Transfer Learning Assignment:

1. Download all the data in this folder https://drive.google.com/open?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu. it contains two file bo
   path/to/the/image.tif,category

   where the categories are numbered 0 to 15, in the following order:

   0 letter
   1 form
   2 email
   3 handwritten
   4 advertisement
   5 scientific report
   6 scientific publication
   7 specification
   8 file folder
   9 news article
   10 budget
   11 invoice
   12 presentation
   13 questionnaire
   14 resume
   15 memo

2. On this image data, you have to train 3 types of models as given below. You have to split the data into Train and Validation dat

3. Try not to load all the images into memory, use the gernarators that we have given the reference notebooks to load the batch of
or you can use this method also
https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1

https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c

4. You are free to choose Learning rate, optimizer, loss function, image augmentation, any hyperparameters. but you have to use the

5. Use tensorboard for every model and analyse your gradients. (you need to upload the screenshots for each model for evaluation)

Note: fit_genarator() method will have problems with the tensorboard histograms, try to debug it, if you could not do use histgrams

6. You can check about Transfer Learning in this link - https://blog.keras.io/building-powerful-image-classification-models-using-v

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 Maxpooling), 2 FC layers and a output layer t
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output L**
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuserc

Enter your authorization code:
```

```
#importing tensorflow
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import random as rn
import pandas as pd
import numpy as np
import tensorflow as tf
import os
data = pd.read_csv('/gdrive/My Drive/transfer_learning/labels_final.csv')


labels = {0: 'letter', 1:'form', 2: 'email', 3: 'handwritten', 4: 'advertisement', 5: 'scientific report', 6: 'scientific publication', 7: 'specifica


import pathlib
data_root = pathlib.Path('/gdrive/My Drive/transfer_learning/data_final/')
print(data_root)
##Getting all image paths
import random
all_image_paths = list(data_root.glob('*/*'))
all_image_paths = [str(path) for path in all_image_paths]
# ##shuffling the images
# random.shuffle(all_image_paths)

image_count = len(all_image_paths)
label_names = list(labels.values())
label_to_index = dict((name, index) for index,name in enumerate(label_names))
label_to_index
```

```
/gdrive/My Drive/transfer_learning/data_final
{'advertisement': 4,
 'budget': 10,
 'email': 2,
 'file folder': 8,
 'form': 1,
 'handwritten': 3,
 'invoice': 11,
 'letter': 0,
 'memo': 15,
 'news article': 9,
 'presentation': 12,
 'questionnaire': 13,
 'resume': 14,
 'scientific publication': 6,
 'scientific report': 5,
 'specification': 7}
```

```
path = '/gdrive/My Drive/transfer_learning/data_final/images'
pathlib.Path(path).name
```

```
'images'
```

```
all_image_paths = [str('/gdrive/My Drive/transfer_learning/data_final/' + i)  for i in list(data['path'])]
all_image_labels = list(data['label'])


import matplotlib
matplotlib.use("Agg")

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import os


data.head()
```

	path	label
0	imagesv/v/o/h/voh71d00/509132755+-2755.tif	3
1	imagesl/l/x/t/lxt19d00/502213303.tif	3
2	imagesx/x/e/d/xed05a00/2075325674.tif	2
3	imageso/o/j/b/ojb60d00/517511301+-1301.tif	3
4	imagesq/q/z/k/qzk17e00/2031320195.tif	7

```
import os
import datetime
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.95,patience=1,mode = 'min')

os.environ['PYTHONHASHSEED'] = '0'

##https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development
## Have to clear the session. If you are not clearing, Graph will create again and again and graph size will increses.
## Varibles will also set to some value from before session
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)
vgg16_layer = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000, cla
for layer in vgg16_layer.layers:
  layer.trainable = False
#Input layer
input_layer = Input(shape=(156,256,3),name='Input_Layer')
#VGG16
vgg16_layer_output = vgg16_layer(input_layer)
#Conv Layer
Conv1 = Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding='valid',data_format='channels_last',
               activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(seed=0),name='Conv1')(vgg16_layer_output)
#MaxPool Layer
Pool1 = MaxPool2D(pool_size=(2,2),strides=(2,2),padding='valid',data_format='channels_last',name='Pool1')(Conv1)

#Flatten
flatten = Flatten(data_format='channels_last',name='Flatten')(Pool1)

