Transfer_Learning

August 18, 2020

[1]:
```
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
```

[2]:
```
tf.__version__
```

[2]:
```
'2.3.0'
```

[4]:
```
!wget --header="Host: doc-0o-bo-docs.googleusercontent.com"␣
↪--header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
↪537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"␣
↪--header="Accept: text/html,application/xhtml+xml,application/xml;q=0.
↪9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9"␣
↪--header="Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7"␣
↪--header="Referer: https://drive.google.com/drive/folders/
↪1eY4pTqtGtXSAYVzaHwWAK2Cq4bHb1Zkq" --header="Cookie:␣
↪AUTH_gnb78hdmdiks9t0b8kec09hpa7nncs5e_nonce=3anpub0rvhtke; _ga=GA1.2.
↪1804417035.1594643089;␣
↪NID=204=WE4tnDNREeWA-i7VkayXZRT0nZVp1JeYYk6hLWon_UP0rMptB4l1jfZkhNOvbLUPVxDowH066xA42Zz173_
↪--header="Connection: keep-alive" "https://doc-0o-bo-docs.googleusercontent.
↪com/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/
↪racrhu4in6ir0gdp8b9fnehnnrlpgjlo/1597721850000/00484516897554883881/
↪03515051603858730688/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu?
↪e=download&authuser=0&nonce=3anpub0rvhtke&user=03515051603858730688&hash=97r5k1fqf0kkoe37ur
↪-c -O 'rvl-cdip.rar'
```

```
--2020-08-18 03:38:49--  https://doc-0o-bo-docs.googleusercontent.com/docs/secur
esc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/racrhu4in6ir0gdp8b9fnehnnrlpgjlo/1597721850
000/00484516897554883881/03515051603858730688/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu?
e=download&authuser=0&nonce=3anpub0rvhtke&user=03515051603858730688&hash=97r5k1f
qf0kkoe37urskk9paglbenr71
Resolving doc-0o-bo-docs.googleusercontent.com (doc-0o-bo-
docs.googleusercontent.com)... 74.125.20.132, 2607:f8b0:400e:c07::84
Connecting to doc-0o-bo-docs.googleusercontent.com (doc-0o-bo-
docs.googleusercontent.com)|74.125.20.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: unspecified [application/rar]
Saving to: rvl-cdip.rar

rvl-cdip.rar            [           <=>      ]   4.34G  58.1MB/s    in 87s

2020-08-18 03:40:17 (51.1 MB/s) - rvl-cdip.rar saved [4660541790]
```

[5]:
```
get_ipython().system_raw("unrar x rvl-cdip.rar")
```

[6]:
```
import os
os.listdir()
```

[6]:
```
['.config', 'data_final', 'rvl-cdip.rar', 'labels_final.csv', 'sample_data']
```

[7]:
```
!pwd
```

```
/content
```

[8]:
```
import pandas as pd
```

[9]:
```
df = pd.read_csv("labels_final.csv")
df['label'] = df['label'].astype(str)
```

[10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split( df, test_size=0.33, random_state=42)
```

[11]:
```
datagen=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
train_generator=datagen.flow_from_dataframe(dataframe=X_train,␣
 ↪directory="data_final", x_col="path", y_col="label",

                                                 ␣
 ↪class_mode="categorical",target_size=(224,224), batch_size=64)
test_generator=datagen.flow_from_dataframe(dataframe=X_test,␣
 ↪directory="data_final", x_col="path", y_col="label",

