NN Performance with different Kernel Schedulers (CFS, RT-Patch) on Raspberry Pi Platforms.

Pawan Kumar, 2018CSZ8014

November 2018

1 Introduction

Sensor data are useful in a variety of ways and our sensor are getting increasingly accurate. With embedded system and mobile phone, the sensors have spawned homes, offices, road and what not. The data available from sensor like accelerometer, guroscope etc. can be used to predict, in case of a vehicle, transportation mode [3]: bus, car, truck etc.; to infer driving style, and driver's behavior [1]; to predict drving events [3] [2] like acceleration-decceleration, turns etc. These information are valuable, they can assist a driver by alarming him/her in case of uncontrolled or dangerous events. Similarly in case of an wearable embedded device, the same sensor data can be used to tell state of the person weather sleeping, running, walking or just sitting[1]. These sensor data can also be used by attackers to get privacy related information like current location, current running apps etc. This report present result on latency for schedulers cfs and rt-patch (section 5 and 6) when a pre-trained neural network, section 3.1 is run in loop. The section 2 gives a summary of works on using sesors for vehicles mode and drving activity. Section 3 relates to work on Activity recognition. Section 3.1 talk about dataset used to train a NN that classifies if a person is walking or running.

2 Vehicle Mode and Driving Activity

The paper by Lu et al [3] has a combined model which uses various sensor available on a smartphone to predict vehicle mode (whether a car or a bus or a non-motorized vehicle) as well as to predict driving activity (whether stopping or going straight or turning left

or turn right); for vehicle mode it uses only accelerometer data, thus saving on energy, while for driving activity it uses a combination of sensors– accelerometer, gyroscope, and magnetometer. During pre-processing of the data, the optimal data-window is obtained, which is then fed into various ML-architecture available with Weka tool. The author shows that accuracy is best for Random Forest model.

3 Activity Recognition

Human Activity Recognition is pervasive today, Predicting human body state – sleeping, sitting idle, walking, running, cycling etc are all example of human activity recognition [4]. There are some good dataset and pre-trained model as well on Kaggle. Once such dataset that was used during the experiment comes from `https://github.com/Kaggle/kaggle-api`. Which is dataset on a person walking or running.

3.1 Walk and Run

A single data point in time series doesn't carry much information. A repeating pattern has to be found, a time-window and its corresponding data-window, should be fed into the neural network. A typical hand-movement while walking last around 1-2 second, therefore a time-frame of 2 second was chosen. With 6 samples/seconds each data-frame will contains 12 samples. The dataset was contributed by Viktor and there is a very nice article by him on medium, which can provide a better insight. It was collected for both wrist in a burst of 10 second after idle period of 10 second.

4 Setting up your Raspberry Pi

We have used Python 3.5, and tensor-flow build 1.8.0.

4.1 Creating Virtual Environment for Python

virtual environment helps maintain separate libraries for python3 from system installed libraries. Thus avoiding possible conflict with python 2.7. It's a good thing to work on virtual environment, which could possibly save you from future trouble.

```
## creating virtual environment
$ sudo apt install python3-dev
```

```
$ sudo apt install python3-pip
$ sudo apt install libatlas-base-dev
$ sudo pip3 install -U virtualenv
$ virtualenv --system-site-packages -p python3 ./venv_tf_1p8
$ source ./venv_tf_1p8/bin/activate
$ pip install --upgrade pip
```

after activating python 3.5 virtual environment we can continue to install other dependencies.

4.2 Installing TensorFlow build for Raspberry

We will take tensorflow build available for raspberry pi.