#FC layer
FC1 = Dense(units=30,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(seed=32),name='FC1')(flatten)

#FC layer
FC2 = Dense(units=15,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(seed=32),name='FC2')(FC1)

#output layer
Out = Dense(units=16,activation='softmax',kernel_initializer=tf.keras.initializers.he_uniform(seed=32),name='Output')(FC2)
#TensorBoard Callback
log_dir="/gdrive/My Drive/transfer_learning/logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
#EarlyStopping Callback
#earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=3, verbose=1)
#Creating a model
model = Model(inputs=input_layer,outputs=Out)
#optimizer=tf.keras.optimizers.Adam(lr=0.01)
model.compile(optimizer=tf.keras.optimizers.Adamax(
    learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Adamax'),loss=tf.keras.losses.CategoricalCrossentropy(),metrics=['accuracy'])
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

```
# Loading data into the /content folder
!pip install pyunpack
!pip install patool
from pyunpack import Archive
Archive('/gdrive/My Drive/transfer_learning/data_final.rar').extractall('/content')
```

```
    Collecting pyunpack
      Downloading https://files.pythonhosted.org/packages/33/fd/4b64817a1d82df78553ceb1bfc5a2d7ac162da8667be586430fab9db5deb/pyunpack-0.2.1-py2.py3
    Collecting easyprocess
      Downloading https://files.pythonhosted.org/packages/48/3c/75573613641c90c6d094059ac28adb748560d99bd27ee6f80cce398f404e/EasyProcess-0.3-py2.py
    Collecting entrypoint2
      Downloading https://files.pythonhosted.org/packages/46/0a/6156f1bc14a44094cff75bb6ecefe1f8e8a12cfff66379ba3d52d0916c49/entrypoint2-0.2.1-py2.
    Collecting argparse
      Downloading https://files.pythonhosted.org/packages/f2/94/3af39d34be01a24a6e65433d19e107099374224905f1e0cc6bbe1fd22a2f/argparse-1.4.0-py2.py3
    Installing collected packages: easyprocess, argparse, entrypoint2, pyunpack
```

```
!rm -rf ./logs/
```

```
      [argparse]
```

```
data['label'] = data['label'].astype(str)
data['path'] = '/content/data_final/' + data['path'].astype('str')
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        rotation_range=20,
        fill_mode = 'nearest',
        horizontal_flip=True,
        validation_split = 0.2)
```

```
#creating a generator
generator_train = train_datagen.flow_from_dataframe(data,  x_col='path', y_col="label", class_mode="categorical",validate_filenames = False ,subet='t
generator_test = train_datagen.flow_from_dataframe(data,  x_col='path', y_col="label", class_mode="categorical", validate_filenames =False,subset='va
```

```
Found 48000 non-validated image filenames belonging to 16 classes.
Found 9600 non-validated image filenames belonging to 16 classes.
```