                                                 ␣
 ↪class_mode="categorical",target_size=(224,224), batch_size=64)
```

```
Found 32160 validated image filenames belonging to 16 classes.
Found 15840 validated image filenames belonging to 16 classes.
```

[12]:
```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import␣
 ↪Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense,Input
from tensorflow.keras.models import Model
import tensorflow as tf
```

[13]:
```
import datetime
%load_ext tensorboard
```

[14]:
```
import numpy as np
```

1 Model_1

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, add a new Conv block (1 Conv layer and 1 Max-pooling), 2 FC layers and a output layer to classify 16 classes. You are free to choose any hyperparameters/parameters of conv block, FC layers, output layer.
3. Final architecture will be INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output Layer
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

[15]:
```
tf.keras.backend.clear_session()
```

[16]:
```
img_shape=(224,224,3)
vgg = VGG16(include_top=False,weights="imagenet",input_shape=img_shape)
vgg.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
```

[32]:
```
from keras import regularizers
```

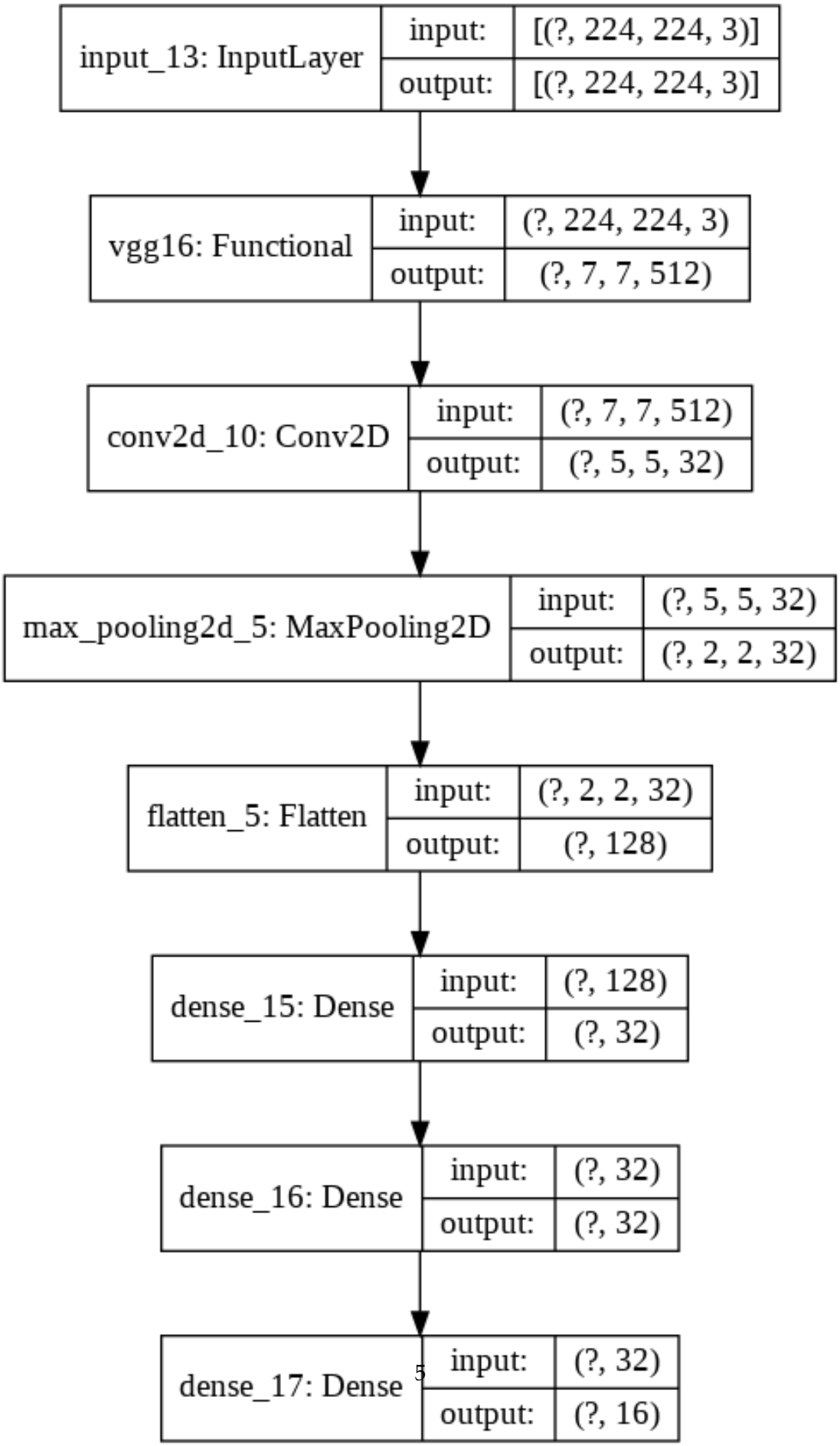
[38]:
```
input_layer = Input(shape=(224,224,3,))
layer=vgg(input_layer)
layer=Conv2D(32,kernel_size=(3,3),strides=(1,1),padding='valid',data_format='channels_last',
                  activation='relu',kernel_initializer=tf.keras.initializers.
 ↪he_normal(seed=0))(layer)
layer=MaxPool2D(pool_size=(2,2),strides=(2,2),padding='valid',data_format='channels_last')(lay
f=Flatten()(layer)
layer=Dense(32,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪glorot_normal(seed=0),kernel_regularizer=regularizers.l2(0.01))(f)
layer=Dense(32,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪glorot_normal(seed=0),kernel_regularizer=regularizers.l2(0.01))(layer)
output=Dense(16,activation="softmax",kernel_initializer=tf.keras.initializers.
 ↪glorot_normal(seed=0))(layer)
model = Model(inputs=input_layer,outputs=output)
model.summary()
```

