In [ ]:

```
# references:
# https://github.com/Kaggle/kaggle-api
# https://towardsdatascience.com/downloading-datasets-into-google-drive-via-google-colab-bcb1b30b0166
# https://www.pyimagesearch.com/2018/12/24/how-to-use-keras-fit-and-fit_generator-a-hands-on-tutorial/
```

In [1]:

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

```
Mounted at /content/gdrive
```

In [2]:

```
# importing necessay libraries
import tensorflow as tf
import datetime
import os
from tensorflow.keras.layers import Dense,Activation,Conv2D,Flatten,MaxPooling2D,Dropout
from tensorflow.keras import regularizers,optimizers,initializers
from tensorflow.keras.models import Model
from keras_preprocessing.image import ImageDataGenerator
import pandas as pd
import numpy as np
```

In [4]:

```
!pip install -q kaggle
from google.colab import files
files.upload()
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```


```
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.


Saving kaggle.json to kaggle.json
```

In [5]:

```
!kaggle datasets download -d brahma0545/aaic-assignment-tl
```

```
Downloading aaic-assignment-tl.zip to /content
100% 4.33G/4.34G [01:24<00:00, 67.3MB/s]
100% 4.34G/4.34G [01:24<00:00, 55.2MB/s]
```

In [6]:

```
# list the files in current directory:
!ls -ltr
```

```
total 4549336
drwxr-xr-x 1 root root       4096 Feb  4 15:26 sample_data
drwx------ 5 root root       4096 Feb  8 04:56 gdrive
-rw-r--r-- 1 root root         62 Feb  8 05:11 kaggle.json
-rw-r--r-- 1 root root 4658501240 Feb  8 05:12 aaic-assignment-tl.zip
```

In [ ]:

```
!unzip "/content/aaic-assignment-tl.zip" -d "/content/TL"
```

In [8]:

```
# label data file
dir_path = "/content/TL/labels_final.csv"
```

In [9]:

```
# fetch labels_final.csv
train_df = pd.read_csv(dir_path)
```

In [10]:

```
train_df.head()
```

Out[10]:

| | 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 |

In [11]:

```
# replacing labels
train_df = train_df.replace({'label':
                             {0:"letter",
                             1:"form",
                             2:"email",
                             3:"handwritten",
                             4:"advertisement",
                             5:"scientifit report",
                             6:"scientific publication",
                             7:"specification",
                             8:"file folder",
                             9:"news article",
                             10:"budget",
                             11:"invoice",
                             12:"presentation",
                             13:"questionnaire",
                             14:"resume",
                             15:"memo"}})
```

In [12]:

```
# how much data for each category:
train_df['label'].value_counts()
```

Out[12]:

```
letter                  3016
questionnaire           3007
presentation            3006
resume                  3006
handwritten             3005
file folder             3003
budget                  3002
news article            3002
specification           3000
scientifit report       2999
```

```
memo                    2996
form                    2994
advertisement           2994
email                   2993
invoice                 2992
scientific publication  2985
Name: label, dtype: int64
```

Observations:

1. data is balanced

In [13]:

```
datagen = ImageDataGenerator(rotation_range=90,width_shift_range=0.25,height_shift_range=0.25,horizontal_flip=0.25,vertical_flip=0.25,
    rescale=1./255,validation_split=0.30)
```

In [14]:

```
# train data generator
train_generator = datagen.flow_from_dataframe(
    dataframe=train_df,
    directory="/content/TL/data_final/",
    x_col = "path",
    y_col = "label",
    subset = "training",
    batch_size = 96,
    seed = 39,
    shuffle = True,
    class_mode="categorical",
    target_size = (224,224)
    )
```

```
Found 33600 validated image filenames belonging to 16 classes.
```

In [15]:

```
# validation data generator
valid_generator = datagen.flow_from_dataframe(
    dataframe=train_df,
    directory="/content/TL/data_final/",
    x_col = "path",
    y_col = "label",
    subset = "validation",
    batch_size = 32,
    seed = 42,
    shuffle = True,
    class_mode="categorical",
    target_size = (224,224)
)
```

```
Found 14400 validated image filenames belonging to 16 classes.
```

In [16]:

```
%load_ext tensorboard
import tensorflow as tf
import datetime, os
```

## VGG16 Pretrained model as base model

## Model-3

In [17]:

```
tf.keras.backend.clear_session()

# loading vgg16 from keras
from keras.applications import VGG16
# load model
base_model = VGG16(include_top=False,input_shape = (224, 224, 3),weights='imagenet')
# need to train last 6 layers of VGG-16 network
for i,layer in enumerate(base_model.layers):
  if (i<13):
    layer.trainable  = False
  else :
    layer.trainable  = True