Increasing gpu speed

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

```
%tensorflow_version 2.x
import tensorflow as tf
import timeit

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  print(
      '\n\nThis error most likely means that this notebook is not '
      'configured to use a GPU.  Change this in Notebook Settings via the '
      'command palette (cmd/ctrl-shift-P) or the Edit menu.\n\n')
  raise SystemError('GPU device not found')

def cpu():
  with tf.device('/cpu:0'):
    random_image_cpu = tf.random.normal((100, 100, 100, 3))
    net_cpu = tf.keras.layers.Conv2D(32, 7)(random_image_cpu)
    return tf.math.reduce_sum(net_cpu)

def gpu():
  with tf.device('/device:GPU:0'):
    random_image_gpu = tf.random.normal((100, 100, 100, 3))
    net_gpu = tf.keras.layers.Conv2D(32, 7)(random_image_gpu)
    return tf.math.reduce_sum(net_gpu)

# We run each op once to warm up; see: https://stackoverflow.com/a/45067900
cpu()
gpu()

# Run the op several times.
print('Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images '
      '(batch x height x width x channel). Sum of ten runs.')
print('CPU (s):')
cpu_time = timeit.timeit('cpu()', number=10, setup="from __main__ import cpu")
```

```
print(cpu_time)
print('GPU (s):')
gpu_time = timeit.timeit('gpu()', number=10, setup="from __main__ import gpu")
print(gpu_time)
print('GPU speedup over CPU: {}x'.format(int(cpu_time/gpu_time)))
```

```
Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x height x width x channel). Sum of ten runs.
CPU (s):
3.126106019999952
GPU (s):
0.04629843799966693
GPU speedup over CPU: 67x
```

```
model.fit_generator(generator_train,steps_per_epoch=1200,validation_data=generator_test,
                    validation_steps=375,epochs=15,workers=16,verbose=1,callbacks=[tensorboard_callback,earlystop,reduce_lr])
```

```
WARNING:tensorflow:From <ipython-input-17-f8dd82481f96>:2: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/15
   1/1200 [..............................] - ETA: 0s - loss: 2.9248 - accuracy: 0.1250WARNING:tensorflow:From /usr/local/lib/python3.6/dist-pac
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
1200/1200 [==============================] - 572s 476ms/step - loss: 2.1982 - accuracy: 0.2817 - val_loss: 1.8762 - val_accuracy: 0.4037
Epoch 2/15
1200/1200 [==============================] - 562s 468ms/step - loss: 1.7998 - accuracy: 0.4211 - val_loss: 1.6798 - val_accuracy: 0.4563
Epoch 3/15
1200/1200 [==============================] - 572s 476ms/step - loss: 1.6725 - accuracy: 0.4632 - val_loss: 1.6130 - val_accuracy: 0.4837
Epoch 4/15
1200/1200 [==============================] - 569s 474ms/step - loss: 1.6032 - accuracy: 0.4915 - val_loss: 1.5689 - val_accuracy: 0.5002
Epoch 5/15
1200/1200 [==============================] - 566s 472ms/step - loss: 1.5659 - accuracy: 0.5047 - val_loss: 1.5471 - val_accuracy: 0.5144
Epoch 6/15
1200/1200 [==============================] - 568s 473ms/step - loss: 1.5337 - accuracy: 0.5187 - val_loss: 1.5392 - val_accuracy: 0.5156
Epoch 7/15
1200/1200 [==============================] - 569s 474ms/step - loss: 1.5070 - accuracy: 0.5267 - val_loss: 1.4946 - val_accuracy: 0.5309
Epoch 8/15
1200/1200 [==============================] - 569s 474ms/step - loss: 1.4964 - accuracy: 0.5292 - val_loss: 1.4732 - val_accuracy: 0.5334
Epoch 9/15
1200/1200 [==============================] - 570s 475ms/step - loss: 1.4891 - accuracy: 0.5292 - val_loss: 1.4278 - val_accuracy: 0.5539
Epoch 10/15
1200/1200 [==============================] - 574s 479ms/step - loss: 1.4646 - accuracy: 0.5412 - val_loss: 1.4202 - val_accuracy: 0.5559
Epoch 11/15
1200/1200 [==============================] - 550s 458ms/step - loss: 1.4587 - accuracy: 0.5453 - val_loss: 1.4310 - val_accuracy: 0.5493
Epoch 12/15
1200/1200 [==============================] - 558s 465ms/step - loss: 1.4386 - accuracy: 0.5501 - val_loss: 1.4299 - val_accuracy: 0.5533
Epoch 00012: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f54f3afe2b0>
```

```
%load_ext tensorboard
```

```
/bin/bash: line 0: kill: (465) - No such process
```

```
%tensorboard --logdir '/gdrive/My Drive/transfer_learning/logs/fit/'
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Show data download links

Ignore outliers in chart scaling

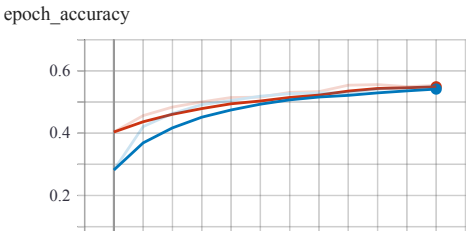
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

Filter tags (regular expressions supported)

epoch_accuracy