```
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 224, 224, 3)]     0
```

```
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 5, 5, 32)          147488
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 32)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_16 (Dense)             (None, 32)                1056
_________________________________________________________________
dense_17 (Dense)             (None, 16)                528
=================================================================
Total params: 14,867,888
Trainable params: 153,200
Non-trainable params: 14,714,688
_________________________________________________________________
```

[39]:
```
import keras
keras.utils.plot_model(model, 'model.png', show_shapes=True)
```

[39]:


input_13: InputLayer
input: [(?, 224, 224, 3)]
output: [(?, 224, 224, 3)]
vgg16: Functional
input: (?, 224, 224, 3)
output: (?, 7, 7, 512)
conv2d_10: Conv2D
input: (?, 7, 7, 512)
output: (?, 5, 5, 32)
max_pooling2d_5: MaxPooling2D
input: (?, 5, 5, 32)
output: (?, 2, 2, 32)
flatten_5: Flatten
input: (?, 2, 2, 32)
output: (?, 128)
dense_15: Dense
input: (?, 128)
output: (?, 32)
dense_16: Dense
input: (?, 32)
output: (?, 32)
dense_17: Dense
input: (?, 32)
output: (?, 16)

[40]:
```
model.compile(optimizer= tf.keras.optimizers.Adam(lr=0.
 ↪001),loss='categorical_crossentropy',metrics=['accuracy'])
np.random.seed(0)
log_dir="Model_1" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model.fit_generator(
        train_generator,
        steps_per_epoch=(32160 //64),
        epochs=6,
        validation_data=test_generator,
        validation_steps=(15840 // 64),callbacks=[tensorboard])
```

```
Epoch 1/6
502/502 [==============================] - 245s 489ms/step - loss: 2.0789 -
accuracy: 0.4630 - val_loss: 1.6340 - val_accuracy: 0.5659
Epoch 2/6
502/502 [==============================] - 267s 531ms/step - loss: 1.5140 -
accuracy: 0.6045 - val_loss: 1.4718 - val_accuracy: 0.6103
Epoch 3/6
502/502 [==============================] - 272s 542ms/step - loss: 1.3527 -
accuracy: 0.6479 - val_loss: 1.3864 - val_accuracy: 0.6327
Epoch 4/6
502/502 [==============================] - 244s 487ms/step - loss: 1.2504 -
accuracy: 0.6751 - val_loss: 1.3359 - val_accuracy: 0.6473
Epoch 5/6
502/502 [==============================] - 256s 510ms/step - loss: 1.1630 -
accuracy: 0.6984 - val_loss: 1.3041 - val_accuracy: 0.6525
Epoch 6/6
502/502 [==============================] - 263s 523ms/step - loss: 1.1131 -
accuracy: 0.7108 - val_loss: 1.2627 - val_accuracy: 0.6710
```

[40]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6ed0107f60>
```

