Transfer Learning Assignment

```
!gdown --id 1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
```

```
Access denied with the following error:

        Cannot retrieve the public link of the file. You may need to change
        the permission to 'Anyone with the link', or have had many accesses.

You may still be able to access the file from the browser:

        https://drive.google.com/uc?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
```

```
!wget --header="Host: doc-00-c4-docs.googleusercontent.com" --header="User-Agent: Mozilla/
```

```
--2022-02-26 06:15:18--  https://doc-00-c4-docs.googleusercontent.com/docs/securesc/e
Resolving doc-00-c4-docs.googleusercontent.com (doc-00-c4-docs.googleusercontent.com)
Connecting to doc-00-c4-docs.googleusercontent.com (doc-00-c4-docs.googleusercontent
HTTP request sent, awaiting response... 200 OK
Length: 4660541790 (4.3G) [application/rar]
Saving to: ‘rvl-cdip.rar’

rvl-cdip.rar        100%[===================>]   4.34G   194MB/s    in 24s

2022-02-26 06:15:42 (182 MB/s) - ‘rvl-cdip.rar’ saved [4660541790/4660541790]
```

```
#import all the required libraries
import tensorflow as tf
import os
import numpy as np
import pandas as pd
from tqdm import tqdm
```

```
#unrar the file
#get_ipython().system_raw("unrar x rvl-cdip-002.rar")
with tf.device('/device:GPU:0'):
  !unrar x /content/rvl-cdip.rar
```

```
Creating    data_final/imagesz/z/z/p/zzp19e00                          OK
Extracting  data_final/imagesz/z/z/p/zzp19e00/2503001544.tif           OK
Creating    data_final/imagesz/z/z/p/zzp22c00                          OK
Extracting  data_final/imagesz/z/z/p/zzp22c00/2069733071.tif           OK
Creating    data_final/imagesz/z/z/p/zzp57d00                          OK
Extracting  data_final/imagesz/z/z/p/zzp57d00/2046288261.tif           OK
Creating    data_final/imagesz/z/z/p/zzp82f00                          OK
Extracting  data_final/imagesz/z/z/p/zzp82f00/tob07124.29.tif          OK
Creating    data_final/imagesz/z/z/q                                   OK
Creating    data_final/imagesz/z/z/q/zzq96c00                          OK
Extracting  data_final/imagesz/z/z/q/zzq96c00/CIRCONTRACTS001527_1.tif  OK
```

```
Extracting  data_final/imagesz/z/z/q/zzq96c00/CTRCONTRACTS001537-1.tif  OK
Creating    data_final/imagesz/z/z/r                                    OK
Creating    data_final/imagesz/z/z/r/zzr32c00                           OK
Extracting  data_final/imagesz/z/z/r/zzr32c00/2071518198_8210.tif       OK
Creating    data_final/imagesz/z/z/r/zzr32d00                           OK
Extracting  data_final/imagesz/z/z/r/zzr32d00/2028593544.tif            OK
Creating    data_final/imagesz/z/z/r/zzr90e00                           OK
Extracting  data_final/imagesz/z/z/r/zzr90e00/91515365.tif              OK
Creating    data_final/imagesz/z/z/r/zzr93e00                           OK
Extracting  data_final/imagesz/z/z/r/zzr93e00/2040794406.tif            OK
Creating    data_final/imagesz/z/z/r/zzr99c00                           OK
Extracting  data_final/imagesz/z/z/r/zzr99c00/40047516-7517.tif         OK
Creating    data_final/imagesz/z/z/s                                    OK
Creating    data_final/imagesz/z/z/s/zzs37d00                           OK
Extracting  data_final/imagesz/z/z/s/zzs37d00/2070909855.tif            OK
Creating    data_final/imagesz/z/z/s/zzs53c00                           OK
Extracting  data_final/imagesz/z/z/s/zzs53c00/98716908_6910.tif         OK
Creating    data_final/imagesz/z/z/s/zzs93f00                           OK
Extracting  data_final/imagesz/z/z/s/zzs93f00/0000341205.tif            OK
Creating    data_final/imagesz/z/z/t                                    OK
Creating    data_final/imagesz/z/z/t/zzt18c00                           OK
Extracting  data_final/imagesz/z/z/t/zzt18c00/505576130+-6130.tif       OK
Creating    data_final/imagesz/z/z/t/zzt40c00                           OK
Extracting  data_final/imagesz/z/z/t/zzt40c00/ti16792723.tif            OK
Creating    data_final/imagesz/z/z/t/zzt52c00                           OK
Extracting  data_final/imagesz/z/z/t/zzt52c00/2075743296_3297.tif       OK
Creating    data_final/imagesz/z/z/t/zzt65d00                           OK
Extracting  data_final/imagesz/z/z/t/zzt65d00/504473130.tif             OK
Creating    data_final/imagesz/z/z/t/zzt92e00                           OK
Extracting  data_final/imagesz/z/z/t/zzt92e00/2046718732.tif            OK
Creating    data_final/imagesz/z/z/u                                    OK
Creating    data_final/imagesz/z/z/u/zzu44a00                           OK
Extracting  data_final/imagesz/z/z/u/zzu44a00/94260985_1005.tif         OK
Creating    data_final/imagesz/z/z/u/zzu86d00                           OK
Extracting  data_final/imagesz/z/z/u/zzu86d00/tcal0288729.tif           OK
Creating    data_final/imagesz/z/z/v                                    OK
Creating    data_final/imagesz/z/z/v/zzv00d00                           OK
Extracting  data_final/imagesz/z/z/v/zzv00d00/50357861-7862.tif         OK
Creating    data_final/imagesz/z/z/v/zzv02e00                           OK
Extracting  data_final/imagesz/z/z/v/zzv02e00/2028719331.tif            OK
Creating    data_final/imagesz/z/z/w                                    OK
Creating    data_final/imagesz/z/z/w/zzw60e00                           OK
Extracting  data_final/imagesz/z/z/w/zzw60e00/93219048.tif              OK
Creating    data_final/imagesz/z/z/w/zzw97d00                           OK
Extracting  data_final/imagesz/z/z/w/zzw97d00/2072300905.tif            OK
Creating    data_final/imagesz/z/z/x                                    OK
Creating    data_final/imagesz/z/z/x/zzx33e00                           OK
Extracting  data_final/imagesz/z/z/x/zzx33e00/2044073803.tif            OK
Creating    data_final/imagesz/z/z/x/zzx98e00                           OK
```

```
df=pd.read_csv('/content/labels_final.csv',dtype=str)
```

```
df.describe()
```