```
tf.keras.models.save_model(
    model, '/gdrive/My Drive/transfer_learning/', overwrite=True, include_optimizer=True)
```

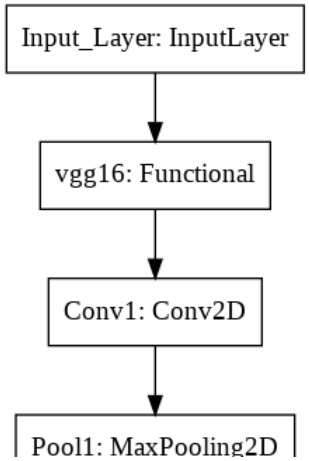
```
INFO:tensorflow:Assets written to: /gdrive/My Drive/transfer_learning/assets
```

```
model = tf.keras.models.load_model('/gdrive/My Drive/transfer_learning/model')
```

```
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (InputLayer)     [(None, 156, 256, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, None, None, 512)   14714688
_________________________________________________________________
Conv1 (Conv2D)               (None, 2, 6, 32)          147488
_________________________________________________________________
Pool1 (MaxPooling2D)         (None, 1, 3, 32)          0
_________________________________________________________________
Flatten (Flatten)            (None, 96)                0
_________________________________________________________________
FC1 (Dense)                  (None, 30)                2910
_________________________________________________________________
FC2 (Dense)                  (None, 15)                465
_________________________________________________________________
Output (Dense)               (None, 16)                256
=================================================================
Total params: 14,865,807
Trainable params: 14,865,807
Non-trainable params: 0
_________________________________________________________________
```

```
tf.keras.utils.plot_model(
    model, to_file='/content/model.png', show_shapes=False, show_layer_names=True
)
```



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
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 cl
4. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

```
generator_train = train_datagen.flow_from_dataframe(data,  x_col='path', y_col="label", class_mode="categorical",validate_filenames = False ,subet='t
generator_test = train_datagen.flow_from_dataframe(data,  x_col='path', y_col="label", class_mode="categorical", validate_filenames =False,subset='va
```

```
Found 48000 non-validated image filenames belonging to 16 classes.
Found 9600 non-validated image filenames belonging to 16 classes.
```

```
import os
import datetime
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.95,patience=1,mode = 'min')

os.environ['PYTHONHASHSEED'] = '0'

##https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development
## Have to clear the session. If you are not clearing, Graph will create again and again and graph size will increses.
## Varibles will also set to some value from before session
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)
vgg16_layer = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_shape=(228,228,3))
for layer in vgg16_layer.layers:
  layer.trainable = False
#Input layer

#VGG16

vgg16_layer_output = vgg16_layer.output
#CONV2
Conv2 = Conv2D(filters = 100 , kernel_size = (7,7),strides=(1,1),padding='valid',activation='relu',kernel_initializer=tf.keras.initializers.he_unifor
#conv3
Conv3 = Conv2D(filters = 50, kernel_size = (1,1),strides=(1,1),padding='valid',activation='relu',kernel_initializer=tf.keras.initializers.he_uniform