[41]:
```
%tensorboard --logdir=/content/Model_120200818-041746
```

```
Output hidden; open in https://colab.research.google.com to view.
```

2 Model_2

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fully connected layer. any FC layer can be converted to a CONV layer. This conversion will reduce the No of Trainable parameters in FC layers. For example, an FC layer with K=4096 that is

looking at some input volume of size 7Œ7Œ512 can be equivalently expressed as a CONV layer with F=7,P=0,S=1,K=4096. In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be 1Œ1Œ4096 since only a single depth column "fits" across the input volume, giving identical result as the initial FC layer. You can refer this link to better understanding of using Conv layer in place of fully connected layers.

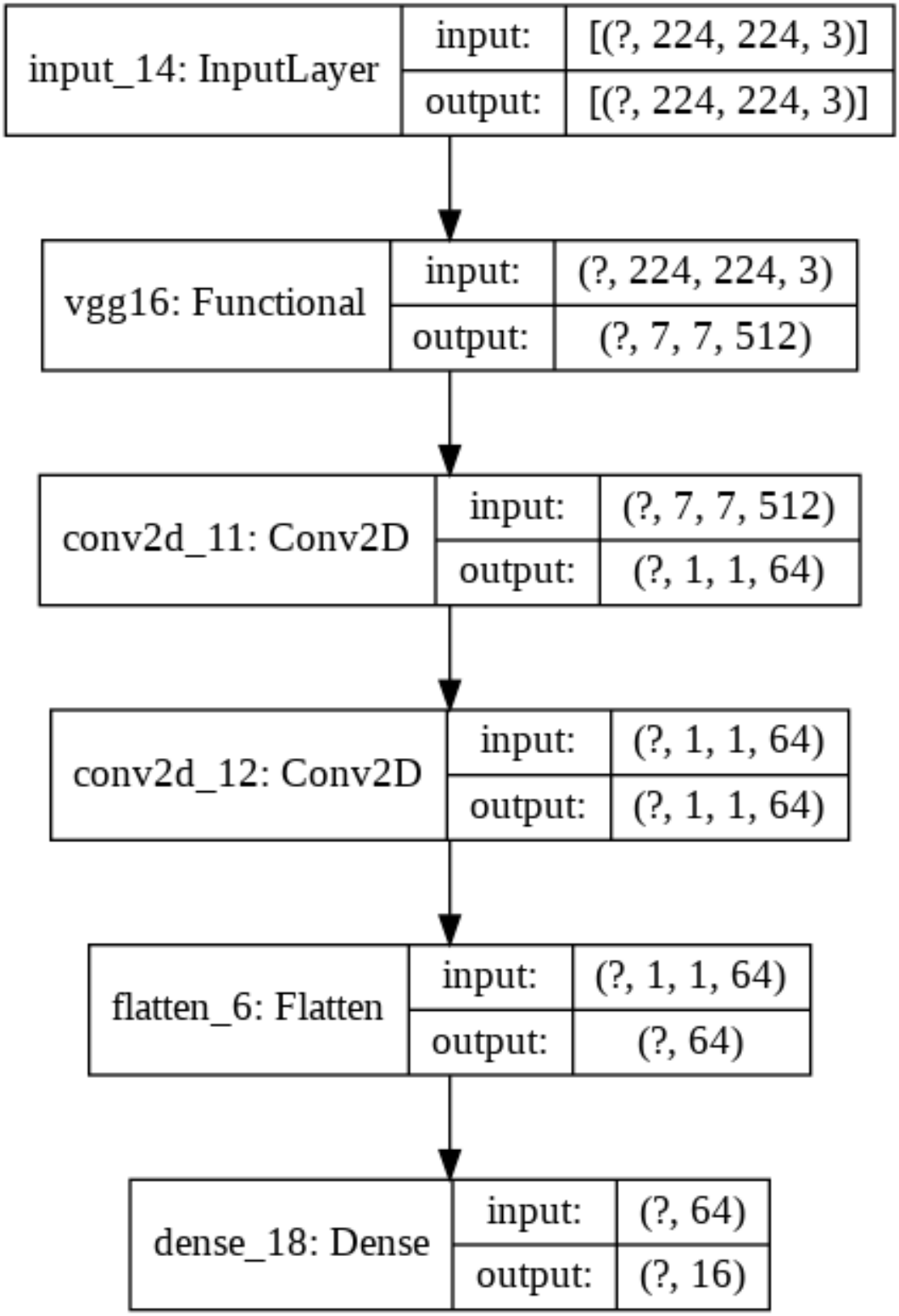
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 class classification. INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer
4. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