1. for pi3 use:

```
wget https://github.com/lhelontra/tensorflow-on-arm/releases/
download/v1.8.0/tensorflow-1.8.0-cp35-none-linux_armv7l.whl
```

2. for pi zero use:

```
wget https://github.com/lhelontra/tensorflow-on-arm/releases/
download/v1.8.0/tensorflow-1.8.0-cp35-none-linux_armv6l.whl
```

for our experiment we have used pi zero. while using virtual environment we can call upon sudo command, which applies system wide.

```
(venv_tf_1p8) $ wget https://github.com/lhelontra/
  tensorflow-on-arm/releases/ download/v1.8.0/tensorflow
  -1.8.0-cp35-none-linux_armv6l.whl
(venv_tf_1p8) $ source venv_tf_1p8/bin/activate
(venv_tf_1p8) $ pip install tensorflow-1.8.0-cp35-none-
  linux_armv6l.whl
(venv_tf_1p8) $ pip uninstall mock
(venv_tf_1p8) $ pip install mock
(venv_tf_1p8) $ sudo apt-get install python3-numpy
(venv_tf_1p8) $ sudo apt-get install libblas-dev
```

```
(venv_tf_1p8) $  sudo apt-get install liblapack-dev
(venv_tf_1p8) $  sudo apt-get install python3-dev
(venv_tf_1p8) $  sudo apt-get install libatlas-base-dev
(venv_tf_1p8) $  sudo apt-get install gfortran
(venv_tf_1p8) $  sudo apt-get install python3-setuptools
(venv_tf_1p8) $  sudo apt-get install python3-scipy
(venv_tf_1p8) $  sudo apt-get update
(venv_tf_1p8) $  sudo apt-get python3-h5py
(venv_tf_1p8) $  sudo apt-get install python3-h5py
(venv_tf_1p8) $  pip install keras
(venv_tf_1p8) $  sudo apt-get install python3-pandas
(venv_tf_1p8) $  sudo apt-get install python3-sklearn
```

