```python
from keras_preprocessing.image import ImageDataGenerator
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense,InputLayer,Conv2D,MaxPooling2D
from tensorflow.keras.models import Model
from keras.applications.vgg16 import VGG16
from tensorflow.keras.callbacks import EarlyStopping,Callback
```

```python
from google.colab import files
uploaded = files.upload()
for fn in uploaded.keys():
  print('uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving labels_final.csv to labels_final (1).csv
uploaded file "labels_final.csv" with length 2108171 bytes
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
!pip install pyunpack
!pip install patool
from pyunpack import Archive
Archive('/content/drive/MyDrive/rvl-cdip.rar').extractall('/content')
```

```
Collecting pyunpack
  Downloading pyunpack-0.2.2-py2.py3-none-any.whl (3.8 kB)
Collecting entrypoint2
  Downloading entrypoint2-0.2.4-py3-none-any.whl (6.2 kB)
Collecting easyprocess
  Downloading EasyProcess-0.3-py2.py3-none-any.whl (7.9 kB)
Installing collected packages: entrypoint2, easyprocess, pyunpack
Successfully installed easyprocess-0.3 entrypoint2-0.2.4 pyunpack-0.2.2
Collecting patool
  Downloading patool-1.12-py2.py3-none-any.whl (77 kB)
     |████████████████████████████████| 77 kB 3.2 MB/s
Installing collected packages: patool
Successfully installed patool-1.12
```

```python
df=pd.read_csv("labels_final.csv",dtype=str)
```

```python
df.head(5)
```

	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

```python
datagen=ImageDataGenerator(rescale=1./255.,validation_split=0.25)
```

```python
train_generator = datagen.flow_from_dataframe(
                  dataframe=df,
                  directory="/content/data_final",
                  x_col="path",
                  y_col="label",
                  has_ext=False,
                  subset="training",
                  batch_size=32,
                  shuffle=True,
                  class_mode="categorical",
                  target_size=(224, 224))

validation_generator=datagen.flow_from_dataframe(
                dataframe=df,
                directory="/content/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 36000 validated image filenames belonging to 16 classes.
Found 12000 validated image filenames belonging to 16 classes.
```

```python
import datetime
import os
! rm -rf ./logs/
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
print(logdir)
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir,histogram_freq=1, write_graph=True)
```

```
logs/20210928-182507
```