# Conv layer
conv1 = Conv2D(512,kernel_size=(7,7),padding='valid',strides=(1,1),activation='relu')(base_model.output
)

# Conv layer
conv2 = Conv2D(128,kernel_size=(1,1),padding='valid',strides=(1,1),activation='relu')(conv1)

# Flatten
flat = Flatten()(conv2)

# output layer
model3 = Dense(16,activation='softmax')(flat)
final_model3 = Model(base_model.input,model3)
final_model3.compile(optimizer= optimizers.SGD(learning_rate=0.001,momentum=0.9),
                     loss="categorical_crossentropy",
                     metrics = ['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_
tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
```

In [22]:

```
final_model3.summary()
```

```
Model: "model"
_________________________________________________________________
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
_________________________________________________________________
conv2d (Conv2D)              (None, 1, 1, 512)         12845568
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 1, 128)         65664
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 16)                2064
=================================================================
Total params: 27,627,984
Trainable params: 22,352,528
Non-trainable params: 5,275,456
_________________________________________________________________
```

In [18]:

```
!rm -rf ./logs/
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
vgghist = final_model3.fit(train_generator,
                           validation_data=valid_generator,
                           callbacks=tensorboard_callback,
                           steps_per_epoch = 350,
                           epochs = 15
                           )
```

```
Epoch 1/15
350/350 [==============================] - 753s 2s/step - loss: 2.2462 - accuracy: 0.2739 - val_loss: 1
.5777 - val_accuracy: 0.5018
Epoch 2/15
350/350 [==============================] - 734s 2s/step - loss: 1.5491 - accuracy: 0.5140 - val_loss: 1
.4659 - val_accuracy: 0.5381
Epoch 3/15
350/350 [==============================] - 713s 2s/step - loss: 1.4142 - accuracy: 0.5552 - val_loss: 1
.3634 - val_accuracy: 0.5717
Epoch 4/15
350/350 [==============================] - 687s 2s/step - loss: 1.3296 - accuracy: 0.5836 - val_loss: 1
.3135 - val_accuracy: 0.5902
Epoch 5/15
350/350 [==============================] - 679s 2s/step - loss: 1.2846 - accuracy: 0.5990 - val_loss: 1
.2661 - val_accuracy: 0.6032
Epoch 6/15
350/350 [==============================] - 668s 2s/step - loss: 1.2540 - accuracy: 0.6070 - val_loss: 1
.2783 - val_accuracy: 0.5998
Epoch 7/15
350/350 [==============================] - 669s 2s/step - loss: 1.2197 - accuracy: 0.6182 - val_loss: 1
.2207 - val_accuracy: 0.6212
Epoch 8/15
350/350 [==============================] - 668s 2s/step - loss: 1.1798 - accuracy: 0.6309 - val_loss: 1
.2176 - val_accuracy: 0.6245
Epoch 9/15
350/350 [==============================] - 673s 2s/step - loss: 1.1514 - accuracy: 0.6432 - val_loss: 1
.1881 - val_accuracy: 0.6338
Epoch 10/15
350/350 [==============================] - 672s 2s/step - loss: 1.1076 - accuracy: 0.6531 - val_loss: 1
.1573 - val_accuracy: 0.6435
Epoch 11/15
350/350 [==============================] - 672s 2s/step - loss: 1.0949 - accuracy: 0.6598 - val_loss: 1
.1255 - val_accuracy: 0.6522
Epoch 12/15
350/350 [==============================] - 672s 2s/step - loss: 1.0574 - accuracy: 0.6690 - val_loss: 1
.1192 - val_accuracy: 0.6544
Epoch 13/15
350/350 [==============================] - 676s 2s/step - loss: 1.0420 - accuracy: 0.6725 - val_loss: 1
.1282 - val_accuracy: 0.6524
Epoch 14/15
350/350 [==============================] - 676s 2s/step - loss: 1.0284 - accuracy: 0.6787 - val_loss: 1
```

```
.0977 - val_accuracy: 0.6619
Epoch 15/15
350/350 [==============================] - 679s 2s/step - loss: 1.0170 - accuracy: 0.6837 - val_loss: 1
.0660 - val_accuracy: 0.6708
```

In [ ]:

```
# !tensorboard dev upload --logdir /content/logs \
#   --name "Transfer Learning Model:3 (15 epochs)" \
#   --description " from TF_3.ipynb " \
#   --one_shot
```

In [ ]:

```

```