[42]:
```
input_layer = Input(shape=(224,224,3,))
layer=vgg(input_layer)
layer=Conv2D(filters=64,kernel_size=(7,7),strides=(1,1),padding='valid',data_format='channels_
                activation='relu',kernel_initializer=tf.keras.initializers.
 ↪he_normal(seed=0),kernel_regularizer=regularizers.l2(0.01))(layer)
layer=Conv2D(filters=64,kernel_size=(1,1),strides=(1,1),padding='valid',data_format='channels_
                activation='relu',kernel_initializer=tf.keras.initializers.
 ↪he_normal(seed=0),kernel_regularizer=regularizers.l2(0.01))(layer)
f=Flatten()(layer)
output=Dense(16,activation="softmax",kernel_initializer=tf.keras.initializers.
 ↪glorot_normal(seed=3))(f)
model = Model(inputs=input_layer,outputs=output)
model.summary()
```

```
Model: "functional_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 64)          1605696
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 1, 1, 64)          4160
_________________________________________________________________
flatten_6 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_18 (Dense)             (None, 16)                1040
=================================================================
Total params: 16,325,584
Trainable params: 1,610,896
Non-trainable params: 14,714,688
_________________________________________________________________
```

[43]:
```
import keras
keras.utils.plot_model(model, 'model.png', show_shapes=True)
```

[43]:


input_14: InputLayer
input: [(?, 224, 224, 3)]
output: [(?, 224, 224, 3)]
vgg16: Functional
input: (?, 224, 224, 3)
output: (?, 7, 7, 512)
conv2d_11: Conv2D
input: (?, 7, 7, 512)
output: (?, 1, 1, 64)
conv2d_12: Conv2D
input: (?, 1, 1, 64)
output: (?, 1, 1, 64)
flatten_6: Flatten
input: (?, 1, 1, 64)
output: (?, 64)
dense_18: Dense
input: (?, 64)
output: (?, 16)

[44]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.
 ↪001),loss='categorical_crossentropy',metrics=['accuracy'])
np.random.seed(0)
log_dir="Model_2" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model.fit_generator(
        train_generator,
        steps_per_epoch=(32160 // 64),
        epochs=6,
        validation_data=test_generator,
        validation_steps=(15840 // 64),callbacks=[tensorboard])
```

