Lab04 report

Goal

The goal of Lab04 is to implement a model based on CNN (Convolutional Neural Networks), train it and deploy it in Raspberry Pi. Tensorflow framework is used in this project.
A pretrained model is used and by transfer learning the final model will be able to identify two new objects, one with cylindrical form and the other with quadratic form.
Different steps need to be taken before a fully functional model works.
The final model is then tested on a robotic arm that sorts these two objects.

Test result

I used different pretrained models, see summary below.

Parameters that I took in consideration during this project were:

Training time: It took much more time to train the model on the PC machine without GPU than with GPU. Also training time depended on the complexity of the model.

Accuracy: During the transfer learning the accuracy of the model depended on the amount of data available. If the dataset contained very few test and training images the model would perform poorly. I ended up using hundred images, 20% test and 80% training.

Performance: To test the performance I used a webcam connected to the PC machine and Raspberry Pi. I measured the FPS from the webcam. I didn't notice any performance loss on the PC machine, the object was detected without any time-lag. However, on Raspberry Pi the same model performed poorly. I needed to find a lightwave model, see summary below.

Models summary

The keywords in all models that I tested are explained below:

MobileNet is a family of light machine learning models that trade accuracy for speed and performance. It is designed to be used in mobile or embedded devices.

SSD stands for Single Shot MultiBox Detector, refers to a method for detecting objects using a single neural network. The counterpart of this “single-shot” method is an architecture that uses an extra component known as the “proposal generator” to find regions of interest with an image.

COCO stands for “Common Objects in Context”, the dataset used to train the model.
It is a collection of over 200k labeled images separated across 90 classes that include “bird, “cat,” “person,” and “car.”

After testing different models I ended up using **faster_rcnn_inception_v2_coco_2018_01_28.**

Below is a summary of each pretrained model that I tested in this project with some metrics:

Pretrained model	**faster_rcnn_inception_v2_coco_2018_01_28**								
Training time without GPU	10h	**Training time with GPU**	4h	**Performance on PC**	Good	**Performance on Raspberry Pi. FPS (Frames Per Second)**	Acceptable 2 > FPS	**Accuracy**	good

Pretrained model	**ssd_mobilenet_v1_coco_2018_01_28**								
Training time without GPU	8h	**Training time with GPU**	3h	**Performance on PC**	Good	**Performance on Raspberry Pi. FPS (Frames Per Second)**	Not Acceptable 0.5 < FPS	**Accuracy**	Not good

Pretrained model	**ssd_mobilenet_v1_coco_2017_11_17**								
Training time without GPU	8h	**Training time with GPU**	4h	**Performance on PC**	Good	**Performance on Raspberry Pi. FPS (Frames Per Second)**	Not acceptable 0.5 < FPS	**Accuracy**	Not good

Pretrained model	**ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03**								
Training time without GPU	10h	**Training time with GPU**	6h	**Performance on PC**	Good	**Performance on Raspberry Pi. FPS (Frames Per Second)**	Not acceptable 0.5 < FPS	**Accuracy**	Not good

Issues

The main issue was to configure the environment on the PC machine.
I encountered compatibility problems with drivers for GPU and also libraries for tensorflow and Python. I was compelled to use older versions of libraries to get it to work.
Another problem was performance, the trained model worked fine on the PC machine but was very slow on Raspberry pi.

Conclusion

I learned a lot during this project.
It wasn't an easy task to configure the host machine, it took me many hours.
Also to find a model that performed well on Raspberry Pi wasn't an easy task.
But I am satisfied with the final result, the performance and accuracy of the trained model on Raspberry Pi is acceptable.
Instead of Raspberry Pi I should use for example the more powerful *nvidia jetson nano* with built-in GPU. There is also the camera Pixy2 that has built in object detection, no need to train a model.

The code

I didn't implement any new code in this project. The work was to modify the configuration files in the Tensorflow framework to train the model.

The python code that I modified is a part of TensorFlow distribution that can be downloaded from the following Github:
https://github.com/tensorflow/models.git

You can find the code that I modified in the following Github repo:
https://github.com/ouaburst/lab04

Setup Instructions

The main steps taken to build the model are the following:

1. Install and configure Tensorflow on a Windows machine. Preferably with GPU.
2. Train the model.
3. Export the model.
4. Run the model on Raspberry PI
5. Optional: Use the model on a robotic arm to sort objects.

The instructions for this lab are published on a wiki page.
You can find all the information on the following pages:

TensorFlow's Object Detection API	http://wiki.obkonsult.com/index.php/TensorFlow%27s_Object_Detection_API
Install Tensorflow on Raspberry Pi	http://wiki.obkonsult.com/index.php/Install_Tensorflow_on_Raspberry_pi

The first part of the tutorial contains detailed information on how to set up TensorFlow's Object Detection API on a Windows machine with a GPU, then train a model and export Inference Graph.

The second part of the tutorial contains detailed information on how to set up TensorFlow on a Raspberry Pi 4, import the pretrained model from a Windows machine and detect objects with a webcam attached to the Raspberry Pi.

The whole solution is then used with a 4 DOF robotic arm to sort the objects (optional).

You can see the robot arm in action on Youtube:
https://www.youtube.com/watch?v=NaAKmE5LD44

If you want to know how I built the robotic arm please let me know and I will give you all the information.

Download the whole tensorflow directory that was created during the first part of the tutorial on Windows.	https://drive.google.com/file/d/1Y-g32OmZsrv7BgZWZZ0WihCAuW_ER79R/view?usp=sharing
Download the whole tensorflow directory that was created during the second part of the tutorial on Raspberry Pi.	https://drive.google.com/file/d/1xqqdJcwTe7cSTarov-cAnWGCuQp-kNCV/view?usp=sharing