	path	label
count	48000	48000

```
df.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

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size=0.33, random_state = 42)

train.shape, test.shape
```

```
((32160, 2), (15840, 2))
```

```
train.values[:5]
```

```
array([['imagesw/w/v/a/wva22f00/24010514.tif', '10'],
       ['imagesc/c/y/x/cyx25e00/2025833346_2025833356.tif', '12'],
       ['imagesk/k/p/g/kpg75f00/0060069726.tif', '13'],
       ['imagesb/b/a/v/bav63c00/2065350767.tif', '12'],
       ['imagesf/f/b/l/fbl10c00/2085279052.tif', '2']], dtype=object)
```

```
df.groupby(by=["label"]).nunique()
```

label	path
0	3016
1	2994
10	3002
11	2992
12	3006

```
if not os.path.exists(r'train_dataset'):
    os.makedirs(r'train_dataset')
if not os.path.exists(r'test_dataset'):
    os.makedirs(r'test_dataset')
```

```
label_lst = pd.unique(df['label'])
for item in label_lst:
    if not os.path.exists(os.path.join('train_dataset',str(item))):
        os.makedirs(os.path.join('train_dataset',str(item)))
    if not os.path.exists(os.path.join('test_dataset',str(item))):
        os.makedirs(os.path.join('test_dataset',str(item)))
```

```
import shutil
with tf.device('/device:GPU:0'):
  for item in tqdm(train.values):
      filename = str.split(item[0],'/')[-1]
      for folder in os.listdir('train_dataset'):
          if item[1] == folder:
              shutil.copyfile(os.path.join('data_final',str(item[0])), os.path.join('train
```

```
100%|██████████| 32160/32160 [01:22<00:00, 389.51it/s]
```

```
with tf.device('/device:GPU:0'):
  for item in tqdm(test.values):
      filename = str.split(item[0],'/')[-1]
      for folder in os.listdir('test_dataset'):
          if item[1] == folder:
              shutil.copyfile(os.path.join('data_final',str(item[0])), os.path.join('test_
```

```
100%|██████████| 15840/15840 [00:45<00:00, 345.27it/s]
```

```
dir_path = 'train_dataset'
```

```
for i in os.listdir(dir_path):
    print("No of Images in ",i," category is ",len(os.listdir(os.path.join(dir_path,i))))
```

```
 No of Images in  2  category is  1969
 No of Images in  6  category is  1954
```