Model 1 Traning

```python
vgg=VGG16(weights='imagenet', include_top=False,input_shape = (224, 224, 3))
# don't train existing weights
for layer in vgg.layers:
  layer.trainable=False

top_model = Conv2D(filters=32, kernel_size=(3,3), activation='relu',padding='same',kernel_initializer=tf.keras.initializers.HeUniform())(vgg.output)
top_model=MaxPooling2D(pool_size=(2,2),strides=2)(top_model)
# Flatten the output layer to 1 dimension
top_model = Flatten()(top_model)
top_model = Dense(64, activation='relu',kernel_initializer=tf.keras.initializers.HeUniform())(top_model)
top_model = Dense(32, activation='relu',kernel_initializer=tf.keras.initializers.HeUniform())(top_model)
output_layer = Dense(16, activation='softmax')(top_model)

model = Model(inputs = vgg.input, outputs = output_layer)
print(model.summary())
```

```
Exception ignored in: <function IteratorResourceDeleter.__del__ at 0x7f6c6e67e8c0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops/iterator_ops.py", line 546, in __del__
    handle=self._handle, deleter=self._deleter)
  File "/usr/local/lib/python3.7/dist-packages/tensorflow/python/ops/gen_dataset_ops.py", line 1264, in delete_iterator
    _ctx, "DeleteIterator", name, handle, deleter)
KeyboardInterrupt:
Model: "model_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (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_16 (Conv2D)           (None, 7, 7, 32)          147488
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_10 (Dense)             (None, 64)                18496
_________________________________________________________________
dense_11 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_12 (Dense)             (None, 16)                528
=================================================================
Total params: 14,883,280
Trainable params: 168,592
Non-trainable params: 14,714,688
_________________________________________________________________
None
```

```python
# Before training a model, we need to configure the learning process, which is done via the compile method
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
 loss='categorical_crossentropy',
 metrics=['accuracy'])
# have to stop the training if  validation accuracy is not increased in last 2 epochs.
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.20, patience=4, verbose=1)
model.fit(train_generator, validation_data = validation_generator,steps_per_epoch=163, epochs = 10,callbacks=[earlystop,tensorboard_callback])
```

```
Epoch 1/10
163/163 [==============================] - 130s 792ms/step - loss: 2.5451 - accuracy: 0.2203 - val_loss: 2.2867 - val_accuracy: 0.3320
Epoch 2/10
163/163 [==============================] - 128s 786ms/step - loss: 2.1003 - accuracy: 0.3815 - val_loss: 1.9501 - val_accuracy: 0.4371
Epoch 3/10
163/163 [==============================] - 128s 785ms/step - loss: 1.8454 - accuracy: 0.4482 - val_loss: 1.7446 - val_accuracy: 0.4883
Epoch 4/10
163/163 [==============================] - 128s 786ms/step - loss: 1.6850 - accuracy: 0.4956 - val_loss: 1.6221 - val_accuracy: 0.5196
Epoch 5/10
163/163 [==============================] - 149s 915ms/step - loss: 1.5693 - accuracy: 0.5280 - val_loss: 1.5567 - val_accuracy: 0.5414
Epoch 6/10
163/163 [==============================] - 147s 904ms/step - loss: 1.4928 - accuracy: 0.5546 - val_loss: 1.4863 - val_accuracy: 0.5563
Epoch 7/10
163/163 [==============================] - 148s 911ms/step - loss: 1.4335 - accuracy: 0.5706 - val_loss: 1.4435 - val_accuracy: 0.5670
Epoch 8/10
163/163 [==============================] - 146s 900ms/step - loss: 1.4045 - accuracy: 0.5817 - val_loss: 1.3994 - val_accuracy: 0.5846
Epoch 9/10
163/163 [==============================] - 143s 879ms/step - loss: 1.3382 - accuracy: 0.6037 - val_loss: 1.3638 - val_accuracy: 0.5923
Epoch 00009: early stopping
```

```
<keras.callbacks.History at 0x7f6c59b20510>
```

```python
%load_ext tensorboard
%tensorboard --logdir $logdir
```

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

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 7×7×512 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 1×1×4096 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**
3. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

```python
vgg_model=VGG16(weights='imagenet', include_top=False,input_shape = (224, 224, 3))
# we are not training existing weights
vgg_model.trainable=False
top_model = Conv2D(4096,kernel_size=(1,1),padding="valid", activation='relu',kernel_initializer=tf.keras.initializers.HeUniform())(vgg_model.output)
top_model=Conv2D(4096,kernel_size=(1,1),padding="valid" ,activation='relu')(top_model)
# Flatten the output layer to 1 dimension
top_model = Flatten()(top_model)
output_layer = Dense(16, activation='softmax')(top_model)
model = Model(inputs = vgg_model.input, outputs = output_layer)
print(model.summary())
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (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_5 (Conv2D)            (None, 7, 7, 4096)        2101248
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 4096)        16781312
_________________________________________________________________
flatten_2 (Flatten)          (None, 200704)            0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                3211280
=================================================================
Total params: 36,808,528
Trainable params: 22,093,840
Non-trainable params: 14,714,688
_________________________________________________________________
None
```

```python
# Before training a model, we need to configure the learning process, which is done via the compile method
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
 loss='categorical_crossentropy',
 metrics=['accuracy'])
# have to stop the training if  validation accuracy is not increased in last 2 epochs.
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.20, patience=2, verbose=1)
model.fit(train_generator, validation_data = validation_generator,steps_per_epoch=220, epochs = 5,callbacks=[earlystop,tensorboard_callback])
```

```
Epoch 1/5
220/220 [==============================] - 231s 852ms/step - loss: 1.7850 - accuracy: 0.4756 - val_loss: 1.3178 - val_accuracy: 0.6050
Epoch 2/5
220/220 [==============================] - 186s 847ms/step - loss: 1.2206 - accuracy: 0.6280 - val_loss: 1.2061 - val_accuracy: 0.6433
Epoch 3/5
220/220 [==============================] - 186s 846ms/step - loss: 1.1177 - accuracy: 0.6595 - val_loss: 1.0847 - val_accuracy: 0.6777
Epoch 00003: early stopping
```

```
<keras.callbacks.History at 0x7f6c50287dd0>
```

```python
%load_ext tensorboard
%tensorboard --logdir $logdir
```

