ML_Caltech101_single_VGG16-Data_Agumentation-SR

November 24, 2019

1 Final project Caltech101

-- 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 = './101_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 :  38.663071474999995
```

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])
```

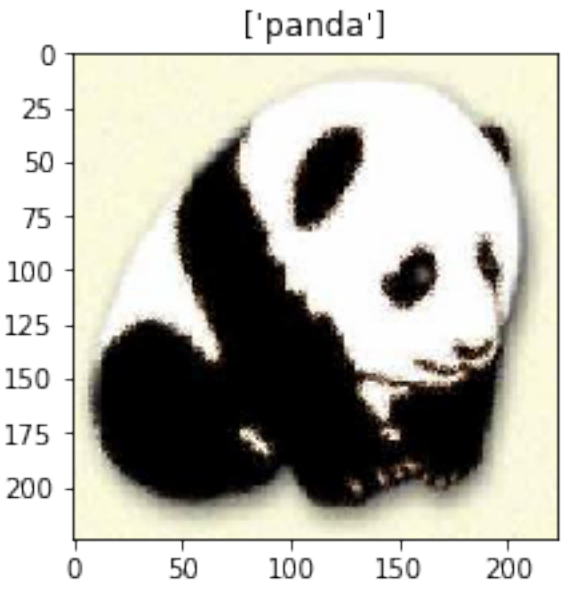
```
Shape of train data:  (3060, 224, 224, 3)
Shape of test data:  (6084, 224, 224, 3)
Shape of train labels:  (3060,)
Shape of test labels:  (6084,)
Total instances:  9144
```

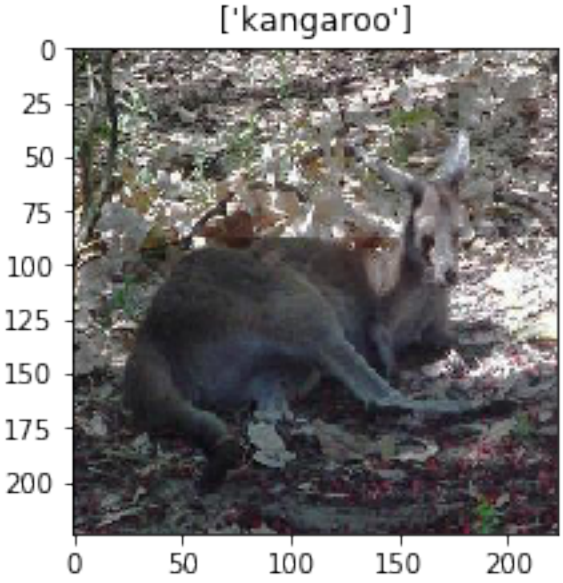
1.3 Flipping images for image augmentation

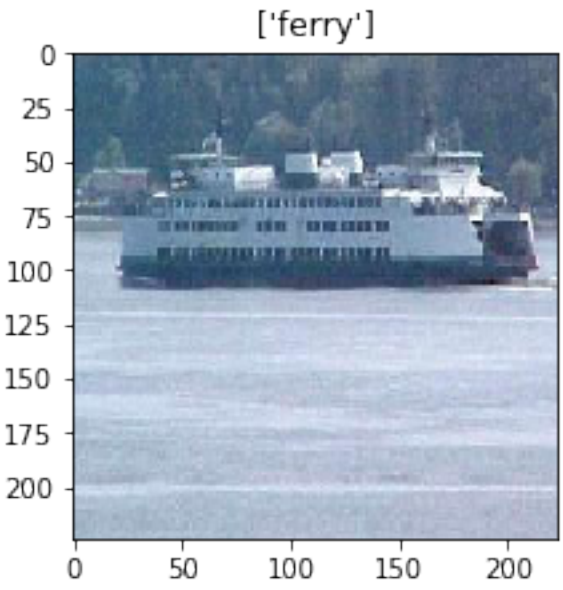
[4]:
```
train_imgs_flip_horizontal = np.flip(train_imgs,axis=2)

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

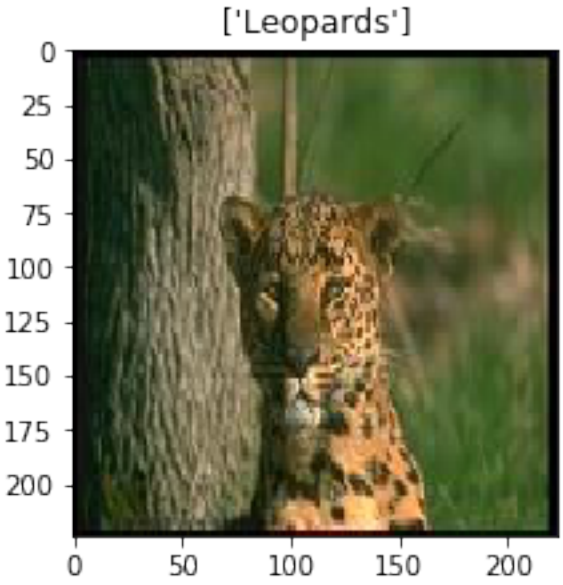
1.4 Plotting some random images

[5]:
```
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()
```


['panda']
0
25
50
75
100
125
150
175
200
0
50
100
150
200


['kangaroo']
0
25
50
75
100
125
150
175
200
0
50
100
150
200





2 Preprocess image using VGG function image

[6]:
```
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 :  5.102038510999989
```

[6]:
```
9136
```

[7]:
```
#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
```