#output layer
Out = Dense(units=16,activation='softmax',kernel_initializer=tf.keras.initializers.he_uniform(seed=32),name='Output')(Conv3)
flatten = Flatten(data_format='channels_last',name='Flatten')(Out)
#TensorBoard Callback
log_dir="/gdrive/My Drive/transfer_learning/logs1/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
#EarlyStopping Callback
earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=3, verbose=1)
#Creating a model
model1 = Model(inputs=vgg16_layer.input,outputs=flatten)
#optimizer=tf.keras.optimizers.Adam(lr=0.01)
model1.compile(optimizer=tf.keras.optimizers.Adamax(
    learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Adamax'),loss=tf.keras.losses.CategoricalCrossentropy(),metrics=['accuracy'])
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

```
model1.fit_generator(generator_train,steps_per_epoch=1200,validation_data=generator_test,
                     validation_steps=375,epochs=15,workers = 16,verbose=1,callbacks=[tensorboard_callback,earlystop,reduce_lr])
```

```
Epoch 1/15
   1/1200 [..............................] - ETA: 0s - loss: 3.2024 - accuracy: 0.0312WARNING:tensorflow:From /usr/local/lib/python3.6/dist-pac
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
1200/1200 [==============================] - 717s 598ms/step - loss: 14.1067 - accuracy: 0.1211 - val_loss: 13.5881 - val_accuracy: 0.1424
Epoch 2/15
1200/1200 [==============================] - 714s 595ms/step - loss: 3.7208 - accuracy: 0.3760 - val_loss: 1.6536 - val_accuracy: 0.4854
Epoch 3/15
1200/1200 [==============================] - 709s 591ms/step - loss: 1.5242 - accuracy: 0.5226 - val_loss: 1.4325 - val_accuracy: 0.5481
Epoch 4/15
1200/1200 [==============================] - 700s 583ms/step - loss: 1.4255 - accuracy: 0.5586 - val_loss: 1.3705 - val_accuracy: 0.5771
Epoch 5/15
1200/1200 [==============================] - 693s 578ms/step - loss: 1.3576 - accuracy: 0.5750 - val_loss: 1.3063 - val_accuracy: 0.5957
Epoch 6/15
1200/1200 [==============================] - 729s 607ms/step - loss: 1.3224 - accuracy: 0.5918 - val_loss: 1.2940 - val_accuracy: 0.5943
Epoch 7/15
1200/1200 [==============================] - 679s 565ms/step - loss: 1.3041 - accuracy: 0.5984 - val_loss: 1.2441 - val_accuracy: 0.6212
Epoch 8/15
1200/1200 [==============================] - 702s 585ms/step - loss: 1.2624 - accuracy: 0.6117 - val_loss: 1.2577 - val_accuracy: 0.6177
Epoch 9/15
1200/1200 [==============================] - 706s 589ms/step - loss: 1.2379 - accuracy: 0.6170 - val_loss: 1.1831 - val_accuracy: 0.6363
Epoch 10/15
1200/1200 [==============================] - 694s 578ms/step - loss: 1.2189 - accuracy: 0.6240 - val_loss: 1.1939 - val_accuracy: 0.6338
Epoch 11/15
1200/1200 [==============================] - 675s 562ms/step - loss: 1.1985 - accuracy: 0.6315 - val_loss: 1.2437 - val_accuracy: 0.6190
Epoch 12/15
1200/1200 [==============================] - 687s 572ms/step - loss: 1.1827 - accuracy: 0.6360 - val_loss: 1.1490 - val_accuracy: 0.6491
Epoch 13/15
1168/1200 [============================>.] - ETA: 13s - loss: 1.1659 - accuracy: 0.6444
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-35-1db3890b2d09> in <module>()
      1 model1.fit_generator(generator_train,steps_per_epoch=1200,validation_data=generator_test,
----> 2                      validation_steps=375,epochs=15,workers = 16,verbose=1,callbacks=[tensorboard_callback,earlystop,reduce_lr])

10 frames

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     58     ctx.ensure_initialized()
     59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:
     62     if name is not None:

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
%tensorboard --logdir '/gdrive/My Drive/transfer_learning/logs1/fit/'
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Show data download links

Ignore outliers in chart scaling

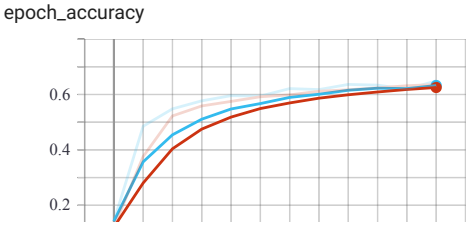
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

Filter tags (regular expressions supported)

epoch_accuracy




```
tf.keras.models.save_model(
    model1, '/gdrive/My Drive/', overwrite=True, include_optimizer=True)
```

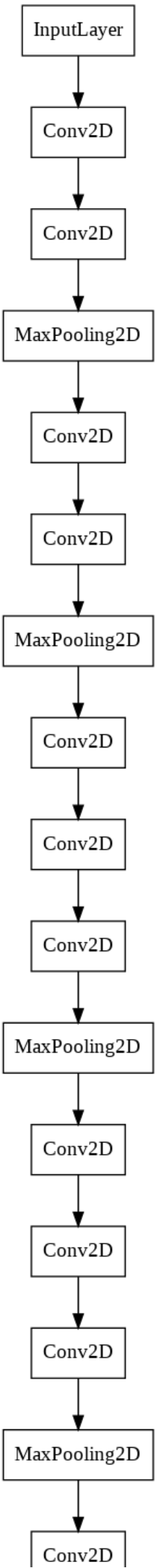
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from t
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates (from tensorf
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: /gdrive/My Drive/assets
```

