tensorboard]




model_updated.compile(loss='categorical_crossentropy', optimizer=opt2,␣
 ↪metrics=['accuracy'])


print('-------------------------Starting Full model training -->after␣
 ↪unfreez-------------------------------------')

## Training full model
history = model_updated.fit(train_imgs, train_labels, batch_size=128,\
shuffle=True, epochs=no_epochs, validation_data=\
                    (test_imgs, test_labels),callbacks =callbacks_list)



## Timer for checking time taken
stop = timeit.default_timer()
print('Time taken for one run is : ', stop - start)
```

```
model_updated.save_weights('CALTECH256_VGG16_model_updated_fin.h5')
print("Model_Updated weights saved")

## saving history for plots
history_fc_list.append(history_fc)
history_full_list.append(history)
```

```
14
-----------------------New run started----------------------------------
Initial model weights loaded , Started training run number  3
------------------------------------------------------------------------
-------------------------Starting Fully connected layer
training-----------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 5.4953 -
acc: 0.2242 - val_loss: 3.2557 - val_acc: 0.6003
Epoch 2/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 2.8511 -
acc: 0.6257 - val_loss: 2.6105 - val_acc: 0.6857
-------------------------Starting Full model training -->after
unfreez-----------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 2.0379 -
acc: 0.7687 - val_loss: 2.1689 - val_acc: 0.7331
Epoch 2/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 1.3977 -
acc: 0.9168 - val_loss: 2.0157 - val_acc: 0.7557
Epoch 3/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 1.1047 -
acc: 0.9733 - val_loss: 1.9355 - val_acc: 0.7656
Epoch 4/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.9498 -
acc: 0.9912 - val_loss: 1.8781 - val_acc: 0.7729
Epoch 5/10
15420/15420 [==============================] - 169s 11ms/sample - loss: 0.8401 -
acc: 0.9977 - val_loss: 1.8450 - val_acc: 0.7781
Epoch 6/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.7703 -
acc: 0.9992 - val_loss: 1.8048 - val_acc: 0.7801
Epoch 7/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.7131 -
acc: 0.9995 - val_loss: 1.7960 - val_acc: 0.7835
Epoch 8/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.6676 -
acc: 0.9997 - val_loss: 1.7900 - val_acc: 0.7860
```

```
Epoch 9/10
15420/15420 [==============================] - 168s 11ms/sample - loss: 0.6290 -
acc: 1.0000 - val_loss: 1.7630 - val_acc: 0.7883
Epoch 10/10
15420/15420 [==============================] - 167s 11ms/sample - loss: 0.6032 -
acc: 1.0000 - val_loss: 1.7744 - val_acc: 0.7894
Time taken for one run is :  1878.8997759830017
Model_Updated weights saved
```

2.6 Tensorboard logs have been save to logs folder

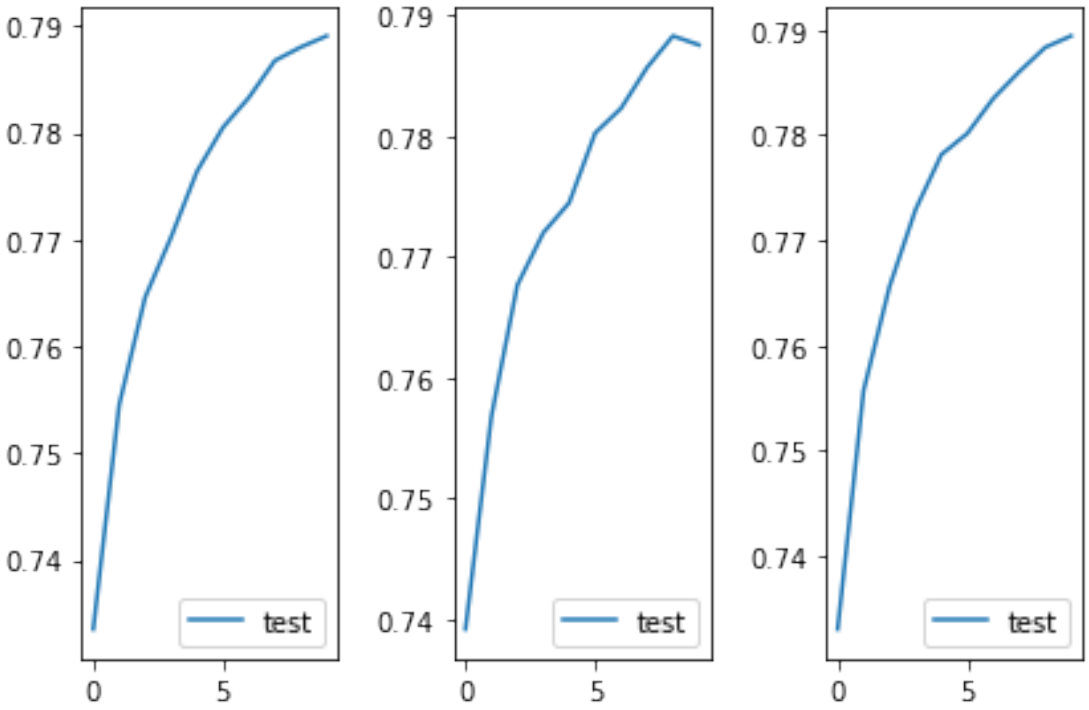
- where "*_Layer" folder means logs where we train only our FC layers
- where "*_Model" folder means logs where we train all layers in our model

2.7 Plotting the graphs of testing accuracy

- Test accuracy
- Plotted for only FC_layer training
- Plotted for full model training

[27]:
```
plt.subplot(1, 3, 1)
plt.plot(history_full_list[0].history['val_acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 2)
plt.plot(history_full_list[1].history['val_acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 3)
plt.plot(history_full_list[2].history['val_acc'])
plt.legend(['test'], loc='lower right')
plt.tight_layout()

