Benchmarking Deep Classifiers on Mobile Devices for Vision-based Transportation Recognition

Sebastien Richoz
sr569@sussex.ac.uk
University of Sussex
Brighton, United Kingdom

Andres Perez-Uribe
Andres.Perez-uribe@heig-vd.ch
Uni. of Applied Sciences Western Switzerland (HEIG-VD)
Yverdon, Switzerland

Philip Birch
p.m.birch@sussex.ac.uk
University of Sussex
Brighton, United Kingdom

Daniel Roggen
daniel.roggen@ieee.org
University of Sussex
Brighton, United Kingdom

ABSTRACT

Vision-based human activity recognition can provide rich contextual information but has traditionally been computationally prohibitive. We present a characterisation of five convolutional neural networks (DenseNet169, MobileNet, ResNet50, VGG16, VGG19) implemented with TensorFlow Lite running on three state of the art Android mobile phones. The networks have been trained to recognise 8 modes of transportation from camera images using the SHL Locomotion and Transportation dataset. We analyse the effect of thread count and back-ends services (CPU, GPU, Android Neural Network API) to classify the images provided by the rear camera of the phones. We report processing time and classification accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Activity recognition and understanding**; • **Theory of computation** → *Discrete optimization*; • **Software and its engineering** → Designing software.

KEYWORDS

Human activity recognition; embedded classifier; embedded convolutional neural network; deep learning; smartphone characterization.

UbiComp/ISWC '19 Adjunct, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6869-8/19/09…$15.00
https://doi.org/10.1145/3341162.3344849

ACM Reference Format:
Sebastien Richoz, Andres Perez-Uribe, Philip Birch, and Daniel Roggen. 2019. Benchmarking Deep Classifiers on Mobile Devices for Vision-based Transportation Recognition. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2019 International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3341162.3344849

1 INTRODUCTION

The mode of transportation and locomotion of users - such as whether a user is still, walking, running, cycling, driving a car, taking a bus, a train or a subway - is an important information to provide contextual services on mobile devices. This knowledge could assist context-aware applications such as health monitoring, parking spot detection or content delivery optimization.

A user often carries a wearable device (e.g smartphone, smartwatch) during travel, which is embedded with multimodal sensors including motion sensors, GPS (global positioning system), microphone and camera. While most work on locomotion and transportation recognition has used motion sensors [1, 2, 8–12, 14] or sound [4, 5, 13], our recent work has shown that vision is also an important modality to recognise modes of locomotion and transportation [6]. We showed that 8 activities (Still, Walk, Run, Bike, Car, Bus, Train, Subway) can be recognised with an F1 score of 82.1% for the best classifier.

Vision becomes particularly interesting as embedding camera is gaining traction: faster and smaller processing units and improved hardware implementations have made this modality available in much more devices including smart glasses, smartwatches and action cameras. Due to privacy issues and power consumption, however, processing has ideally to be done locally.

This paper provides a benchmark of deep learning algorithms for vision-based locomotion and transportation mode

recognition on smartphone devices. Our contributions are the following:

- We introduce the large scale Sussex-Huawei Locomotion-Transportation (SHL) dataset, which is used to train machine learning models.
- We summarize five classifiers - DenseNet169, MobileNet, ResNet50, VGG16, VGG19 - and their characteristics, which we identified from our previous work and which we have trained on the SHL dataset.
- We present an Android mobile application based on Tensorflow Lite framework that embeds these classifiers and which is wrapped by our performance evaluation tool.
- We benchmark the system on 3 phones: Huawei Mate 9, OnePlus One, Samsung Galaxy S9. We analyse the computation time in function of the thread count (from 1 to 8) and computing back-end (CPU, GPU, NNAPI).
- We discuss the impact of classifiers size from the benchmark results and the feasibility of deploying such deep learning architectures on constraint mobile environments.

2 DATASET

The Sussex-Huawei Locomotion-Transportation (SHL) dataset is one of the biggest multimodal dataset for transaportation and locomotion mode recognition [3]. It was recorded over 7 months by 3 users and includes 8 different transportation modes: Still, Walk, Run, Bike, Car, Bus, Train and Subway. As a result, the dataset contains 16 sensors modalities including motion, GPS (global positioning system), sound and image. The images were recorded with a body-worn camera mounted on the chest and facing forward. Fig. 1 shows a sample of the images available in SHL dataset, capturing one image every 30 second. In total, 14600 images out of 86075 were used.