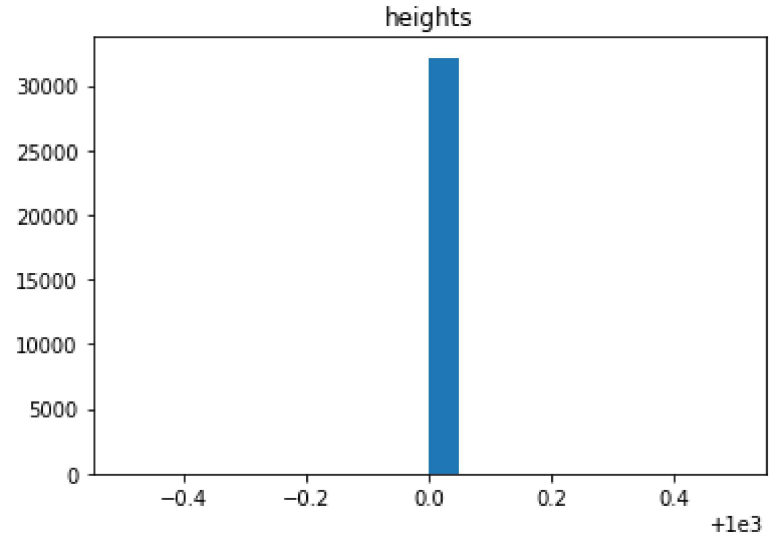
```
No of Images in  4  category is  2009
No of Images in  8  category is  1935
No of Images in  10  category is  2034
No of Images in  15  category is  2009
No of Images in  7  category is  1993
No of Images in  13  category is  2017
No of Images in  11  category is  2045
No of Images in  9  category is  2026
No of Images in  0  category is  2054
No of Images in  12  category is  2037
No of Images in  3  category is  2040
No of Images in  5  category is  1990
No of Images in  14  category is  2020
No of Images in  1  category is  2026
```

```
#import openCV
import cv2

##Getting size of images
list_of_heights = []
list_of_widths = []
for i in tqdm(os.listdir(dir_path)):
    for image in os.listdir(os.path.join(dir_path,i)):
        img = cv2.imread(os.path.join(os.path.join(dir_path,i),image), cv2.IMREAD_UNCHANGE
        # get dimensions of image
        shape = img.shape
        list_of_heights.append(shape[0])
        list_of_widths.append(shape[1])
```

```
100%|██████████| 16/16 [02:24<00:00,  9.05s/it]
```

```
##plotting
import matplotlib.pyplot as plt
plt.hist(list_of_heights,bins=20)
#plt.xlim(0,900)
plt.title('heights')
```

```
Text(0.5, 1.0, 'heights')
```