```
(6120, 224, 224, 3)
```

[7]:
```
(6084, 224, 224, 3)
```

2.1 Applying one hot encoding on our labels

[8]:
```
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:  (6120, 102)
Shape of test labels:  (6084, 102)
```

[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.

[12]:
```
#define variable learning rates

def learning_rate_schedule(epoch):
    if epoch <= 3:
        return 1e-4 # 0.00001
    elif epoch <= 8:
        return 1e-5
    elif epoch <= 13:
        return 1e-6
    else:
        return 1e-7
    return LR
```

[13]:
```
#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(600, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(102, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)
```

```
## save our model weights
model_updated.save_weights('CALTECH101_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

[15]:
```
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(600, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(102, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH101_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('CALTECH101\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('CALTECH101\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('CALTECH101_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 6120 samples, validate on 6084 samples
Epoch 1/2
6120/6120 [==============================] - 38s 6ms/sample - loss: 4.7781 -
acc: 0.2312 - val_loss: 2.3259 - val_acc: 0.7217
Epoch 2/2
6120/6120 [==============================] - 37s 6ms/sample - loss: 2.2889 -
acc: 0.6538 - val_loss: 1.6120 - val_acc: 0.8172
-------------------------Starting Full model training -->after
unfreez-------------------------------------
Train on 6120 samples, validate on 6084 samples
Epoch 1/10
6120/6120 [==============================] - 71s 12ms/sample - loss: 1.4769 -
acc: 0.7735 - val_loss: 0.8499 - val_acc: 0.9043
Epoch 2/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.5037 -
acc: 0.9840 - val_loss: 0.7030 - val_acc: 0.9145
Epoch 3/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.3414 -
acc: 0.9992 - val_loss: 0.6253 - val_acc: 0.9254
Epoch 4/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2883 -
acc: 0.9995 - val_loss: 0.6202 - val_acc: 0.9259
Epoch 5/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2639 -
acc: 1.0000 - val_loss: 0.6193 - val_acc: 0.9259
Epoch 6/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2596 -
acc: 1.0000 - val_loss: 0.6213 - val_acc: 0.9267
Epoch 7/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2562 -
acc: 1.0000 - val_loss: 0.6196 - val_acc: 0.9270
```

```
Epoch 8/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2518 -
acc: 1.0000 - val_loss: 0.6219 - val_acc: 0.9260
Epoch 9/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2487 -
acc: 1.0000 - val_loss: 0.6212 - val_acc: 0.9274
Epoch 10/10
6120/6120 [==============================] - 70s 11ms/sample - loss: 0.2461 -
acc: 1.0000 - val_loss: 0.6216 - val_acc: 0.9272
Time taken for one run is :  1065.469159298
Model_Updated weights saved
```