3 CLASSIFIERS

In our previous work [6], we selected five convolutional neural networks as deep learning classifiers with a variety of architecture, size and number of parameters, which were pre-trained on ImageNet [7]. Fig. 2 illustrates the generalized pipeline for the five classifiers, as each architecture can be expressed in term of blocks, where each block is made of convolutional, max-pooling and dropout layers. We modified the original architectures by replacing their final output layers with one global average pooling (GAP) layer to reduce the number of parameters, one fully connected layer of 512 neurons (FC_1) and one fully connected layer of 8 neurons with softmax activation (FC_2) to predict the eight transportation modes. Then, we optimized them for the SHL dataset

Figure 1: SHL images for each transportation mode.

by applying transfer learning on all or part of their architecture. The original input image of size 1024x576 pixels is preprocessed into a 224x224 pixels square image and fed to the classifiers which infer specific features of the image to recognise the transportation mode. More processing details can be found in [6].

Mobile implementation

In order to implement the classifiers in the Android application presented in Section Mobile Application, each classifier must be converted in the appropriate format. The classifiers stored as a Keras HDF5 model are converted with the TensorFlow Lite Converter tool into an optimized 'FlatBuffer' format, so that they can be interpreted by the TensorFlow Lite framework. 'FlatBuffer' is an efficient serialization library originally created by Google for performance-critical applications like games. After conversion, the classifiers are stored as '.tflite' file. Table 1 describes the characteristics of the classifiers and compares the sizes between a HDF5 file and an optimized FlatBuffered file.

4 MOBILE APPLICATION

We sought to characterize the classifiers on a constrained yet realistic environment. Smartphones provide powerful computational resources and enough memory storage to embed the classifiers in. We developed an Android application

Table 1: Characteristics of the CNNs. HDF5 and FlattBuffer are encoding libraries. The values within these columns refer to the size of the file encoded with. A HDF5 file is converted into a FlatBuffer with the TensorFlow Lite Converter tool. Depth refers to the maximum depth of the classifiers calculated as the maximum number of kernels for all convolutional layers. CPU, NNAPI and GPU are the median classification time recorded across the 3 phones with 8 threads.

Classifier	*SHL F1-score [6]*	*HDF5*	*FlatBuff*	*#Param.*	*Depth*	*Layers*	*Blocks*	*CPU*	*NNAPI*	*GPU*
DenseNet169	76.2 ± 0.2 %	67 MB	53 MB	14.3 k	169	595	5	490 ms	1695 ms	221 ms
MobileNet	77.0 ± 0.4 %	**27 MB**	**15 MB**	4.3 k	88	87	13	**71 ms**	172 ms	**72 ms**
ResNet50	79.1 ± 0.5 %	172 MB	98 MB	25.6 k	152	168	16	319 ms	**145 ms**	349 ms
VGG16	81.4 ± 0.9 %	120 MB	60 MB	138.4 k	23	19	5	1468 ms	2481 ms	1840 ms
VGG19	**82.1 ± 0.2** %	162 MB	81 MB	143.7 k	26	22	5	1524 ms	3135 ms	2528 ms

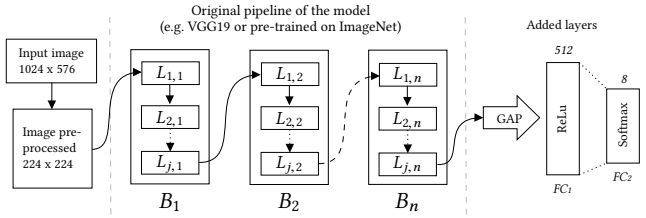

Figure 2: Generalized Pipeline. Each classifier has B blocks which contains j layers $L_{j,n}$ according their architecture (see Tab. 1)

based on Tensorflow Lite [1] with tensorflow-lite version '0.0.0-gpu-experimental' to provide GPU support and the Android Neural Network API (NNAPI) on top of the CPU.