5 Latency with CFS and RT-patch scheduler on Raspberry Pi 0

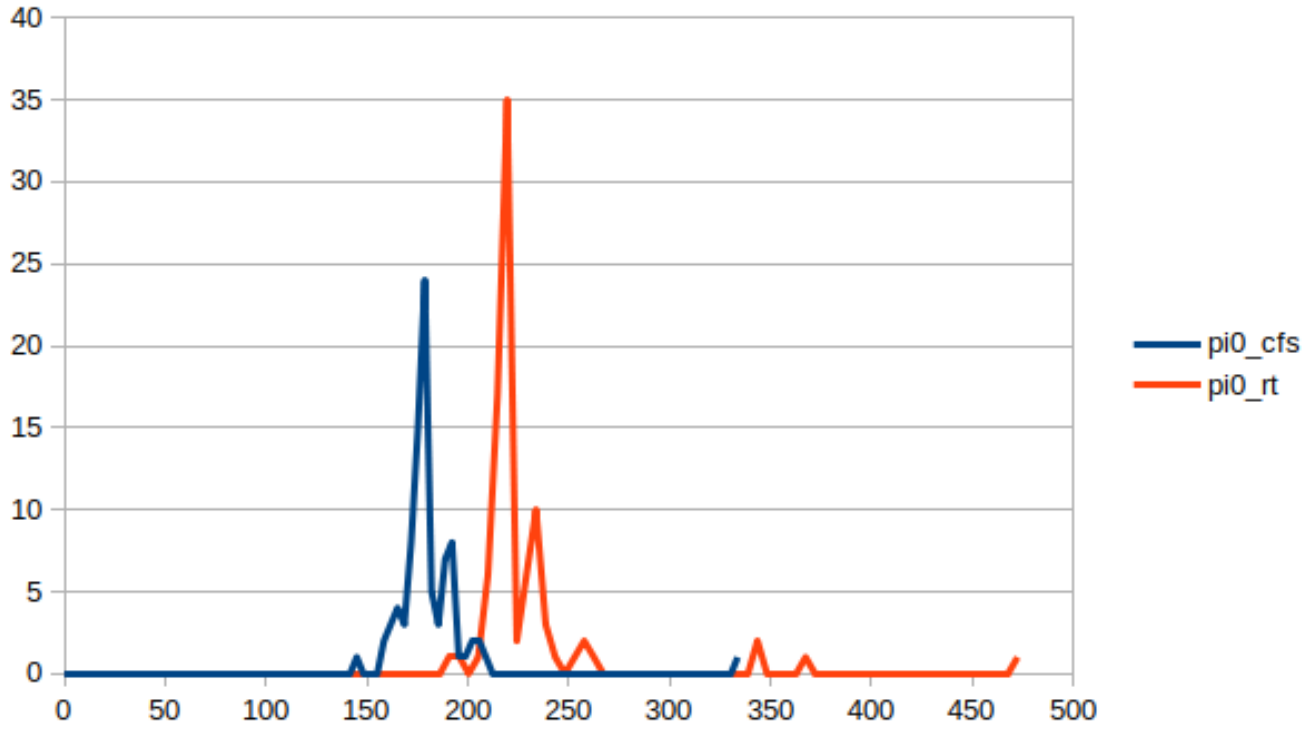

Figure 1: histogram of latency for CFS-scheduler vs RT-scheduler carried on Pi0

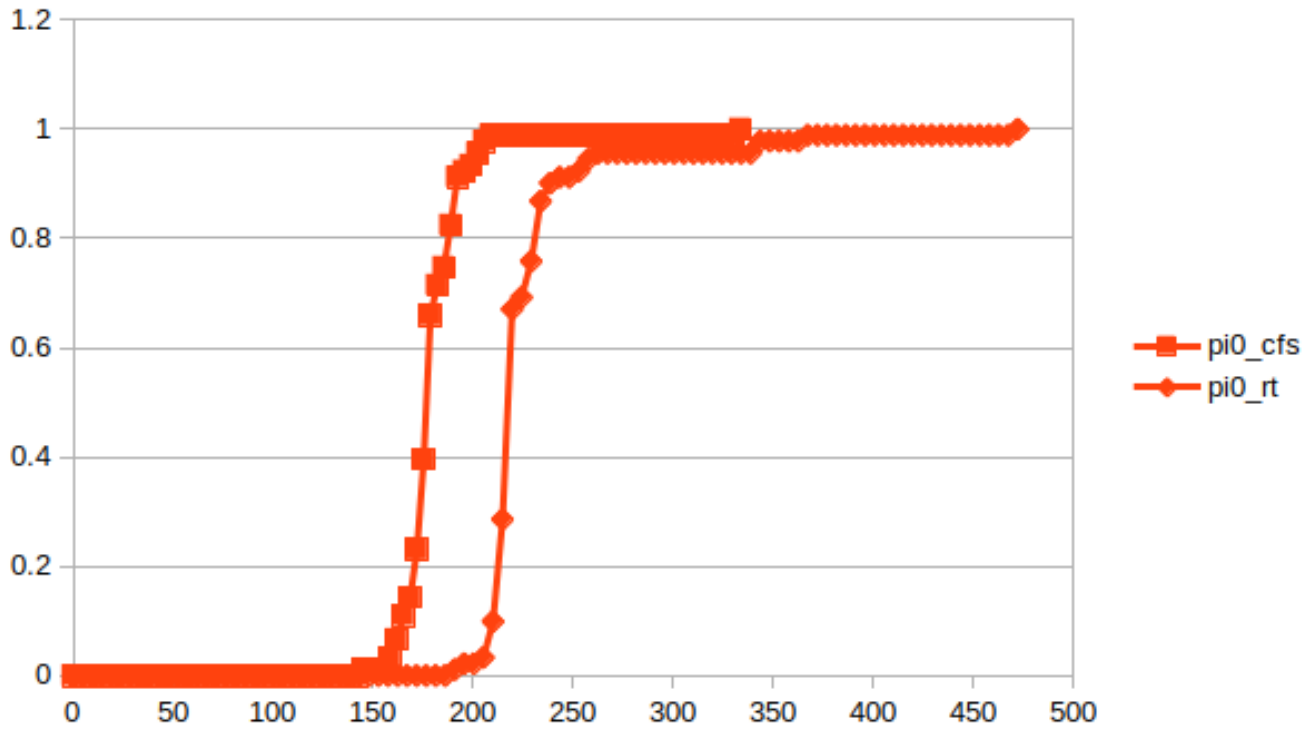

Figure 2: CDF of latency for cfs vs rt carried on Pi0

6 Latency with CFS and RT-patch scheduler on Raspberry Pi 3

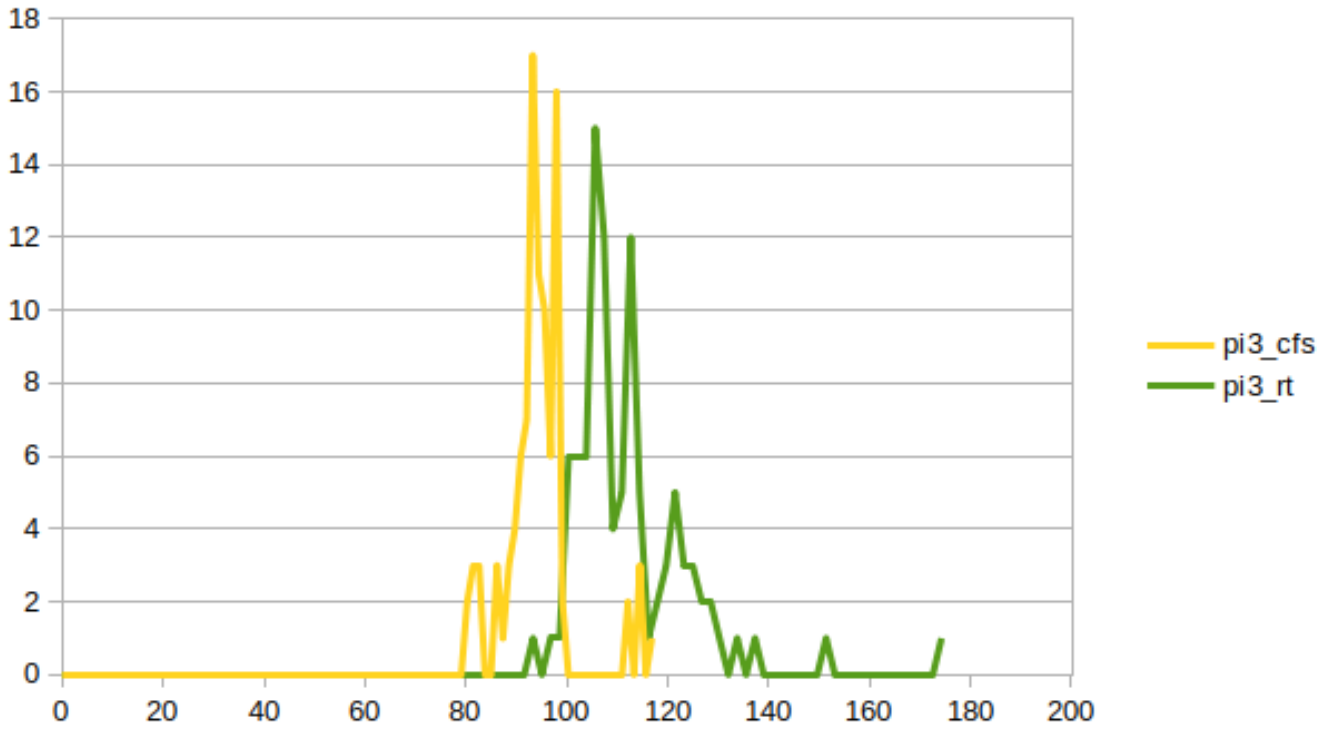

Figure 3: histogram of latency for CFS-scheduler vs RT-scheduler carried on Pi3

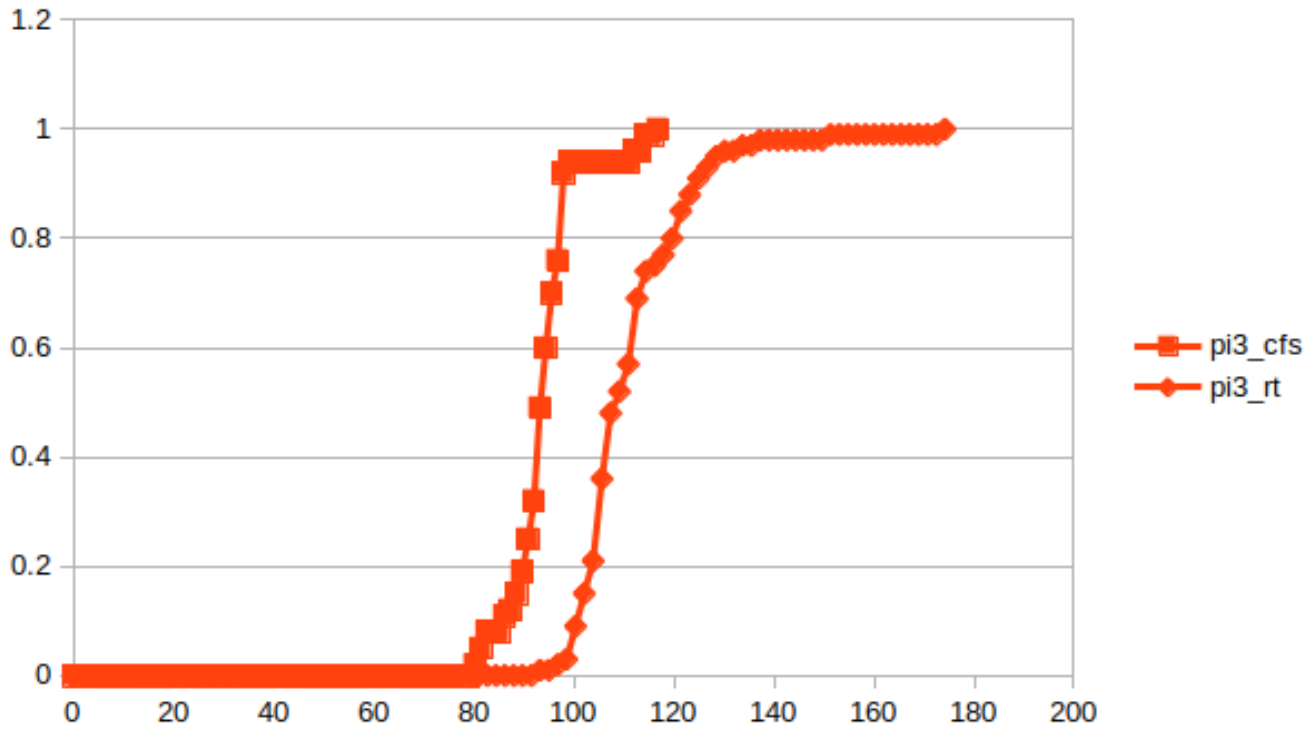

Figure 4: CDF of latency for cfs vs rt carried on Pi3

7 Conclusion