2.4 Run 2

[17]:
```
#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(600, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(102, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH101_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('CALTECH101\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('CALTECH101\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('CALTECH101_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)
```

```
395
------------------------New run started---------------------------------
Initial model weights loaded , Started training run number  2
------------------------------------------------------------------------
-------------------------Starting Fully connected layer
training-----------------------------------
Train on 6120 samples, validate on 6084 samples
Epoch 1/2
6120/6120 [==============================] - 33s 5ms/sample - loss: 4.6895 -
acc: 0.2400 - val_loss: 2.3079 - val_acc: 0.7257
Epoch 2/2
6120/6120 [==============================] - 32s 5ms/sample - loss: 2.2616 -
acc: 0.6642 - val_loss: 1.5899 - val_acc: 0.8192
-------------------------Starting Full model training -->after
unfreez-------------------------------------
Train on 6120 samples, validate on 6084 samples
Epoch 1/10
6120/6120 [==============================] - 60s 10ms/sample - loss: 1.5012 -
acc: 0.7732 - val_loss: 0.8586 - val_acc: 0.8981
Epoch 2/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.5132 -
acc: 0.9820 - val_loss: 0.6917 - val_acc: 0.9177
Epoch 3/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.3432 -
acc: 0.9990 - val_loss: 0.6366 - val_acc: 0.9257
Epoch 4/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2865 -
acc: 1.0000 - val_loss: 0.6294 - val_acc: 0.9267
Epoch 5/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2640 -
acc: 1.0000 - val_loss: 0.6205 - val_acc: 0.9283
Epoch 6/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2596 -
acc: 1.0000 - val_loss: 0.6176 - val_acc: 0.9301
Epoch 7/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2550 -
acc: 1.0000 - val_loss: 0.6174 - val_acc: 0.9298
Epoch 8/10
```

```
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2527 -
acc: 1.0000 - val_loss: 0.6174 - val_acc: 0.9290
Epoch 9/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2491 -
acc: 1.0000 - val_loss: 0.6186 - val_acc: 0.9300
Epoch 10/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2468 -
acc: 1.0000 - val_loss: 0.6189 - val_acc: 0.9300
Time taken for one run is :  1849.665929521
Model_Updated weights saved
```

2.5 Run 3

[18]:
```
#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(600, activity_regularizer=l1(0.001), name='my_fc3')(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
x = Dense(102, activation='softmax', name='predictions')(x)
model_updated = Model(inputs=vgg_net.input, outputs=x)

gc.collect()


model_updated.load_weights('CALTECH101_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('CALTECH101\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('CALTECH101\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('CALTECH101_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 6120 samples, validate on 6084 samples
Epoch 1/2
6120/6120 [==============================] - 32s 5ms/sample - loss: 4.7494 -
acc: 0.2299 - val_loss: 2.3026 - val_acc: 0.7240
Epoch 2/2
6120/6120 [==============================] - 32s 5ms/sample - loss: 2.2933 -
acc: 0.6529 - val_loss: 1.6217 - val_acc: 0.8116
-------------------------Starting Full model training -->after
unfreez-----------------------------------
Train on 6120 samples, validate on 6084 samples
Epoch 1/10
6120/6120 [==============================] - 60s 10ms/sample - loss: 1.4404 -
acc: 0.7859 - val_loss: 0.8135 - val_acc: 0.9190
Epoch 2/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.5049 -
acc: 0.9828 - val_loss: 0.6575 - val_acc: 0.9301
Epoch 3/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.3412 -
acc: 0.9989 - val_loss: 0.6281 - val_acc: 0.9282
Epoch 4/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2862 -
acc: 0.9998 - val_loss: 0.6056 - val_acc: 0.9321
Epoch 5/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2631 -
acc: 1.0000 - val_loss: 0.6168 - val_acc: 0.9320
Epoch 6/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2593 -
acc: 1.0000 - val_loss: 0.6139 - val_acc: 0.9313
Epoch 7/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2550 -
acc: 1.0000 - val_loss: 0.6114 - val_acc: 0.9321
Epoch 8/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2520 -
acc: 1.0000 - val_loss: 0.6105 - val_acc: 0.9321
```

```
Epoch 9/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2483 -
acc: 1.0000 - val_loss: 0.6108 - val_acc: 0.9326
Epoch 10/10
6120/6120 [==============================] - 59s 10ms/sample - loss: 0.2463 -
acc: 1.0000 - val_loss: 0.6108 - val_acc: 0.9326
Time taken for one run is :  662.3431134560001
Model_Updated weights saved
```