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.

```python
vgg_=VGG16(weights='imagenet', include_top=False,input_shape = (224, 224, 3))
# Freeze  convolution blocks training only last 6 layer
for layer in vgg_.layers[:13]:
    layer.trainable = False

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

```
0 input_6 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
```

```python
top_model = Conv2D(4096,kernel_size=(1,1),padding="valid", activation='relu',kernel_initializer=tf.keras.initializers.HeUniform())(vgg_.output)
top_model=Conv2D(4096,kernel_size=(1,1),padding="valid" ,activation='relu')(top_model)
# Flatten the output layer to 1 dimension
top_model = Flatten()(top_model)
output_layer = Dense(16, activation='softmax')(top_model)
model = Model(inputs = vgg_.input, outputs = output_layer)
print(model.summary())
```

```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (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_13 (Conv2D)           (None, 7, 7, 4096)        2101248
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 7, 7, 4096)        16781312
_________________________________________________________________
flatten_6 (Flatten)          (None, 200704)            0
_________________________________________________________________
dense_6 (Dense)              (None, 16)                3211280
=================================================================
Total params: 36,808,528
Trainable params: 31,533,072
Non-trainable params: 5,275,456
_________________________________________________________________
None
```

```python
# Before training a model, we need to configure the learning process, which is done via the compile method
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
 loss='categorical_crossentropy',
 metrics=['accuracy'])
# have to stop the training if  validation accuracy is not increased in last 2 epochs.
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.20, patience=2, verbose=1)
model.fit(train_generator, validation_data = validation_generator,steps_per_epoch=163, epochs = 10,callbacks=[earlystop,tensorboard_callback])
```

```
Epoch 1/10
163/163 [==============================] - 177s 1s/step - loss: 2.1504 - accuracy: 0.3317 - val_loss: 1.7481 - val_accuracy: 0.4532
Epoch 2/10
163/163 [==============================] - 175s 1s/step - loss: 1.5033 - accuracy: 0.5303 - val_loss: 1.3989 - val_accuracy: 0.5672
Epoch 3/10
163/163 [==============================] - 175s 1s/step - loss: 1.3339 - accuracy: 0.5964 - val_loss: 1.3170 - val_accuracy: 0.6037
Epoch 00003: early stopping
```

```
<keras.callbacks.History at 0x7f6b6128bed0>
```

```python
%load_ext tensorboard
%tensorboard --logdir $logdir
```

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

TensorBoard Observation

Model 1

a. We can see from the loss function at every epoch. Since the loss drops very quickly and stabilizes after the 2nd epochs, it indicates that our model has learned

b. We can see from the Accuracy function at every epoch. accuracy is not much improving .

Model 2

a. We can see from the loss function at every epoch. Since the loss drops very quickly and stabilizes after the 2nd epochs, it indicates that our model has learned

b. We can see from the Accuracy function at every epoch. accuracy is not much improving .

Model 3

a. We can see from the loss function at every epoch. Since the loss drops very quickly and stabilizes after the 2nd epochs, it indicates that our model has learned

b. We can see from the Accuracy function at every epoch. accuracy is not much improving .