```
Epoch 1/6
502/502 [==============================] - 269s 536ms/step - loss: 2.2215 -
accuracy: 0.5382 - val_loss: 1.6942 - val_accuracy: 0.6095
Epoch 2/6
502/502 [==============================] - 271s 541ms/step - loss: 1.6148 -
accuracy: 0.6283 - val_loss: 1.5765 - val_accuracy: 0.6372
Epoch 3/6
502/502 [==============================] - 265s 527ms/step - loss: 1.5652 -
accuracy: 0.6368 - val_loss: 1.5479 - val_accuracy: 0.6364
Epoch 4/6
502/502 [==============================] - 269s 536ms/step - loss: 1.5281 -
accuracy: 0.6432 - val_loss: 1.5158 - val_accuracy: 0.6500
Epoch 5/6
502/502 [==============================] - 265s 527ms/step - loss: 1.4907 -
accuracy: 0.6547 - val_loss: 1.4600 - val_accuracy: 0.6683
Epoch 6/6
502/502 [==============================] - 268s 534ms/step - loss: 1.4743 -
accuracy: 0.6549 - val_loss: 1.5208 - val_accuracy: 0.6395
```

[44]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6c9b698c88>
```

[45]:
```
tf.keras.backend.clear_session()
```

[46]:
```
%tensorboard --logdir=/content/Model_220200818-044749
```

```
Output hidden; open in https://colab.research.google.com to view.
```

3 Model_3

1. Use same network as Model-2 'INPUT --> VGG-16 without Top layers(FC) -->
2. Conv Layers identical to FC --> Output Layer' and train only Last 6 Layers of VGG-16 network, 2 Conv layers identical to FC layers, 1 output layer.

[47]:
```
img_shape=(224,224,3)
vgg = VGG16(include_top=False,weights="imagenet",input_shape=img_shape)
for layer in vgg.layers[:13]:
    layer.trainable=False
input_layer = Input(shape=(224,224,3,))
layer=vgg(input_layer)
layer=Conv2D(filters=128,kernel_size=(7,7),strides=(1,1),padding='valid',data_format='channels
                activation='relu',kernel_initializer=tf.keras.initializers.
 ↪he_normal(seed=0))(layer)
layer=Conv2D(filters=128,kernel_size=(1,1),strides=(1,1),padding='valid',data_format='channels
                activation='relu',kernel_initializer=tf.keras.initializers.
 ↪he_normal(seed=0))(layer)
f=Flatten()(layer)
output=Dense(16,activation="softmax",kernel_initializer=tf.keras.initializers.
 ↪glorot_normal(seed=3))(f)
model = Model(inputs=input_layer,outputs=output)
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
conv2d (Conv2D)              (None, 1, 1, 128)         3211392
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 1, 128)         16512
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 16)                2064
=================================================================
Total params: 17,944,656
Trainable params: 12,669,200
Non-trainable params: 5,275,456
_________________________________________________________________
```