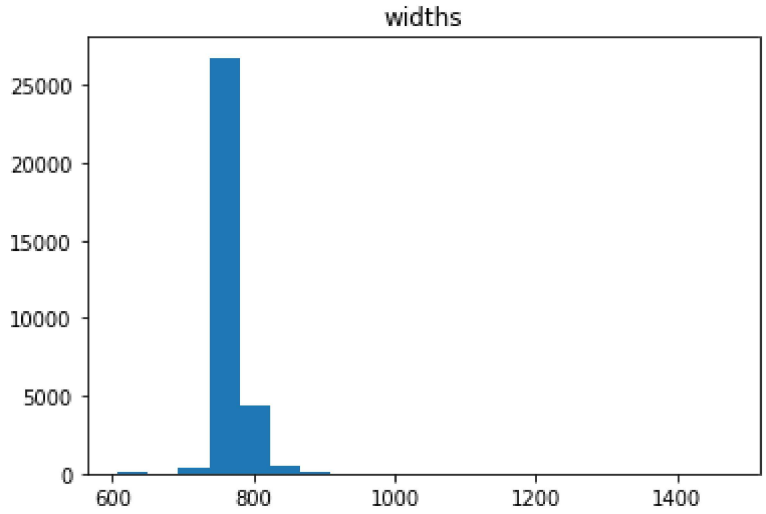



```
np.median(list_of_heights), np.mean(list_of_heights)
```

```
(1000.0, 1000.0)
```

```
plt.hist(list_of_widths,bins=20)
plt.title('widths')
#plt.xlim(0,900)
```

```
Text(0.5, 1.0, 'widths')
```




```
np.median(list_of_widths), np.mean(list_of_widths)
```

```
(762.0, 766.3176503513901)
```

```
###Image data Generator class
ImageFlow = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
class_subset = sorted(os.listdir(dir_path))
print(class_subset)
##We are fitting the data to Image data generator.
trainImageGenerator = ImageFlow.flow_from_directory('train_dataset',target_size=(224,224),
testImageGenerator = ImageFlow.flow_from_directory('test_dataset',target_size=(224,224),cl
```

```
['0', '1', '10', '11', '12', '13', '14', '15', '2', '3', '4', '5', '6', '7', '8', '9
Found 32158 images belonging to 16 classes.
Found 15840 images belonging to 16 classes.
```

```
#importing tensorflow
from tensorflow.keras.layers import Dense,Input,Conv2D,Conv1D,MaxPool2D,Activation,Dropout
from tensorflow.keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import random as rn
```

```
#Get back the convolutional part of a VGG network trained on ImageNet
model_vgg16_conv = VGG16(weights='imagenet', input_shape = (224,224,3), include_top=False)
model_vgg16_conv.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vg
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
# Make vgg16 model layers as non trainable
for layer in model_vgg16_conv.layers:
    layer.trainable = False
```

```
with tf.device('/device:GPU:0'):
```

```
  conv1 = Conv2D(256,kernel_size=(3,3),activation='relu')
  maxpool1 = MaxPool2D(5,5)
  flatten = Flatten()
  fc1 = Dense(128,activation=tf.keras.layers.LeakyReLU(alpha=0.01))
  fc2 = Dense(64,activation=tf.keras.layers.LeakyReLU(alpha=0.01))
  output = Dense(16,activation='softmax')
```

```
from tensorflow.keras import models
model = models.Sequential([
    model_vgg16_conv,
    conv1,
    maxpool1,
    flatten,
    fc1,
    fc2,
    output
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 conv2d (Conv2D)             (None, 5, 5, 256)         1179904

 max_pooling2d (MaxPooling2D  (None, 1, 1, 256)        0
 )

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 16)                1040

=================================================================
Total params: 15,936,784
Trainable params: 1,222,096
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
# there are other ways of doing this: https://www.dlology.com/blog/quick-guide-to-run-tens
%load_ext tensorboard
```