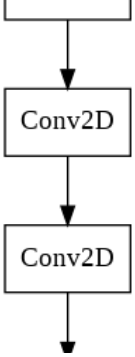
20200814-153935/train

```
model1.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (InputLayer)     [(None, 156, 256, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, None, None, 512)   14714688
_________________________________________________________________
Conv2 (Conv2D)               (None, 4, 8, 512)         262656
_________________________________________________________________
Conv3 (Conv2D)               (None, 4, 8, 206)         105678
_________________________________________________________________
Output (Dense)               (None, 4, 8, 16)          3312
=================================================================
Total params: 15,086,334
Trainable params: 371,646
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
tf.keras.utils.plot_model(
    model1, to_file='/content/model1.png', show_shapes=False, show_layer_names=False,expand_nested = False, rankdir ='TB'
)
```


InputLayer
Conv2D
Conv2D
MaxPooling2D
Conv2D
Conv2D
MaxPooling2D
Conv2D
Conv2D
Conv2D
MaxPooling2D
Conv2D
Conv2D
Conv2D
MaxPooling2D
Conv2D



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 tra

```
import os
import datetime
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.95,patience=1,mode = 'min')

os.environ['PYTHONHASHSEED'] = '0'

##https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development
## Have to clear the session. If you are not clearing, Graph will create again and again and graph size will increses.
## Varibles will also set to some value from before session
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)
vgg16_layer = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_shape=(228,228,3))
for layer in vgg16_layer.layers[0:-6]:
  layer.trainable = False
#for layer in vgg16_layer.layers[-6]:
 # layer.trainable = True
#Input layer

#VGG16

vgg16_layer_output = vgg16_layer.output
#CONV2
Conv2 = Conv2D(filters = 100 , kernel_size = (7,7),strides=(1,1),padding='valid',activation='relu',kernel_initializer=tf.keras.initializers.he_unifor
#conv3
Conv3 = Conv2D(filters = 50, kernel_size = (1,1),strides=(1,1),padding='valid',activation='relu',kernel_initializer=tf.keras.initializers.he_uniform


#output layer
Out = Dense(units=16,activation='softmax',kernel_initializer=tf.keras.initializers.he_uniform(seed=32),name='Output')(Conv3)
flatten = Flatten(data_format='channels_last',name='Flatten')(Out)
#TensorBoard Callback
log_dir="/gdrive/My Drive/transfer_learning/logs1/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_grads=True)
#EarlyStopping Callback
earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=3, verbose=1)
#Creating a model
model2 = Model(inputs=vgg16_layer.input,outputs=flatten)
#optimizer=tf.keras.optimizers.Adam(lr=0.01)
model2.compile(optimizer=tf.keras.optimizers.Adamax(
    learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Adamax'),loss=tf.keras.losses.CategoricalCrossentropy(),metrics=['accuracy'])
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

```
model2.fit_generator(generator_train,steps_per_epoch=1200,validation_data=generator_test,
                     validation_steps=375,epochs=15,workers = 16,verbose=1,callbacks=[tensorboard_callback,earlystop,reduce_lr])
```

```
WARNING:tensorflow:From <ipython-input-19-75620b802b8d>:2: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/15
   1/1200 [..............................] - ETA: 0s - loss: 3.1850 - accuracy: 0.0625WARNING:tensorflow:From /usr/local/lib/python3.6/dist-pac
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
   2/1200 [..............................] - ETA: 3:39 - loss: 9.3997 - accuracy: 0.0469WARNING:tensorflow:Callbacks method `on_train_batch_end
1200/1200 [==============================] - 629s 524ms/step - loss: 15.1130 - accuracy: 0.0617 - val_loss: 15.1456 - val_accuracy: 0.0603
Epoch 2/15
1200/1200 [==============================] - 636s 530ms/step - loss: 15.1095 - accuracy: 0.0626 - val_loss: 15.1322 - val_accuracy: 0.0612
Epoch 3/15
```

```
tf.keras.models.save_model(
    model2, '/gdrive/My Drive/', overwrite=True, include_optimizer=True)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from t
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Layer.updates (from tensorf
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: /gdrive/My Drive/assets
```

```
%tensorboard --logdir '/gdrive/My Drive/transfer_learning/logs1/fit/'
```

```
Reusing TensorBoard on port 6006 (pid 1143), started 0:03:03 ago. (Use '!kill 1143' to kill it.)
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