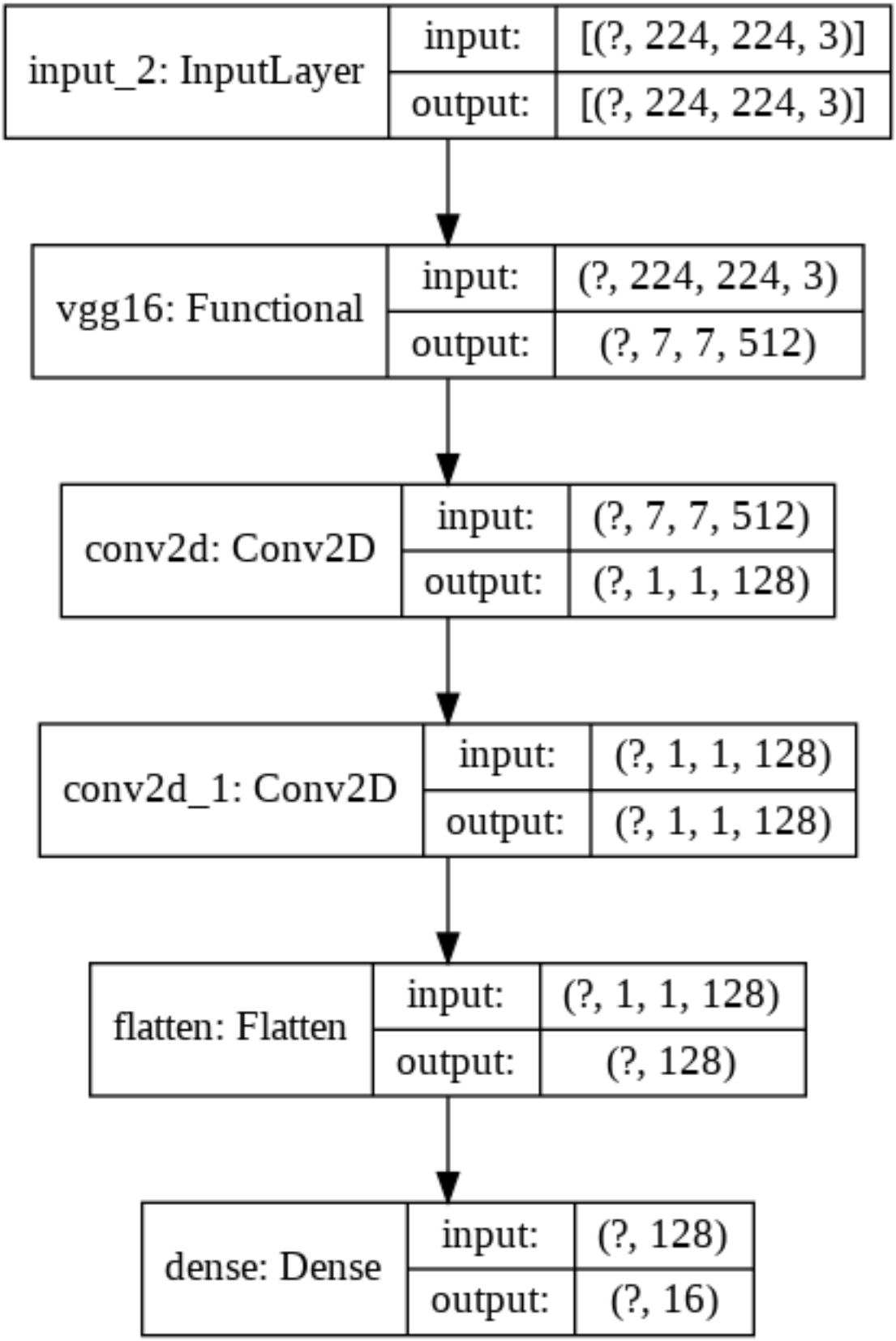
[48]:
```
for layer in vgg.layers:
  print(layer, layer.trainable)
```

```
<tensorflow.python.keras.engine.input_layer.InputLayer object at 0x7f6c9b485470>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6ed010a550>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b485f28>
```

```
False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7f6c9b496630>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b485ef0>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b4852e8>
False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7f6c9b48bf98>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b48b8d0>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b496e10>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b49e2e8>
False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7f6c9b4a65f8>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b49e9b0>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b49e5f8>
False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b4af860>
True
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7f6c9a435b70>
True
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b4aff28>
True
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9b4afb70>
True
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7f6c9a43edd8>
True
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7f6c9a445da0>
True
```

[49]:
```
import keras
keras.utils.plot_model(model, 'model.png', show_shapes=True)
```

[49]:


input_2: InputLayer
input: [(?, 224, 224, 3)]
output: [(?, 224, 224, 3)]
vgg16: Functional
input: (?, 224, 224, 3)
output: (?, 7, 7, 512)
conv2d: Conv2D
input: (?, 7, 7, 512)
output: (?, 1, 1, 128)
conv2d_1: Conv2D
input: (?, 1, 1, 128)
output: (?, 1, 1, 128)
flatten: Flatten
input: (?, 1, 1, 128)
output: (?, 128)
dense: Dense
input: (?, 128)
output: (?, 16)

