*Tensorflow Object Detection*

*on Raspberry Pi and Intel Movidius*

Melentev Nikita

Skoltech

Moscow, Russia, 2019

ndmelentev@skoltech.ru

*Abstract* — This paper describes parameters such as time of execution, overheating and CPU load using Raspberry Pi alone and with Intel Movidius to execute pre-trained neural networks for object detection. Several sets of experiments were conducted on neural networks with different depth and input size. Limitations of using Movidius are described and guide as well as code to launch experiments is provided.

Keywords—machine learning, object detection, neural networks, raspberry pi, embedded systems, movidius, tensorflow.

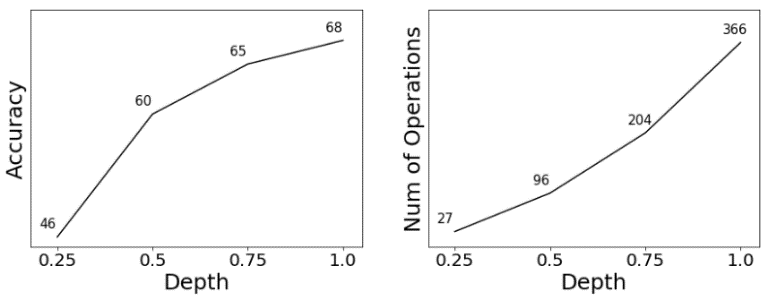
# Introduction

With recent progress in machine learning there as still some cases in which we cannot use the full potential of strong algorithms such as neural networks due to lack of computational power of some devices. One way to improve this situation is to use portable extensions to execute these algorithms. In our work we will use Intel Movidius [0] with Raspberry Pi.

To test the difference between raw Raspberry Pi and Movidius performances Mobilenet\_v1 [1] neural networks with different depth and image input size are used. They are pre-trained on ILSVRC-2012-CLS [2] dataset and can predict up to 1000 classes of objects. The following measurements are compared: FPS that camera can obtain, CPU load, CPU temperature, RAM load.

# Requirements and Specification

## Harware requirements

* Raspberry Pi 3 model B;
* SD card running Raspbian stretch;
* Movidius USB stick;
* Power cable, USB extension cable;
* Keyboard, mouse;
* HDMI screen, USB camera;
* Internet access (Ethernet or Wi-fi).

## Software requirements

* Raspbian stretch OS;
* Python3.5;
* Tensorflow;
* Movidius neural compute SDK.

# Preparing Raspberry Pi for Experiments

Movidius can only work with graphs that can be converted from Tensorflow or Caffee models. Because of this, we will use pure Tensorflow neural networks. So, the first step to launch experiments on Raspberry Pi is to install Tensorflow and all its dependencies [3].

After successful installation we can download and execute pre-trained Mobilenet v1 tensorflow models.

The next step is to convert previously downloaded models into Movidius format [4][5]. Another limitation that we must consider is that only models with strict input size can be converted.

Finally, we can now launch the same models but using power of Movidius and check how much performance did we gain.

Step by step instructions are available in my GitHub repository [6].

# Experiments

Mobilenet v1 set of neural networks are different in depth and image input size. Depth is encoded in four classes: 0.25, 0.5, 0.75 and 1, each class corresponds to accuracy that can be achieved and number of iterations in such network (Figure 1). As we can see, greater depth parameter results in greater accuracy, but we will need to do additional computations.

Figure 1

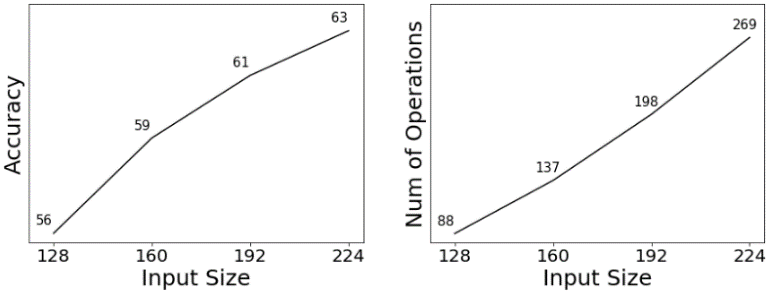
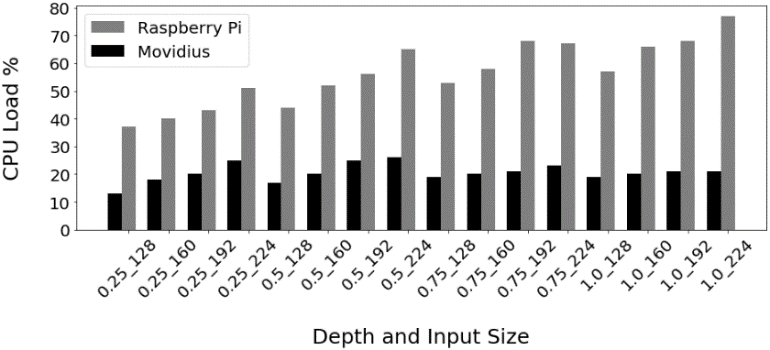
Similar situation can be observed with input size. It is encoded in four classes: 128, 160, 192 and 224 pixels. All input sizes are square. With increase in input size, accuracy is increased but so is number of iterations (Figure 2).

Figure 2

On figures we will use combined notation of model depth and input size.