Cumulative distribution of latency shows steep transition, prominently appearing for Pi0, Figure 2. Indicating that rt-scheduler brings some predictability, giving a tighter latency window. This has been observed on both platform Raspberry Pi0, Figure 2 and Raspberry Pi3, Figure 4. The table below, however seems to say otherwise, standard deviation is higher for rt-scheduler on both platform. With a larger data-set, will take weeks of running this experiment, we can have a better latency window measurement and better insight.

	cfs	rt-patch
Mean	183	231
Std	20	37

Table 1: Statistics on Pi0 for cfs and rt-patch

	cfs	rt-patch
Mean	95	113
Std	7	11

Table 2: Statistics on Pi3 for cfs and rt-patch

References

[1] Huseyin Eren, Semiha Makinist, E Akin, and A Yilmaz. Estimating driving behavior by a smartphone. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 234–239, 06 2012.

[2] Sara Hernández Sánchez, Rubén Fernández Pozo, and Luis A. Hernández Gómez. Estimating vehicle movement direction from smartphone accelerometers using deep neural networks. *Sensors*, 18(8), 2018.

[3] Dang-Nhac Lu, Duc-Nhan Nguyen, Thi-Hau Nguyen, and Ha-Nam Nguyen. Vehicle mode and driving activity detection based on analyzing sensor data of smartphones. *Sensors*, 18(4), 2018.

[4] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1729–1734. AAAI Press, 2011.