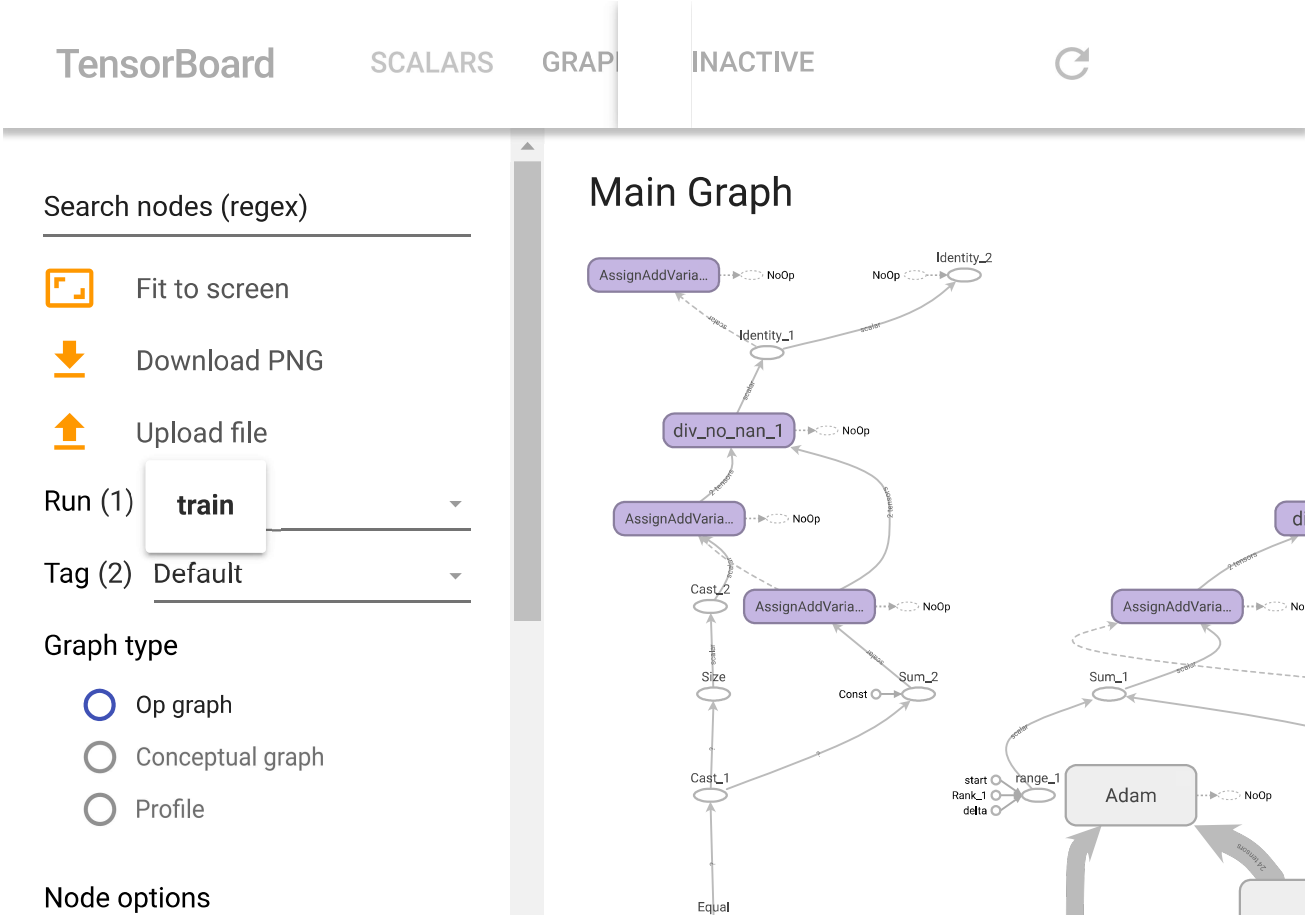
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
from tensorflow.keras.callbacks import ModelCheckpoint
```

```
from tensorflow.keras.callbacks import EarlyStopping

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])

filepath="./best_model-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_b

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.02, patience=1, verbose=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./logs")

with tf.device('/device:GPU:0'):
  model.fit(trainImageGenerator,steps_per_epoch=2010,epochs=100,validation_data=testImageG
```

```
Epoch 1/100
2010/2010 [==============================] - ETA: 0s - loss: 1.4165 - accuracy: 0.557
Epoch 1: val_accuracy improved from -inf to 0.62134, saving model to ./best_model-01-
2010/2010 [==============================] - 483s 234ms/step - loss: 1.4165 - accurac
Epoch 2/100
2010/2010 [==============================] - ETA: 0s - loss: 1.0091 - accuracy: 0.691
Epoch 2: val_accuracy improved from 0.62134 to 0.69324, saving model to ./best_model-
2010/2010 [==============================] - 456s 227ms/step - loss: 1.0091 - accurac
Epoch 3/100
2010/2010 [==============================] - ETA: 0s - loss: 0.8513 - accuracy: 0.737
Epoch 3: val_accuracy improved from 0.69324 to 0.71534, saving model to ./best_model-
2010/2010 [==============================] - 456s 227ms/step - loss: 0.8513 - accurac
Epoch 4/100
2010/2010 [==============================] - ETA: 0s - loss: 0.7541 - accuracy: 0.764
Epoch 4: val_accuracy improved from 0.71534 to 0.72090, saving model to ./best_model-
2010/2010 [==============================] - 456s 227ms/step - loss: 0.7541 - accurac
Epoch 4: early stopping
```

```
%tensorboard --logdir logs
```



Observations:

1. Model-1 achieved 71% accuracy afterwards there is no improvement in the learning accuracy of the model. So, training the model is stopped.
2. At the end of 3rd epoch model reached to 71% accuracy
3. Accuracy increased w.r.t epochs
4. Loss decreased w.r.t epochs

Model-2