- 20200814-152934/train
- 20200814-153935/train
- 20200814-164841/train
- 20200814-165236/train
- 20200814-165404/train
- 20200814-165404/validation

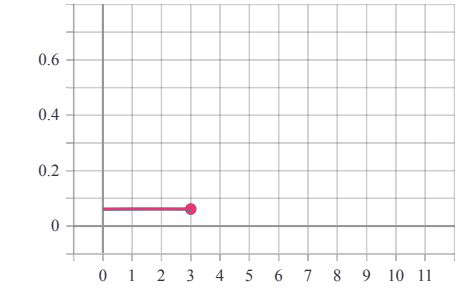
TOGGLE ALL RUNS

/gdrive/My Drive/transfer_learning/logs1/fit/

Filter tags (regular expressions supported)

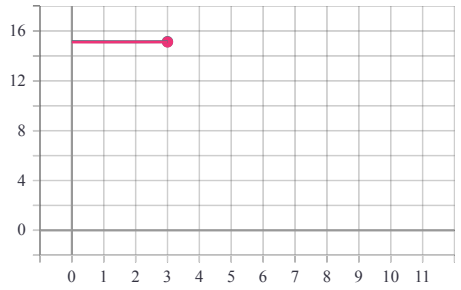
epoch_accuracy

epoch_accuracy

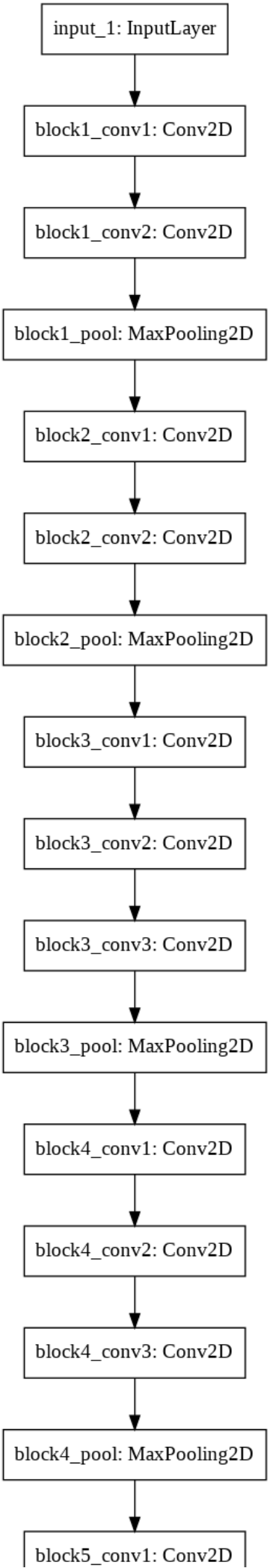


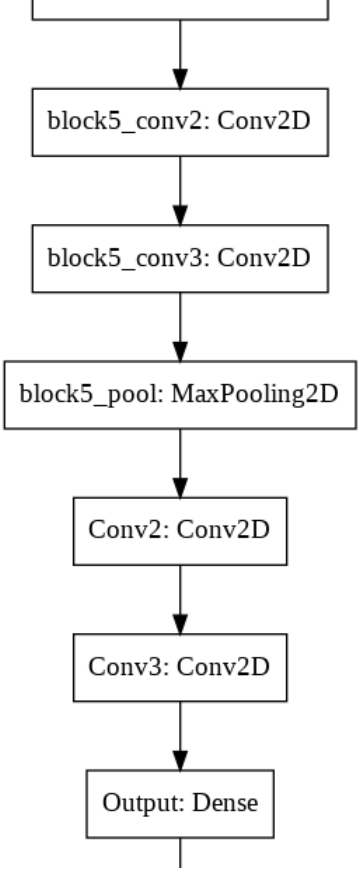
epoch_loss

epoch_loss




```
tf.keras.utils.plot_model(
    model2, to_file='/content/model2.png', show_shapes=False, show_layer_names=True
)
```


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



Conclusion:

Model 1:

```
Accuracy = 55%
Validation Accuracy = 55.3%
Epochs = 12
```

Model 2:

```
Accuracy = 64%
Validation Accuracy = 64.91%
Epochs = 13
```

Model 3:

```
Accuracy = 6.21%
Validation Accuracy = 6%
Epochs = 4 (early stopping)
```