Fig. 3 shows a screenshot of the application which let us select the number of threads, the classifier and the backend. The classifier continuously classify the feed of images provided by the rear camera. For each classification, the ranking of the 8 activities is displayed with the corresponding prediction and the time taken to perform the classification. The application was developed on Android Studio 3.4.1 in Kotlin 1.3.40 and targets Android API levels from 21 (Lollipop, 5.0) to 28 (Pie, 9.0). It is publicly available on the Github repository of the Sensor Technology Research Centre [2].

The TensorFlow Lite framework provides delegates, such as the GPU, which allow running all or part of the classifier on dedicated hardware of the mobile device. To reduce the workload on the CPU, the framework will delegate simple yet repetitive calculations, such as 2D convolutions, to the GPU. As a result, a gain of computation time is expected. On our application, if 'CPU' device is selected, then all the classification will run on the CPU only.

The Android Neural Network API (NNAPI) is an Android C API available on all devices running Android 8.1 (API level 27) or higher. It is designed for running computationally intensive machine learning classifiers on mobile devices [3]. Based on the hardware capabilities of the device, it will efficiently distribute the workload of classification across available hardware such as neural networks hardware, GPUs, and digital signal processors (DSPs).

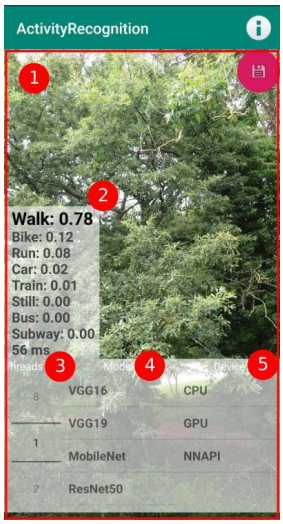

Figure 3: Mobile Application. Legend: 1 = Image to classify, 2 = Predicted classes with score and duration of classification, 3 = Number of threads, 4 = Classifier, 5 = Device.

Experiments

To characterize the classifiers, we observe the effect of thread count from 1 to 8 and backend variation between CPU, NNAPI and GPU across three smartphones described in Table 2. The Android application is installed on each phones and they are set on airplane mode for performance purposes. The time to classify one image is recorded and saved in a file. The battery consumption is recorded but not used as the phones were charging during the whole experiment to avoid running out of power. Each classifier performs the

[1] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/java/demo

[2] https://github.com/sussexwearlab/SHLVisionApp

[3] https://developer.android.com/ndk/guides/neuralnetworks

classification task over a minimum of 60 seconds for a given thread count and device. We ensured to classify different images by facing the camera of the phones towards dynamic videos containing a variety of transportation modes.

5 RESULTS AND DISCUSSION

Fig. 4 shows the result of the experiment. The median classification times for each classifier and device are indicated in Table 1. We report the main results as follows:

- Globally, MobileNet is the fastest classifier and VGGs are the worst, despite they achieve the best classification on SHL.
- The GPU of the Huawei Mate 9 reduces drastically the classification time and outperforms the CPU and NNAPI.
- The CPU of the Samsung Galaxy S9 is as fast as the GPU of the Huawei Mate 9 (for this specific task).
- Increasing thread count from 1 to 4 reduces the classification time for the Huawei Mate 9 and Samsung Galaxy S9 for all classifiers except DenseNet169 and MobileNet. The OnePlus One does not perform better with more threads. The computational time stays the same or increases with thread count after 5 threads, possibly because all phones have only 4 high-performance cores (S9 and Mate have 4 high-performance and 4 low-powered).
- The NNAPI achieves the same computation time as the CPU for the Huawei and OnePlus. For the Samsung, NNAPI is faster than the CPU only for ResNet50. For the remaining classifiers, the CPU is faster than NNAPI, probably because of the overhead generated from the split of the computational workload across the processing units.

We expected faster computation time with the NNAPI but overheads might have been introduced due to the distribution of the computational workload which might penalize lightweight models like MobileNet. NNAPI is probably not ready for prime time, latest mobile devices are just embedding the hardware dedicated to neural networks. Overall, we noticed high battery decrease even though the phones were in airplane mode and all other applications were closed. Most of concurrent operations were then reduced but some background tasks might still perform.

