Object Detection Model Training

Initial Installation Instructions

Create virtual environment for the project
```
conda create --name tfod
```

List the virtual environments
```
conda info --envs
```

Activate the virtual environment
```
conda activate tfod
```

Install ipykernel in the virtual environment
```
conda -c install anaconda ipykernel
python -m ipykernel install --user --name=tfod
jupyter notebook
```

If the final command does not work, do the following commands and try again
```
pip uninstall pyzmq
pip uninstall jsonschema
pip install pyzmq
pip install jsonschema
pip install jupyter
pip install opencv-python
jupyter notebook
```

Download the following libraries for the labelImg
```
pip install pyqt5
pip install lxml
```

Open up a new notebook in the specified virtual environment. This can be changed on the top right of the page
Choose tfod here

Label Images using software like labelImg
- Different angles and lighting conditions
- 10-20 pictures of each class
- Labels as tight as possible

Uploaded sample labels: cars, trucks, and pedestrians

Create a directory called Tensorflow/workspace/images/collectedimages and add the Images used in the dataset.

Create a test and train folder to partition the labeled images in our dataset - Going for an 80-20 split where 80 % of the images are training and 20% of the images are test
The directory for these images will be Tensorflow/workspace/images/test and Tensorflow/workspace/images/test

Training Models

Precision = #TP/(#TP + #FP)
Recall = #TP/(#TP + #FN)
T True F False N Negative P Positive
Loss = keep it low, this detects how well our method performs against provided data
Accuracy is inversely proportional to the speed of detection

Check out the tf detection zoo object detection models at the following link:
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
For the purpose of this model training, we will use the following model

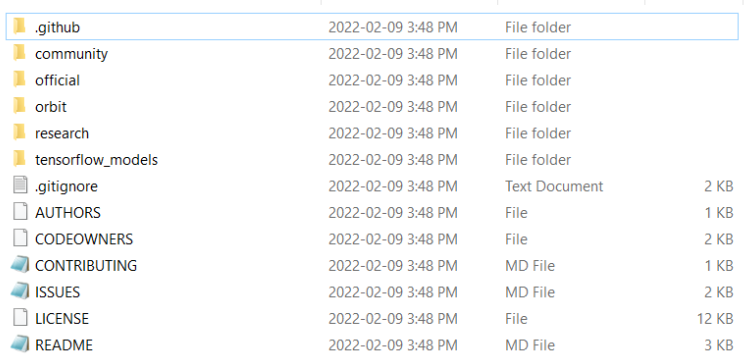
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes

MAP = Mean average precision is 22.2
Speed in ms is 22ms
320X320 conversion is performed by the model itself in pre-processing stage

Following this task, some necessary directories need to be created. This can be done by running the ipynb file called model_training.ipynb

Install wget using pip
```
pip install wget
```

Install the Tensorflow models from github
Clone the tensorflow models repository in the Tensorflow/models directory
https://github.com/tensorflow/models
The models directory should look like this:



The protoc install is used for protocol buffers

The following installations had to be made to make sure the verification script provided in model_training.ipynb works

```
!pip install tensorflow --upgrade
!pip install cycler
!pip install kiwisolver
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
!pip install pyyaml
```

The output of the verification script should be as follows

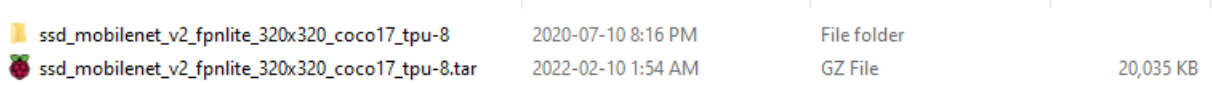
```
I0210 01:32:19.816815 19808 test_util.py:2373] time(__main__.Mode
0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_me
I0210 01:32:19.816815 19808 test_util.py:2373] time(__main__.Mode
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extract
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ss
I0210 01:32:19.817816 19808 test_util.py:2373] time(__main__.Mode
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extract
----------------------------------------------------------------------
Ran 24 tests in 18.690s

OK (skipped=1)
```

