Image Recognition with TensorFlow Deeplearning in ROS

Use a TensorFlow Model that has learned hundreds of images from the ImageNet Database

catkin_create_pkg tf_unit1_pkg rospy std_msgs sensor_msgs

cd .. ; source devel/setup.bash ; rospack profile

Scripts :

- One responsible for retrieving ROS images from a topic and sending them to a classification class that will decide which objects are in the scene

  - classify_image.maybe_download_and_extract()

    It will download the freezed_tensorflow_model -> classify_image_graph_def

    from the imagenet_2012_challenge. It's a model prepared for image recognition of hundreds of objects, trained with thousands of prelabeled images -> imagenet_synset_to_human_label_map -> 21840 different classes

  - self._session = tf.Session()

    starting a TensorFlow session -> access to all functionalities and be able to use tensor soft-max from the downloaded model

  - classify_image.create_graph()

    initialising all required elements for recognition using the model

  - self._cv_bridge = CvBridge()
  - self.score_threshold = rospy.get_param('~score_threshold', 0.1)

    you can increase up to 1.0. The higher the value, the most sure the detection has to be to consider it a correct and a valid one The lower, the more detections

- classify_image.py

This one does all the heavy lifting of preparing the downloaded model to have human readable tags

- DATA_URL = "download.tensorflow.org/models/image/imagenet/....tgz"

It will download and extract the compressed model and labeling that you choose to use in /tmp/imagenet folder each time you run the script so you have updated versions. You can also download it to your package and use it from there

Launch TensorBoard and inspect a TensorFlow Model

To visualise Deep learning Model file graphically

to see how learning process is going

1° Select a Tensorboard model → download dataset

2° Generate the log files for TensorBoard from the model

From the model from the previous episode (classify_image_graph_def.pb

TensorBoard needs to convert it to .log files

import_pb_to_tensorboard.py

- ↳ python import_pb_to_tensorboard.py --model_dir =.../showcase_pb_mod
  --log_dir = learning_logs

- tensorboard --logdir = learning_logs

3° Connect to tensorboard

- public_ip
- in the browser : ip : 6006

Train you own TensorFlow image recognition model

What happens if you want to recognise something that is not on the ImageNet model list?

1. Labeling images

The most time-consuming task, unavoidable if you want to train a custom ~~model~~ element

lableImg → generates .xml based on images and how you label them
↳ so you dont have to write by hand

∘ python3 /home/user/.labelImg/labelImg.py

Once labeled, copy 10 % of images into a test folder and the other 90% into a train folder

IMPORTANT: images in test folder DON'T AAPEAR in train folder
This guarantees that when testing, training model is tested with images it doesn't know

2. Prepare Image Data for TensorFlow training

TensorFlow needs .records instead of .xml

.xml → .csv → .record
1st 2nd

(rm -rf → linux command to delet folders entirely)

xml_to_csv.py

∘ python scripts/xml_to_csv.py

generate_tfrecord_n.py + extract_training_labels.py
[... weird middle process] → copy from repo course_tflow_image_student_data
and compile the protobuffer

∘ python scripts/generate_tfrecords n.py --image_path_input = images/train
--csv_input = data/train_labels.csv
--output_path = data/train.record

• "do the same for test"

3° Copy Model Data for training

4° Create label list file object-detection.pbtxt

5° Time to train

6° Tensorboard to check training process

7° Export inference graph

8° Copy validation images

9° Launch testing training Script

10° Launch testing training script

Domain Randomization in DNN with Keras

- Use a Deep Convolutional Neural Network trained with images to recognize an object and know where it is located in the world

1° Train a NN so it can recognise an object and tell us its location in 3D space

2° Use the info from the recognition and send it to the robot to do something

- Keras: high-level NN API allows to use transparently TensorFlow (We will use MobileNetV2 model to make robot learn)

a)
```
catkin_create_pkg my_randomgazebomanager_pkg gazebo_ros rospy roscpp
```
b)
```
mkdir launch ; mkdir scripts
```
c) create_training_material.py
```
touch ./scripts/create_training_material.py
chmod +x ./ "            "
```

- ...
- store an image of the scene
- get pose data of the SpamCan -> it does it through gazebo