2.6 Tensorboard logs have been save to logs folder

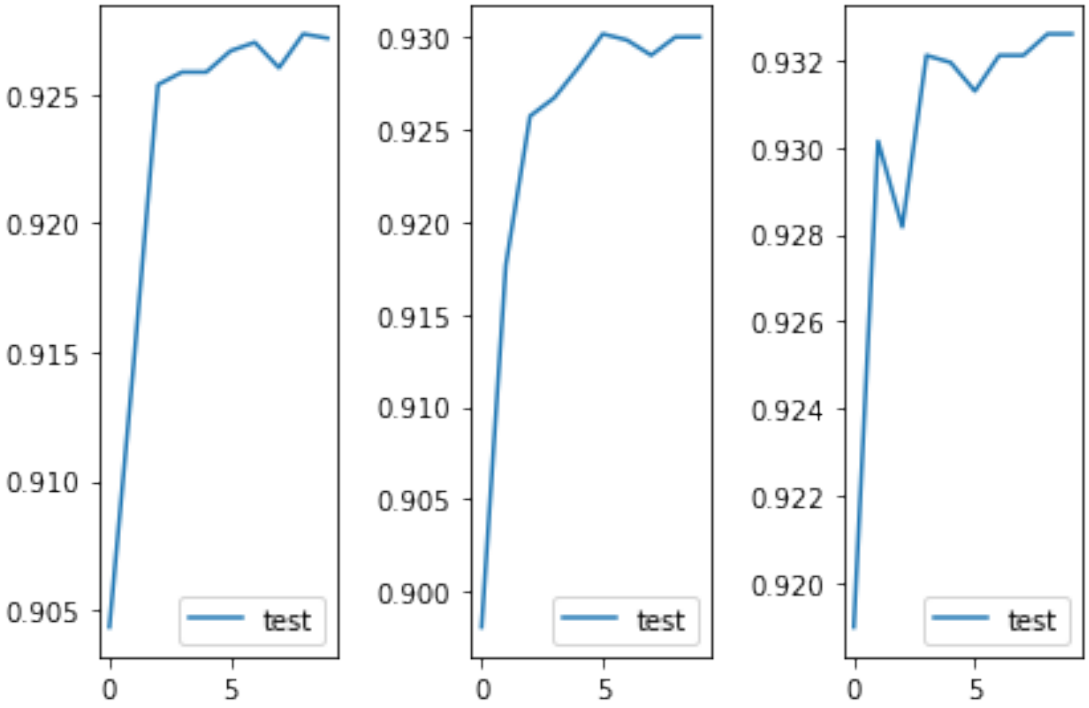
- where "*_Layer" folder means logs where we train only our FC layers
- where "*_Model" folder means logs where we train all layers in our model

2.7 Plotting the graphs of testing accuracy

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

[19]:
```
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))
```

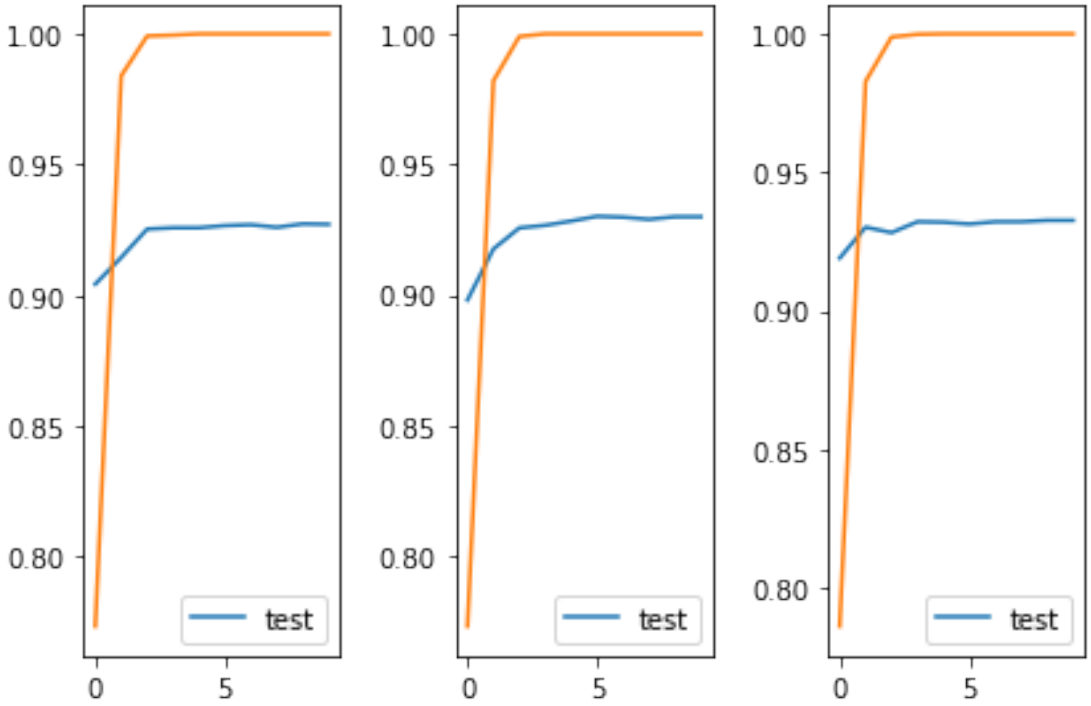



```
Average testing accuracy: 0.9299255013465881
```