plt.show()

avg_train_acc = 0
avg_test_acc = 0
final_train_loss = []
final_test_loss = []

for his in history_full_list:
    final_train_loss.append(his.history['loss'][-1])
    final_test_loss.append(his.history['val_loss'][-1])
    avg_train_acc = avg_train_acc + his.history['acc'][-1]
    avg_test_acc = avg_test_acc + his.history['val_acc'][-1]

avg_train_acc = avg_train_acc/len(history_fc_list)
avg_test_acc = avg_test_acc/len(history_fc_list)
```

```
print("Average testing accuracy: {}".format(avg_test_acc))
```

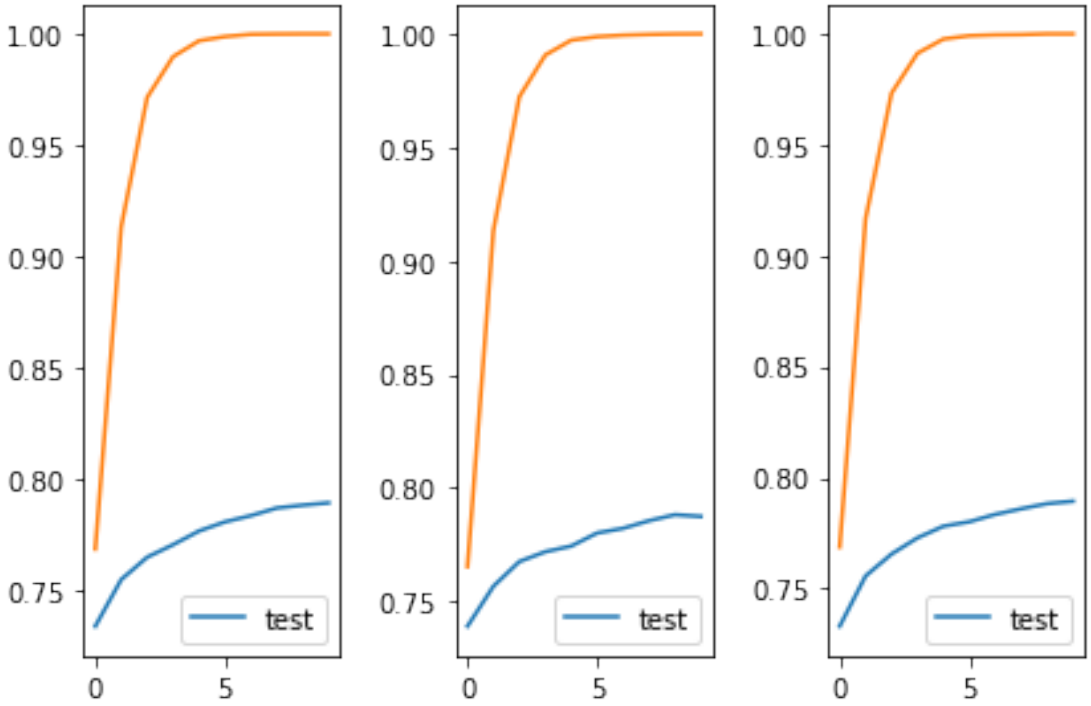



```
Average testing accuracy: 0.7886622746785482
```

2.8 Train acc vs Test acc

[28]:
```
plt.subplot(1, 3, 1)
plt.plot(history_full_list[0].history['val_acc'])
plt.plot(history_full_list[0].history['acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 2)
plt.plot(history_full_list[1].history['val_acc'])
plt.plot(history_full_list[1].history['acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 3)
plt.plot(history_full_list[2].history['val_acc'])
plt.plot(history_full_list[2].history['acc'])
plt.legend(['test'], loc='lower right')
plt.tight_layout()
```

