IMAGE RECOGNITION USING MACHINE LEARNING

INSTALLATION:

Inorder to implement custom image recognition we need to install python3 and tensorflow-gpu

pip install tensorflow-gpu==2.2.0

Clone the tensorflow model repository using the command:

git clone https://github.com/tensorflow/models.git

Install protobuf compiler :

apt install protobuf-compiler

Install the required python packages:

```
cd models/research
# compile protos:
protoc object_detection/protos/*.proto --python_out=.
# Install TensorFlow Object Detection API as a python package:
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

Inorder to test the installation:

python object_detection/builders/model_builder_tf2_test.py

Installation of additional dependencies:

```
pip install opencv-python
pip install opencv-contrib-python
```

To clone the reference github repository:

git clone
https://github.com/abdelrahman-gaber/tf2-object-detection-api-tutorial.git

PREPARING CUSTOM DATASET FOR TRAINING:

- Obtain the required images to be trained
- In this project, we have obtained 5 different varieties of fishes for training the model
- Each category of fishes are placed in individual folders
- Inorder to provide custom labels, we have used the **LabelImg** tool
- **Installation:**

```
sudo apt-get install pyqt5-dev-tools
sudo pip3 install -r
requirements/requirements-linux-python3.txt
make qt5py3
python3 labelImg.py
python3 labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

LabelImg:

- Open the required directory where the images are placed
- Label the images as per preference and save the image in the required folder
- This tool converts the images into .xml files

TRAINING AND TESTING DATASETS:

- Create separate folders for training and testing
- Place 20% of the total images in the testing images folder and the rest in the training images folder
- Repeat the same for the .xml files and place them in the respective annotation folders

TRAINING OBJECT DETECTION MODEL WITH CUSTOM DATASET:

- First you need to convert the xml annotations files to csv
  cd data_gen
  python xml_to_csv.py --annot_dir ../data/raccoon_data/train/annotations --out_csv_path ../data/raccoon_data/train_labels.csv
- After generating the csv file, use it to generate the tfrecord files. You can find the file in data_gen/generate_tfrecord.py, and you can use it as follows:
  *python generate_tfrecord.py --path_to_images ../data/raccoon_data/train/images *
  *--path_to_annot ../data/raccoon_data/train_labels.csv *
  *--path_to_label_map ../models/raccoon_labelmap.pbtxt *
  --path_to_save_tfrecords ../data/raccoon_data/train.record

(or)

- Instead of the previous steps, just run this shell file as follows:
  cd data_gen/
  bash gen_data.sh

- To start training our model, we need to prepare a configuration file
  cd models/
  # download the mobilenet_v2 model
  wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz
  # extract the downloaded file
  tar -xzvf ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz

We downloaded ssd_mobilenet_v2_320x320_coco17_tpu-8.config and made the following changes:

- Used **num_classes**: 1 as we have only one class (raccoon), instead of 90 classes in coco dataset.
- Changed **fine_tune_checkpoint_type**: "classification" to fine_tune_checkpoint_type: "detection" as we are using the pre-trained detection model as initialization.
- Added the path of the pretrained model in the field fine_tune_checkpoint:, for example using the mobilenet v2 model I added fine_tune_checkpoint: "../models/ssd_mobilenet_v2_320x320_coco17_tpu-8/checkpoint/ckpt-0"
- Changed **batch_size**: 512 and used a reasonable number to my GPU memory. I have a 4GB of GPU memory, so I am using batch_size: 16
- Added the maximum number of training iterations in **num_steps**:, and also used the same number in **total_steps**
- Adapted the learning rate to our model and batch size (originally they used higher learning rates because they had bigger batch sizes). This values needs some testing and tuning, but finally I used this configuration:

  cosine_decay_learning_rate {
  learning_rate_base: 0.025
  total_steps: 3000
  warmup_learning_rate: 0.005
  warmup_steps: 100 }

- The **label_map_path**: should point to our labelmap file (here the fish labelmap) label_map_path: "../models/raccoon_labelmap.pbtxt"
- You need to set the tf_record_input_reader under both train_input_reader and eval_input_reader. This should point to the tfrecords we generated (one for training and one for validation).

  train_input_reader: {
  label_map_path: "../models/raccoon_labelmap.pbtxt"
  tf_record_input_reader {
  input_path: "../data/raccoon_data/train.record"
  }
  }

- You should also prepare the labelmap according to your data. For our fish dataset, the labelmap file contains:

  item {
  id: 1

```
  name: 'Auri_Fish'
}
item {
  id: 2
  name: 'Catla_Fish'
}


item {
  id: 3
  name: 'Kari_Fish'
}
item {
  id: 4
  name: 'Rohu_Fish'
}
item {
  id: 5
  name: 'Tilapia_Fish'
}
```

- The labelmap file and the modified configuration files are added to this repo. You can find them in models/raccoon_labelmap.pbtxt and models/ssd_mobilenet_v2_raccoon.config.
- Once you prepare the configuration file, you can start the training by typing the following commands:

```
# you should run training script from train_tf2/ directory
cd train_tf2/
bash start_train.sh
```

- The start_train.sh file is a simple shell script that runs the training with all the required parameters. The shell file contains the following command:

```
out_dir=../models/ssd_mobilenet_v2_fish/
mkdir -p $out_dir
python model_main_tf2.py --alsologtostderr --model_dir=$out_dir
--checkpoint_every_n=500  \
```

*--pipeline_config_path=../models/ssd_mobilenet_v2_fish.config *
--eval_on_train_data 2>&1 | tee $out_dir/train.log

EXPORTING TRAINED MODEL FOR INFERENCE:

- Start the inference by running this shell script.
  cd train_tf2/
  bash export_model.sh

- The export_model.sh file contains:

  model_dir=../models/ssd_mobilenet_v2_fish
  out_dir=$model_dir/exported_model
  mkdir -p $out_dir

  # start the exporter
  *python exporter_main_v2.py *
  *--input_type="image_tensor" *
  *--pipeline_config_path=$model_dir/pipeline.config *
  *--trained_checkpoint_dir=$model_dir/ *
  --output_directory=$out_dir

- The export directory will contain 2 folders; saved_model and checkpoint. The saved_model directory contains the frozen model, and that is what we will use for inference. The checkpoint contains the last checkpoint in training
- Now we will use our model trained with fish dataset, so we need to set the path to the fish labelmap, and our frozen model

python3 trail.py --video_input --threshold 0.8 --model_path models/ssd_mobilenet_v2_fish/exported_model/saved_model \--path_to_labelmap models/fish_labelmap.pbtxt