```
conv_fc1 = Conv2D(4096,kernel_size=(7,7),activation='relu')
conv_fc2 = Conv2D(4096,kernel_size=(1,1),activation='relu')
maxpool = MaxPool2D(5,5)
flatten = Flatten()
output = Dense(16,activation='softmax')


model2 = models.Sequential([
    model_vgg16_conv,
    conv_fc1,
    conv_fc2,
    flatten,
    output
])
```

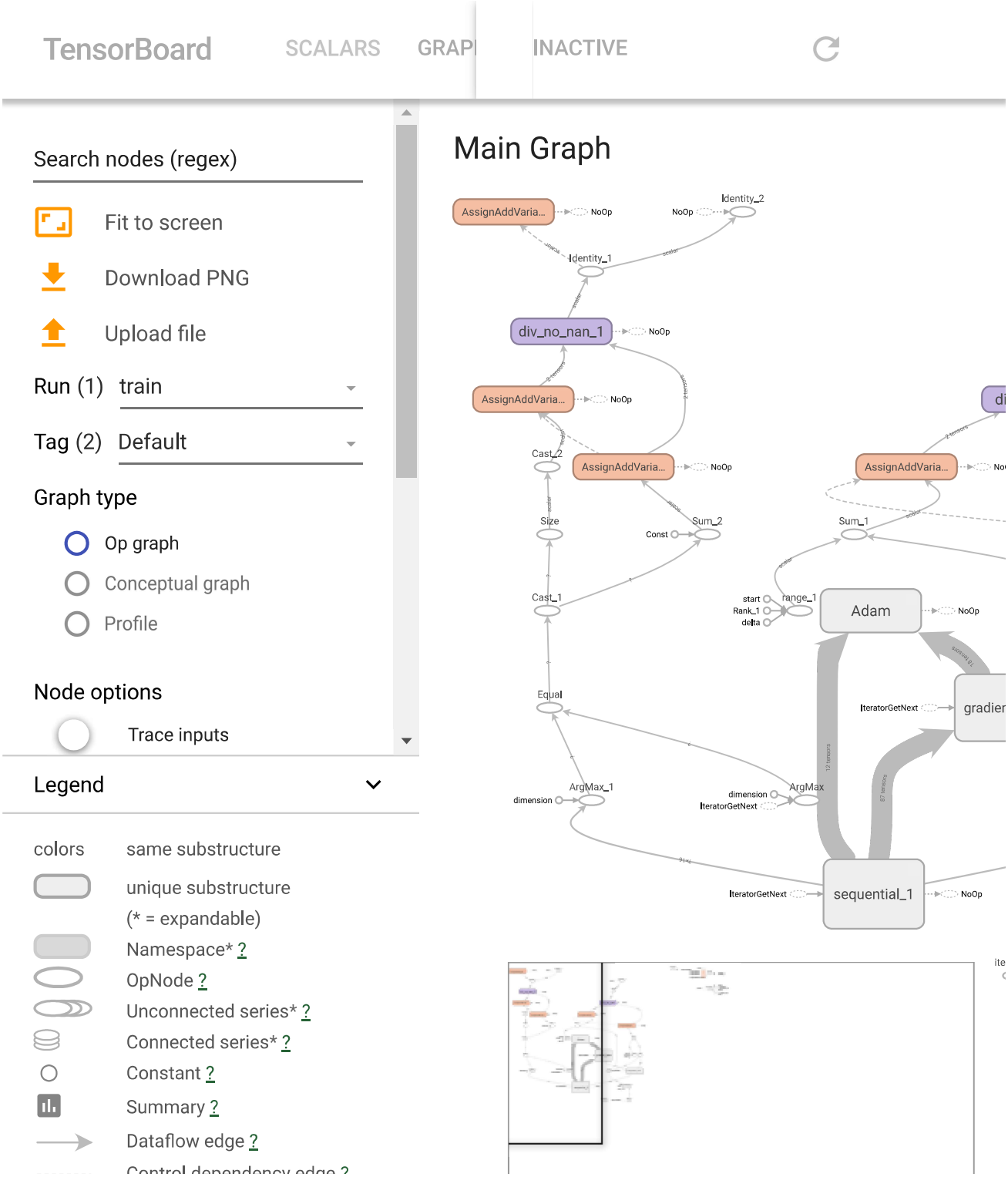
```
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 conv2d_1 (Conv2D)           (None, 1, 1, 4096)        102764544

 conv2d_2 (Conv2D)           (None, 1, 1, 4096)        16781312

 flatten_1 (Flatten)         (None, 4096)              0

 dense_3 (Dense)             (None, 16)                65552

=================================================================
Total params: 134,326,096
Trainable params: 119,611,408
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model2.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])

filepath="./model2-best_model-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_b

