Workflow eigenes Modell auf Raspi mit eigen trainiertem Modell

1. Folge dem Tutorial https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#exporting-a-trained-inference-graph

1. If you have followed the tutorial, you should by now have a folder `Tensorflow`, placed under `<PATH_TO_TF>` (e.g. `C:\Users\sglvladi\Documents`), with the following directory tree:

```
TensorFlow
├─ addons
|    └── labelImg
└─ models
    ├── official
    ├── research
    ├── samples
    └── tutorials
```

2. Now create a new folder under `TensorFlow` and call it `workspace`. It is within the `workspace` that we will store all our training set-ups. Now let's go under workspace and create another folder named `training_demo`. Now our directory structure should be as so:

```
TensorFlow
├─ addons
|    └── labelImg
├─ models
|    ├── official
|    ├── research
|    ├── samples
|    └── tutorials
└─ workspace
    └── training_demo
```

3. The `training_demo` folder shall be our *training folder*, which will contain all files related to our model training. It is advisable to create a separate training folder each time we wish to train a different model. The typical structure for training folders is shown below.

```
training_demo
├─ annotations
├─ images
|    ├── test
|    └── train
├─ pre-trained-model
├─ training
└─ README.md
```

- `annotations` : This folder will be used to store all `*.csv` files and the respective TensorFlow `*.record` files, which contain the list of annotations for our dataset images.
- `images` : This folder contains a copy of all the images in our dataset, as well as the respective `*.xml` files produced for each one, once `labelImg` is used to annotate objects.
    - `images\train` : This folder contains a copy of all images, and the respective `*.xml` files, which will be used to train our model.
    - `images\test` : This folder contains a copy of all images, and the respective `*.xml` files, which will be used to test our model.
- `pre-trained-model` : This folder will contain the pre-trained model of our choice, which shall be used as a starting checkpoint for our training job.
- `training` : This folder will contain the training pipeline configuration file `*.config`, as well as a `*.pbtxt` label map file and all files generated during the training of our model.
- `README.md` : This is an optional file which provides some general information regarding the training conditions of our model. It is not used by TensorFlow in any way, but it generally helps when you have a few training folders and/or you are revisiting a trained model after some time.

If you do not understand most of the things mentioned above, no need to worry, as we'll see how all the files are generated further down.

2. Label the Images with LabelImg
3. Create TF-Records
    a. Entweder weg über xml -> csv -> tf records
        i. `# python generate_tfrecord.py --label=ship --csv_input=C:\Users\sglvladi\Documents\TensorFlow\workspace\training_demo\annotations\train_labels.csv --output_path=C:\Users\sglvladi\Documents\TensorFlow\workspace\training_demo\annotations\train.record --img_path=C:\Users\sglvladi\Documents\TensorFlow\workspace\training_demo\images\train`

    b. Oder direkt mit Skript dataset_to_tfrecord.py in einem Zug
4. Download pre trained model from Model Zoo
    a. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md#coco-trained-models-coco-models
5. Download according config file
    a. https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs
6. Edit Config file
    a. Num_classes
    b. Speicherort tfrecords und label_map
7. Train the model

Before we begin training our model, let's go and copy the `TensorFlow/models/research/object_detection/legacy/train.py` script and paste it straight into our `training_demo` folder. We will need this script in order to train our model.

Now, to initiate a new training job, `cd` inside the `training_demo` folder and type the following:

```
python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/ssd_inception_v2_co
```

8. Watch it on tensorboard

```
tensorboard --logdir=training\
```

9. Export frozen graph

```
python export_inference_graph.py --input_type image_tensor --pipeline_config_path
training/ssd_inception_v2_coco.config --trained_checkpoint_prefix
training/model.ckpt-38456 --output_directory trained-inference-graphs/
```

10. (Install Openvino on Main Machine if not done yet)
11. (Install Openvino on Raspi)
12. Convert TF Modell to IR
    a. CD in C:\Program Files (x86)\IntelSWTools\openvino_2019.1.148\deployment_tools\model_optimizer
    b. Copy frozen graph and pipeline.config in directory
    c. Copy ssd_support.json in directory
    d. Run as admin:

```
mo_tf.py \
  --input_model ~/Path/to/frozen_inference_graph.pb \
  --tensorflow_use_custom_operations_config
/opt/intel/computer_vision_sdk/deployment_tools/model_optimizer/extensi
ons/front/tf/ssd_support.json \
  --tensorflow_object_detection_api_pipeline_config
~/Path/to/pipeline.config \
  --data_type FP16
```

13. Copy the created Files in Directory and copy it on raspi
14. Make new directory on raspi
15. Run

```
python3 object_detection_demo_ssd_async.py \
  -m frozen_inference_graph.xml \
  -i cam  \
  -d MYRIAD \
  -pt 0.6
```

https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_tf_specific_Convert_Object_Detection_API_Models.html

https://www.pyimagesearch.com/2019/04/08/openvino-opencv-and-movidius-ncs-on-the-raspberry-pi/