

The screenshot shows a Jupyter Notebook titled "FeatureExtractionNotebook (2)" running on a local host. The notebook has a menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu bar, there are tabs for "Part1.pdf" and "Show all". The main area of the notebook contains three code cells. The first cell, labeled "In [57]:", imports various libraries including tensorflow, os, cv2, numpy, matplotlib, tqdm, datetime, json, sklearn, pickle, and LSTM\_XX\_Training. The second cell, labeled "In [2]:", enables eager execution using tf.enable\_eager\_execution(). The third cell, labeled "In [3]:", creates a MobileNetV2 feature extraction model using tf.keras.applications.mobilenet\_v2.MobileNetV2, and then defines the pooling\_output and feature\_extraction\_model using tf.keras.layers.GlobalAveragePooling2D and tf.keras.Model respectively. The output of the third cell is not visible in the screenshot.

```
In [57]: import tensorflow as tf
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tqdm
import datetime
import json
from sklearn.preprocessing import LabelBinarizer
import pickle
import import_ipynb
import LSTM_XX_Training

In [2]: # enable eager execution to guarantee the tensors have discrete instead of symbolic values,
# so you can do operations on them
tf.enable_eager_execution()
tf.executing_eagerly()

Out[2]: True

In [3]: # create MobileNetV2 feature extraction model
mobilenet_v2 = tf.keras.applications.mobilenet_v2.MobileNetV2(input_shape=(224,224,3), include_top=False, weights='imagenet')
x = mobilenet_v2.output

pooling_output = tf.keras.layers.GlobalAveragePooling2D()(x)
feature_extraction_model = tf.keras.Model(mobilenet_v2.input, pooling_output)
```

Import all dependencies

The screenshot shows the same Jupyter Notebook interface as before, but now the third code cell, labeled "In [3]:", is executed. The output of the cell is a summary of the feature\_extraction\_model. The summary is displayed as a table with columns: Layer (type), Output Shape, Param #, and Connected to. The table lists the layers of the MobileNetV2 model, including input\_1, Conv1\_pad, Conv1, bn\_Conv1, Conv1\_relu, expanded\_conv\_depthwise, and expanded\_conv\_depthwise\_BN. The output shape, number of parameters, and connections for each layer are provided. The fourth code cell, labeled "In [4]:", calls feature\_extraction\_model.summary(), which triggers the display of the model summary.

```
In [3]: # create MobileNetV2 feature extraction model
mobilenet_v2 = tf.keras.applications.mobilenet_v2.MobileNetV2(input_shape=(224,224,3), include_top=False, weights='imagenet')
x = mobilenet_v2.output

pooling_output = tf.keras.layers.GlobalAveragePooling2D()(x)
feature_extraction_model = tf.keras.Model(mobilenet_v2.input, pooling_output)

In [4]: feature_extraction_model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	
Conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	input_1[0][0]
Conv1 (Conv2D)	(None, 112, 112, 32)	864	Conv1_pad[0][0]
bn_Conv1 (BatchNormalization)	(None, 112, 112, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 112, 112, 32)	0	bn_Conv1[0][0]
expanded_conv_depthwise (Depthwise Conv2D)	(None, 112, 112, 32)	288	Conv1_relu[0][0]
expanded_conv_depthwise_BN (Batch Normalization)	(None, 112, 112, 32)	128	expanded_conv_depthwise[0][0]

Create a feature extraction model

```
In [5]: # save CNN model in SavedModel format for converting to tflite
SAVED_MODEL_DIR = 'C:/Users/STSC/Desktop/Project3/SavedModelDir/LSTM/v1'
tf.keras.experimental.export_saved_model(feature_extraction_model, SAVED_MODEL_DIR)

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:97: calling GlorotUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:97: calling Zeros.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:97: calling Ones.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\saved_model\signature_def_utils_impl.py:201: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
INFO:tensorflow:Signatures INCLUDED in export for Classify: None
INFO:tensorflow:Signatures INCLUDED in export for Detect: None
```

## Save the feature extraction model