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.02, patience=1, verbose=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./model2_logs")

with tf.device('/device:GPU:0'):
  model2.fit(trainImageGenerator,steps_per_epoch=2010,epochs=100,validation_data=testImage
```

```
Epoch 1/100
2010/2010 [==============================] - ETA: 0s - loss: 1.4452 - accuracy: 0.573
Epoch 1: val_accuracy improved from -inf to 0.63636, saving model to ./model2-best_mo
2010/2010 [==============================] - 975s 484ms/step - loss: 1.4452 - accurac
Epoch 2/100
2010/2010 [==============================] - ETA: 0s - loss: 1.0197 - accuracy: 0.689
Epoch 2: val_accuracy improved from 0.63636 to 0.69804, saving model to ./model2-best
2010/2010 [==============================] - 972s 483ms/step - loss: 1.0197 - accurac
Epoch 3/100
2010/2010 [==============================] - ETA: 0s - loss: 0.8738 - accuracy: 0.733
Epoch 3: val_accuracy did not improve from 0.69804
2010/2010 [==============================] - 966s 481ms/step - loss: 0.8738 - accurac
Epoch 3: early stopping
```

```
%tensorboard --logdir model2_logs
```



Observations:

1. Model-2 achieved 69% accuracy afterwards there is no improvement in the learning accuracy of the model. So, training the model is stopped.
2. At the end of 2nd epoch model reached to 69% accuracy
3. Accuracy increased w.r.t epochs and after 1st epoch accuracy increased slowly
4. Loss decreased w.r.t epochs and sfter 1st epoch loss increased little

Model-3

1. Use same network as Model-2 '**INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layer**

```
model3_vgg16_conv = VGG16(weights='imagenet', input_shape = (224,224,3), include_top=False


# Make vgg16 model layers as non trainable
for layer in model3_vgg16_conv.layers[:13]:
    layer.trainable = False


for i, layer in enumerate(model3_vgg16_conv.layers):
  print(i, layer.name, layer.trainable)
```

```
0 input_3 False
1 block1_conv1 False
2 block1_conv2 False
3 block1_pool False
4 block2_conv1 False
5 block2_conv2 False
6 block2_pool False
7 block3_conv1 False
8 block3_conv2 False
9 block3_conv3 False
10 block3_pool False
11 block4_conv1 False
12 block4_conv2 False
13 block4_conv3 True
14 block4_pool True
15 block5_conv1 True
16 block5_conv2 True
17 block5_conv3 True
18 block5_pool True
```

```
model3 = models.Sequential([
    model3_vgg16_conv,
    conv_fc1,
    conv_fc2,
    flatten,
    output
])