6 CONCLUSION

In this work, we compared the classification time of five convolutional neural networks across three smartphones and noticed that MobileNet is the fastest on mobile devices with an average classification time of 71 ms using the CPU and 72 ms with the GPU for an input image size of 1080x1920 pixels. The NNAPI achieved the second fastest classification time on ResNet50. The GPU reduces significantly the classification time of all classifiers on the Huawei Mate 9. The experimental GPU version of TensorFlow Lite caused few crashes with VGG networks on the Samsung Galaxy S9 and on the OnePlus One.

In the future, we might consider different optimization for the GPU by changing the floating precision from float-32 to float-16 or by using 8-bit quantization to approximate the float value on a 8-bit integer. Therefore, we might reduce the computational time as more operations could be done at a time. Also, most of the new hardware implemented in recent mobile devices integrates SIMD instructions which allows to process multiple mathematical operations on a single instruction, which could as well reduce the classification time but might also reduce the accuracy of the classifier. Comparing different versions of Android on a single phone will provide indication on the software-side impact.

REFERENCES

[1] S. Fang, Y. Fei, Z. Xu, and Y. Tsao. 2017. *IEEE Sensors Journal* 17, 18. 6111–6118.

[2] T. Feng and H.J.P. Timmermans. 2013. *Transportation Research Part C: Emerging Technologies* 37. 118 – 130.

[3] H. Gjoreski, M. Ciliberto, L. Wang, F. J. Ordonez Morales, S. Mekki, S. Valentin, and D. Roggen. 2018. *IEEE Access* 6. 42592–42604.

[4] S. Lee, J. Lee, and K. Lee. 2017. VehicleSense: A reliable sound-based transportation mode recognition system for smartphones. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 1–9.

[5] H. Lu, J. Yang, Z. Liu, N.D Lane, T. Choudhury, and A.T Campbell. 2010. The Jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM conference on embedded networked sensor systems*. ACM, 71–84.

[6] S. Richoz, M. Ciliberto, L. Wang, P. Birch, H. Gjoreski, A. Perez-Uribe, and D. Roggen. 2019. Human and machine recognition of transportation modes from body-worn camera images.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. 2015. *International Journal of Computer Vision (IJCV)* 115, 3. 211–252.

[8] P. Nurmi S. Hemminki and S. Tarkoma. 2013. Accelerometer-based Transportation Mode Detection on Smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*. ACM, New York, NY, USA, Article 13, 14 pages.

[9] P. Siirtola and J. Röning. 2012. *Int. J. of Interactive Multimedia and Artificial Intelligence* 1. 38–45.

[10] X. Su, H. Caceres, H. Tong, and Q. He. 2016. *IEEE Transactions on Intelligent Transportation Systems* 17, 10. 2921–2934.

[11] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen. 2018. Benchmarking the SHL Recognition Challenge with Classical and Deep-Learning Pipelines *(UbiComp '18)*. ACM, New York, NY, USA, 1626–1635.

[12] L. Wang, H. Gjoreskia, K. Murao, T. Okita, and D. Roggen. 2018. Summary of the Sussex-Huawei Locomotion-Transportation Recognition Challenge *(UbiComp '18)*. ACM, New York, NY, USA, 1521–1530.

[13] L. Wang and D. Roggen. 2019. Sound-based Transportation Mode Recognition with Smartphones. 930–934.

[14] H. Xia, Y. Qiao, J. Jian, and Y. Chang. 2014. *Sensors (Basel, Switzerland)* 14. 20843–20865.

Table 2: Characteristics of the smartphones.