```
In [26]: # base path of video dataset
BASE_PATH = 'C:\\Users\\STSC\\Desktop\\Project3\\Training_Set\\Full_Data'
VIDEOS_PATH = os.path.join(BASE_PATH, '**', '*.mp4')

# sequence length LSTM will process
SEQUENCE_LENGTH = 40
BATCH_SIZE = 16

In [27]: dataset = tf.data.Dataset.from_generator(frame_generator,
                                                output_types=(tf.float32, tf.string),
                                                output_shapes=((224, 224, 3), ()))

dataset = dataset.batch(BATCH_SIZE, drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)

In [11]: current_path = None
all_features = []

# go through the dataset and use the mobilenet_v2 model to
# extract the features for each frame
for img, batch_paths in tqdm.tqdm(dataset):
    batch_features = feature_extraction_model(img)
    # reshape the tensor
    batch_features = tf.reshape(batch_features,
                               (batch_features.shape[0], -1))

    for features, path in zip(batch_features.numpy(), batch_paths.numpy()):
        if path != current_path and current_path is not None:
```

Set the path for video dataset and go through all videos and perform feature extraction

```
Out[28]: LabelBinarizer(neg_label=0, pos_label=1, sparse_output=False)

In [29]: logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
         tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=0)

In [30]: callbacks=[tensorboard_callback]

In [31]: #set up a keras Sequential model with 1) Masking Layer 2) LSTM Layer with 512 cells, dropout 0.5, recurrent_dropout of 0.5
         # 3) a fully connected relu activation layer with 256 outputs, 4) a dropout layer 5) a final decision fully connected layer of
         # (which is the number of classes) with softmax activation
         model = tf.keras.Sequential([
             tf.keras.layers.Masking(mask_value=0.),
             tf.keras.layers.LSTM(512, dropout=0.5, recurrent_dropout=0.5),
             tf.keras.layers.Dense(256, activation='relu'),
             tf.keras.layers.Dropout(0.5),
             tf.keras.layers.Dense(len(LABELS), activation='softmax')
         ])

In [32]: model.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy', 'top_k_categorical_accuracy'])
```

setup a keras Sequential model

```
In [41]: # Setup the train_dataset and valid_dataset (validation/testing).
         # Here we setting up training batch sets of 16.

         train_dataset = tf.data.Dataset.from_generator(make_generator(train_list),
             output_types=(tf.float32, tf.int16),
             output_shapes=(tf.TensorShape([SEQUENCE_LENGTH, 1280]), tf.TensorShape([len(LABELS)])))
         train_dataset = train_dataset.batch(16, drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)

         valid_dataset = tf.data.Dataset.from_generator(make_generator(test_list),
             output_types=(tf.float32, tf.int16),
             output_shapes=(tf.TensorShape([SEQUENCE_LENGTH, 1280]), tf.TensorShape([len(LABELS)])))
         valid_dataset = valid_dataset.batch(16, drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)

In [42]: print(train_dataset)

<DatasetV1Adapter shapes: ((16, 40, 1280), (16, 37)), types: (tf.float32, tf.int16)>

In [44]: # fit the model with the training data (can increase the # of epochs & validation_steps if desired)
         model.fit(train_dataset, epochs=100, validation_data=valid_dataset, validation_steps=4)

ical_accuracy: 0.36 - 116s 1s/step - loss: 3.2105 - acc: 0.0885 - top_k_categorical_accuracy: 0.36 - 117s 1s/step - loss: 3.2
096 - acc: 0.0901 - top_k_categorical_accuracy: 0.36 - 118s 1s/step - loss: 3.2087 - acc: 0.0898 - top_k_categorical_accuac
y: 0.36 - 119s 1s/step - loss: 3.2095 - acc: 0.0895 - top_k_categorical_accuracy: 0.36180/180 [=====
].2078 - acc: 0.0893 - top_k_categorical_accuracy: 0.36 - 122s 1s/step - loss: 3.2094 - acc: 0.0890 - top_k_categorical_accu
```

Set up the training dataset and validation dataset and fit the model with training data.