```
plt.show()
print("Average train accuracy: {}".format(avg_train_acc))
```



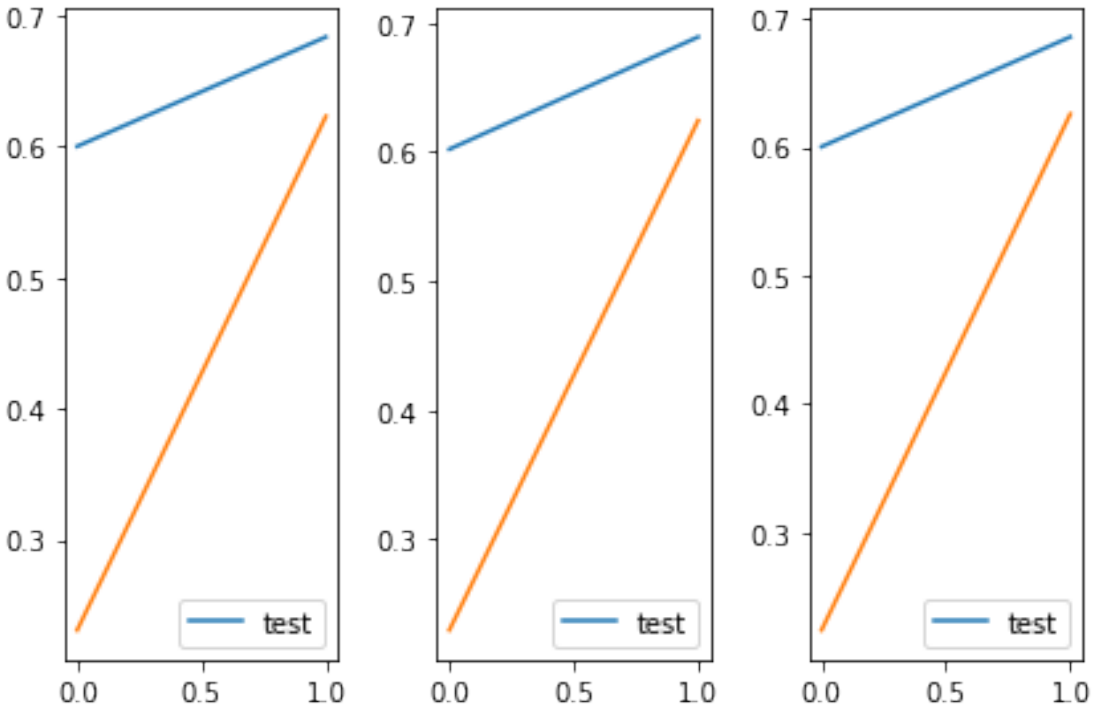

```
Average train accuracy: 1.0
```

2.9 FC layers test and train accuracy

- Fc layers training accuracy
- graphs look linear since we have only 2 epoch
- accuracy graph of full model training is plotted above

[31]:
```
plt.subplot(1, 3, 1)
plt.plot(history_fc_list[0].history['val_acc'])
plt.plot(history_fc_list[0].history['acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 2)
plt.plot(history_fc_list[1].history['val_acc'])
plt.plot(history_fc_list[1].history['acc'])
plt.legend(['test'], loc='lower right')

plt.subplot(1, 3, 3)
plt.plot(history_fc_list[2].history['val_acc'])
plt.plot(history_fc_list[2].history['acc'])
```

```
plt.legend(['test'], loc='lower right')
plt.tight_layout()

plt.show()
```



3 Final model average accuracy on test set

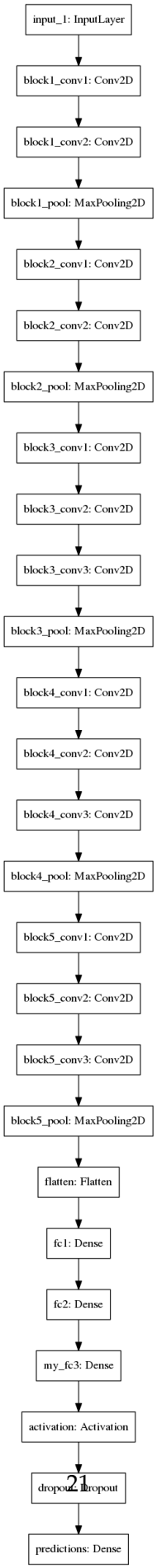
[29]:
```
print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")
```

```
Average Model Train accuracy is  :  78.86622746785483 %
```

3.0.1 Model structure is saved to an image.

[30]:
```
from tensorflow.keras.utils import plot_model
plot_model(model_updated, to_file='model_CALTECH256_VGG16.png')
```

[30]:


input_1: InputLayer
block1_conv1: Conv2D
block1_conv2: Conv2D
block1_pool: MaxPooling2D
block2_conv1: Conv2D
block2_conv2: Conv2D
block2_pool: MaxPooling2D
block3_conv1: Conv2D
block3_conv2: Conv2D
block3_conv3: Conv2D
block3_pool: MaxPooling2D
block4_conv1: Conv2D
block4_conv2: Conv2D
block4_conv3: Conv2D
block4_pool: MaxPooling2D
block5_conv1: Conv2D
block5_conv2: Conv2D
block5_conv3: Conv2D
block5_pool: MaxPooling2D
flatten: Flatten
fc1: Dense
fc2: Dense
my_fc3: Dense
activation: Activation
dropout: Dropout
predictions: Dense