model3.summary()
```

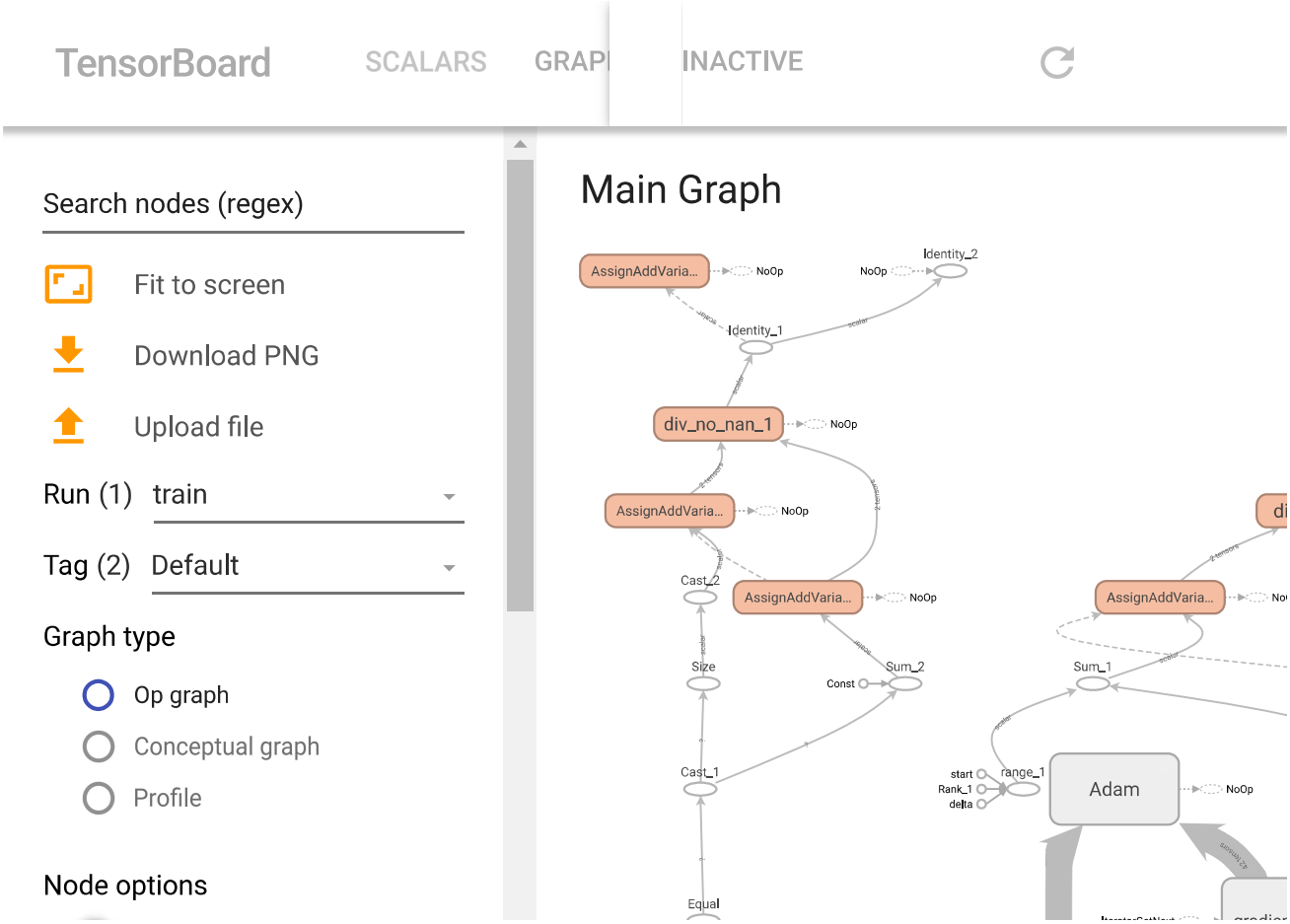
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 conv2d_1 (Conv2D)           (None, 1, 1, 4096)        102764544

 conv2d_2 (Conv2D)           (None, 1, 1, 4096)        16781312
```

```
 flatten_1 (Flatten)         (None, 4096)              0

 dense_3 (Dense)             (None, 16)                65552

=================================================================
Total params: 134,326,096
Trainable params: 129,050,640
Non-trainable params: 5,275,456
_________________________________________________________________
```

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model3.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])

filepath="./model3-best_model-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_b

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.02, patience=1, verbose=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./model3_logs")

with tf.device('/device:GPU:0'):
  model3.fit(trainImageGenerator,steps_per_epoch=2010,epochs=100,validation_data=testImage
```

```
Epoch 1/100
2010/2010 [==============================] - ETA: 0s - loss: 2.8116 - accuracy: 0.064
Epoch 1: val_accuracy improved from -inf to 0.06067, saving model to ./model3-best_mo
2010/2010 [==============================] - 1342s 667ms/step - loss: 2.8116 - accura
Epoch 2/100
2010/2010 [==============================] - ETA: 0s - loss: 2.7730 - accuracy: 0.059
Epoch 2: val_accuracy did not improve from 0.06067
2010/2010 [==============================] - 1332s 663ms/step - loss: 2.7730 - accura
Epoch 2: early stopping
```

```
%tensorboard --logdir model3_logs
```



Observations:

1. Model-3 achieved 6% accuracy afterwards there is no improvement in the learning accuracy of the model. So, training the model is stopped.
2. At the end of 1st epoch, model reached to 6% accuracy