Once the verification script runs properly, the object_detection api can be imported
import object_detection
Restart the kernel and run the code again. Except for the file creations, setup and git clone

Install the pretrained model

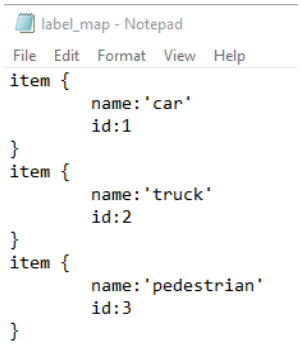
Install the pretrained model from the tensorflow zoo and add it to the Tensorflow/workspace/pre trained models directory
The directory should include a tar.gz file and extracted folder as follows



The already existing model assists in custom model training
The extracted folder will contain all information we usually have when we freeze our model to help in exporting it

Create a Label Map File

The labelmap.pbtxt file for our current model training should look as follows:



Generate TF Records

Upon running the code for generating the records, the output should be shown as follows

```
Successfully created the TFRecord file: Tensorflow\workspace\annotations\train.record
Successfully created the TFRecord file: Tensorflow\workspace\annotations\test.record
```

NOTE: Make sure the labels in label map file are accurate or record generation does not work

Updating pipeline.config file based on your own model

Refer to the model_training.ipynb

Training the model in python

This is longest step and this is where a lot of resolveable module errors come up
Number of steps can be limited using an option --num_train_steps=xxxx
But ideally the model should be kept running until the total_loss is between 0.15-0.2

```
I0210 19:52:32.169100 28688 model_lib_v2.py:705] Step 8000 per-step time 0.856s
INFO:tensorflow:{'Loss/classification_loss': 0.05318356,
 'Loss/localization_loss': 0.021380665,
 'Loss/regularization_loss': 0.11949028,
 'Loss/total_loss': 0.1940545,
 'learning_rate': 0.07603875}
I0210 19:52:32.170100 28688 model_lib_v2.py:708] {'Loss/classification_loss': 0.05318356,
 'Loss/localization_loss': 0.021380665,
 'Loss/regularization_loss': 0.11949028,
 'Loss/total_loss': 0.1940545,
 'learning_rate': 0.07603875}
```

As in the above picture, the Loss/total_loss is 0.194 which is acceptable
Also visible is the step number 8000

Evaluate the Model

Command available in the model_training.ipynb. The output is as follows:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.391
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.615
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.444
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.115
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.264
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.645
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.210
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.498
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.528
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.210
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.461
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.676
```

Evaluating the Model using Tensorboard

In the specified path, the train and eval folders can be found

```
(tfod) C:\Users\tegve\Projects Winter 2022\TFOD Capstone\Tensorflow\workspace\models\my_ssd_mobnet>
```

If the tensorboard is opened in the train folder, we get loss metrics

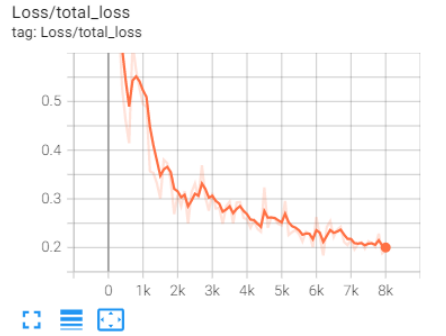
If the tensorboard is opened in the eval folder, we get evaluation metrics

Navigate to the any of these directories and run

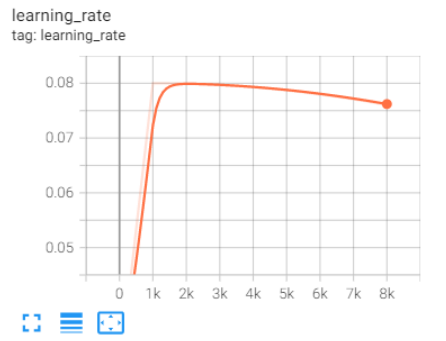
`tensorboard --logdir=.`

Tensorboard will run on localhost 6006

Following is the output of the tensorboard



learning_rate



The my_ssd_mobnet folder represents our custom model with its own checkpoints and pipeline