[50]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.
 ↪001),loss='categorical_crossentropy',metrics=['accuracy'])
np.random.seed(0)
log_dir="Model3" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model.fit_generator(
        train_generator,
        steps_per_epoch=(32160 // 64),
        epochs=6,
        validation_data=test_generator,
        validation_steps=(15840 // 64),callbacks=[tensorboard])
```

```
Epoch 1/6
502/502 [==============================] - 277s 551ms/step - loss: 2.7868 -
accuracy: 0.0611 - val_loss: 2.7729 - val_accuracy: 0.0598
Epoch 2/6
502/502 [==============================] - 284s 566ms/step - loss: 2.7727 -
accuracy: 0.0599 - val_loss: 2.7732 - val_accuracy: 0.0607
Epoch 3/6
502/502 [==============================] - 271s 541ms/step - loss: 2.7727 -
accuracy: 0.0623 - val_loss: 2.7732 - val_accuracy: 0.0607
Epoch 4/6
502/502 [==============================] - 276s 550ms/step - loss: 2.7727 -
accuracy: 0.0613 - val_loss: 2.7732 - val_accuracy: 0.0607
Epoch 5/6
502/502 [==============================] - 273s 544ms/step - loss: 2.7726 -
accuracy: 0.0619 - val_loss: 2.7732 - val_accuracy: 0.0608
Epoch 6/6
502/502 [==============================] - 278s 555ms/step - loss: 2.7726 -
accuracy: 0.0629 - val_loss: 2.7731 - val_accuracy: 0.0608
```

[50]:
```
<tensorflow.python.keras.callbacks.History at 0x7f6c9a37aa58>
```

[51]:
```
tf.keras.backend.clear_session()
```

[52]:
```
%tensorboard --logdir=/content/Model320200818-051918
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[53]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model number", "Epoch", "Loss", "Accuracy", "Val loss", "Val␣
 ↪accuracy"]
x.add_row([1, 6, 1.1131,0.7108,1.2627,0.6710])
x.add_row([2, 6, 1.4743,0.6549,1.5208,0.6395])
x.add_row([3, 6, 2.7726,0.0629,2.7731,0.0608])
```

```
print(x)
```

```
+--------------+-------+--------+----------+----------+--------------+
| Model number | Epoch |  Loss  | Accuracy | Val loss | Val accuracy |
+--------------+-------+--------+----------+----------+--------------+
|      1       |   6   | 1.1131 |  0.7108  |  1.2627  |    0.671     |
|      2       |   6   | 1.4743 |  0.6549  |  1.5208  |    0.6395    |
|      3       |   6   | 2.7726 |  0.0629  |  2.7731  |    0.0608    |
+--------------+-------+--------+----------+----------+--------------+
```

Observations

1. In earlier submission my models are getting overfit. So, I have used L2 kernel_regularizer, which helped me in avoid overfitting problem.
2. In Model-3, Due to fully connected layers, the trainable parameters (14714688) increased compared to Model-1 and Model-2, the complexity of the model increases, where it leads decrease in Accuracy.
3. In all models, used "relu" and "softmax" functions for Conv2D, dense and output layers respectively.

Model-1 Observations

1. Most of the trainable parameters where taken from the pre-trained model (VGG16)
2. As the Epochs increasing, both (Train and Validation) the decreasing slightly but Accuracy's are increasing with large difference with respect to epochs (present and previous).
3. Used Adam optimizer with learning rate 0.001, which gave good results.

Model-2 Observations

1. We used only two Conv2D and one Dense layer in Model-2 with L2 regularizers of 0.01 to reduce overfitting.
2. As the both the loss decreasing slightly, accuracy increases slightly.
3. Model-2 gave less accuracy compered to Model-1 for 6 epochs.

Model-3 Observations

1. We have last six layers of pre-trained model VGG16
2. Accuracy is too low compared with Model-1 and Model-2
3. I think this is because model complexity.
4. Both the loss decreasing slightly and Accuary increasing slighlty (diffeeence of the accuarcy from present epoch to previous epcoh is 0.0001%)