## FPS

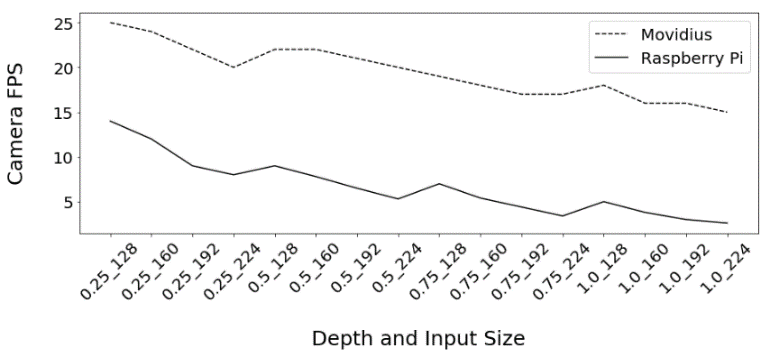
Let’s start with comparing FPS that camera can obtain. On each successfully captured frame neural network is executed (Figure 3).

Figure 3

For the majority of the experiments FPS achieved with Movidius is staying near 20, while mean value of Rasbperry Pi FPS is 7. In general, for object detection applications 1 FPS is enough, so Raspberry Pi can be used even for stronger Mobilenet neural networks in terms of frames-per-second metric.

## CPU and RAM Load

Next let’s look and CPU and RAM load figures (Figure 4 and Figure 5). In real applications, in addition to running camera with neural networks predicting labels, we will probably want some other processes, like writing to database or other system scripts. It is important to take this into account, because as we can see, only running heavy neural network on Raspberry Pi can take up almost all of CPU power and a considerable portion of RAM.

Figure 4

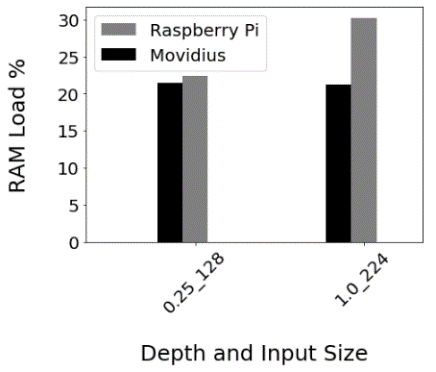
Using Movidius does not affect CPU, and only 20 % is used for image preprocessing. On the contrary, using only Raspberry Pi takes additional 20 to 60 % of CPU depending on the neural network size.

Figure 5

On Figure 5 we compare RAM Load while using the lightest and the heaviest models. Movidius requires the same amount of memory for both cases, which is around 22%. If using only Raspberry, we will need additional 10% when launching the heavies model.

## CPU Temperature

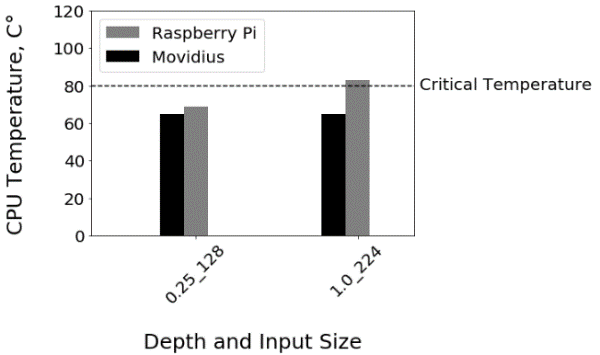
The most critical part of our experiments is to determine, if we can safely execute neural networks for an extended period of time. If we have sufficient power supply, the only thing that we need to avoid is overheating. On the next figure is CPU Temperature comparison while using only Raspberry Pi and Movidius (Figure 6).

Figure 6

Once again we compare CPU Temperature with lightest and heaviest models. Running Movidius heats up CPU to 64° in both experiments. But executing the same neural networks only on Raspberry results in overheating in the second experiment with the largest model. This means that if we to continue running this example over the extended period of time, we will face problems such as reduced performance, power overconsumption and in long terms, CPU quality reduction.

In case with Mobilenet v1 set of nerural networks, we can execute up to depth = 1 and input size = 160 models on Raspberry Pi without Movidius. Heavier models (with depth 1 and input sizes 192 and 224 pixels) result in overheating and we should use Movidius.

# Conclusion

To successfully launch neural networks on embedded systems we should monitor CPU Load and CPU Temperature, otherwise our device can malfunction. In case of Raspberry Pi we can launch most of Movilenet v1 models without such risk. But there are cases when algorithm takes up too much space and executing it results in overheating when using heavier models and we should use portable devices such as Intel Movidius to increase performance.

# References

[0] Intel Movidius. Neural Comput SDK. <https://movidius.github.io/ncsdk/index.html>

[1] Mobilenet pre-trained neural network models. <https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md>

[2] Large Scale Visual Recognition Challenge 2012.

<http://www.image-net.org/challenges/LSVRC/2012/>

[3] EdgeElectronics. Setting up TensorFlow on Raspberry Pi. <https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi>

[4] Humphrey Sheil. Movidius with Raspberry Pi. <https://medium.com/@hsheil/movidius-neural-compute-stick-and-raspberry-3-quick-start-guide-a89ff5e1d7ca>

[5] Intel Movidius. Guide to compile tensorflow models. <https://movidius.github.io/ncsdk/tf_compile_guidance.html>

[6] Melentev Nikita. Complete guide to launch object detection of Raspberry Pi with Movidius. <https://github.com/ndmel/tensorflow_object_detection_on_raspberry_pi>