2.8 Train acc vs Test acc

[20]:
```
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))
```



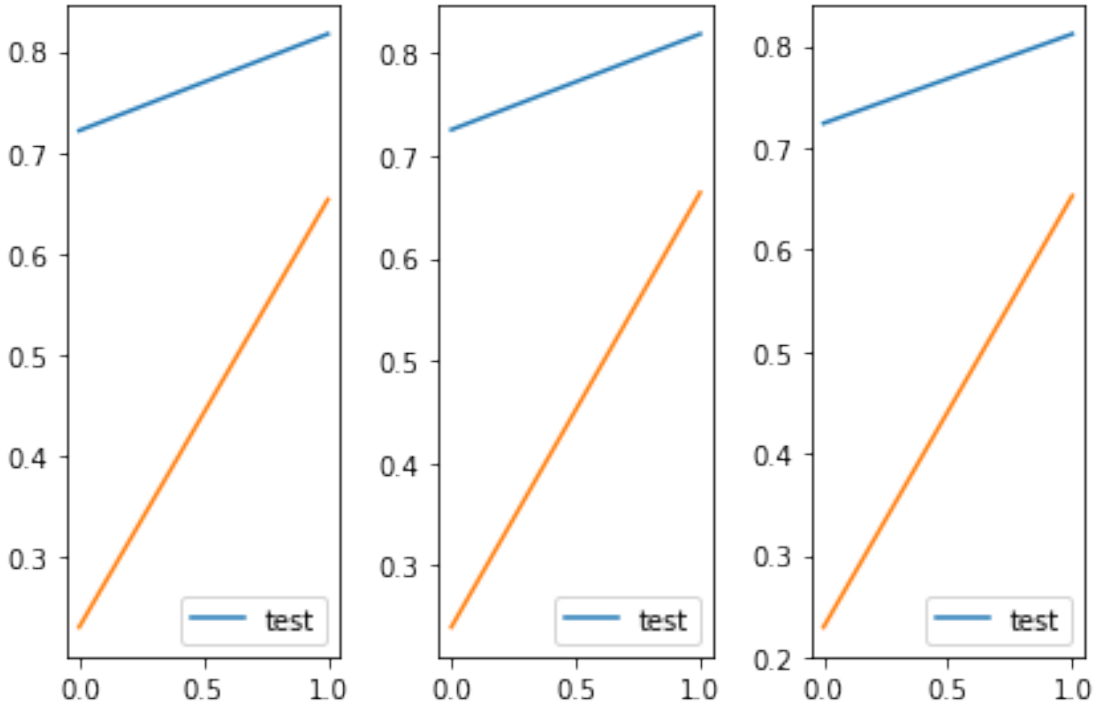

```
Average train accuracy: 1.0
```

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

[21]:
```
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

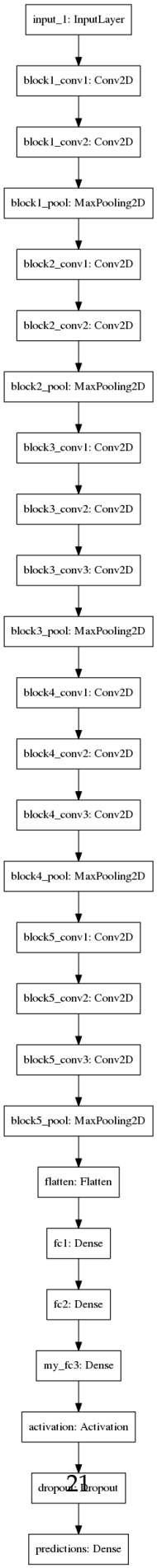
[22]:
```
print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")
```

```
Average Model Train accuracy is  :  92.99255013465881 %
```

3.0.1 Model structure is saved to an image.

[23]:
```
from tensorflow.keras.utils import plot_model
plot_model(model_updated, to_file='model_CALTECH101_VGG16.png')
```

[23]:


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