Phone	*Model*	*Android*	*CPU*	*Number of cores*	*RAM*	*GPU*
Huawei Mate 9	MHA-L29	7.0 (24)	AArch64 2.4 GHz	8 (4+4)	4 GB	ARM Mali-G71
OnePlus One	A0001	6.0.1 (23)	ARMv7 2.4 GHz	4	3 GB	Qualcomm Adreno (TM) 330
Samsung Galaxy S9	SM-G960F	9.0 (28)	ARMv8 2.7 GHz	8 (4+4)	4 GB	ARM Mali-G72

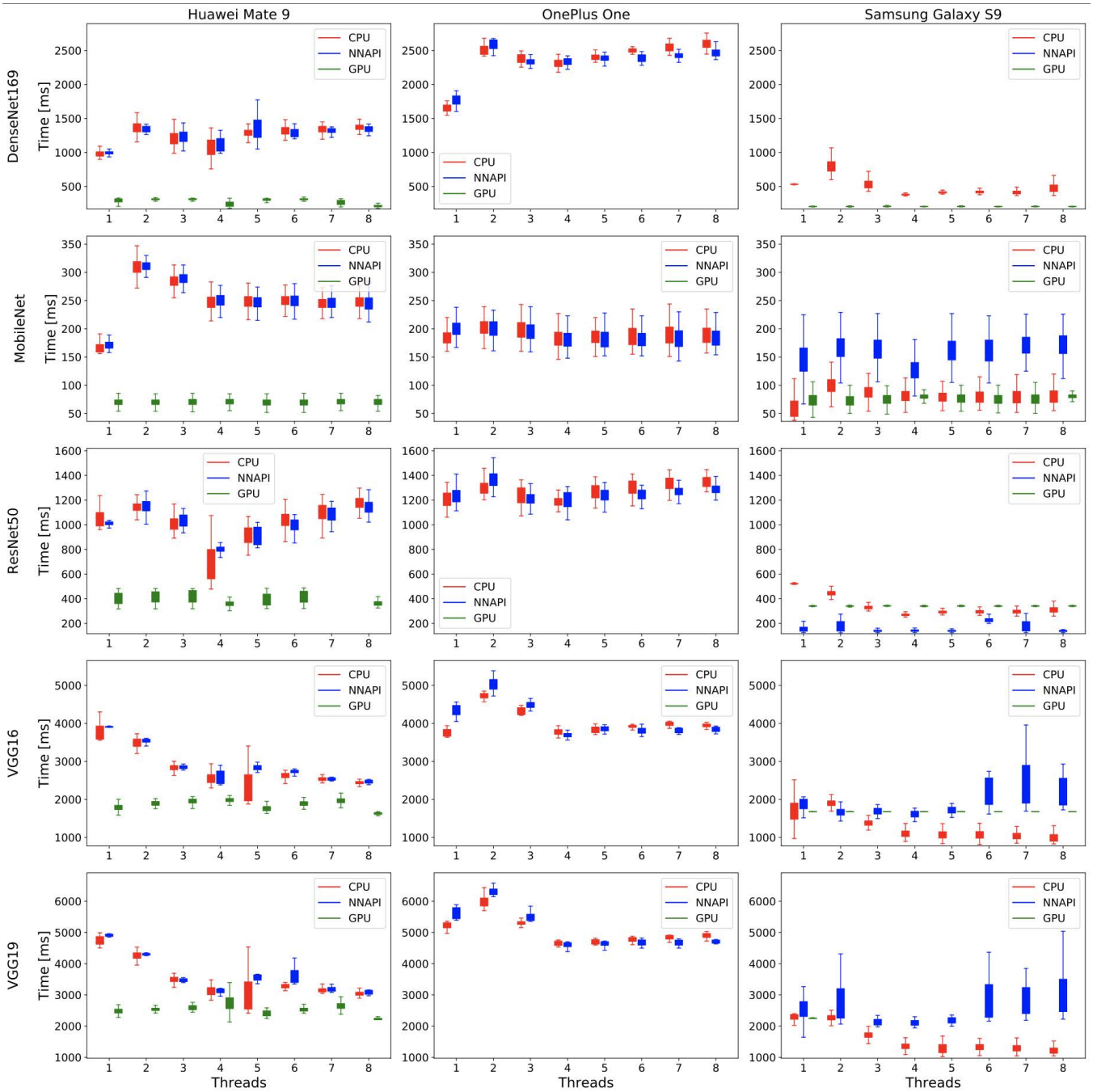

Figure 4: Results of the experiment. Some records are missing because the application crashed for these specific parameters.