Are Android Smartphones Ready to Locally Execute Intelligent Algorithms?

M. Ricardo Carlos[1], Fernando Martínez[1(✉)], Raymundo Cornejo[2], and Luis C. González[1]

[1] Facultad de Ingeniería, Universidad Autónoma de Chihuahua, 31125 Chihuahua, Mexico
ricardo.carlos@gmail.com, {fmartine,lcgonzalez}@uach.mx
[2] CONACYT, Universidad Autónoma de Chihuahua, Chihuahua, Mexico
rcornejoga@conacyt.mx

Abstract. Given that thousands of applications are already available for smartphones, we may be inclined to believe that ubiquitous computing is just around the corner, with online processing in these mobile devices. But, how well prepared is current smartphone technology to support the execution of demanding algorithms? Surprisingly, few researchers have addressed the processing capabilities of currently available smartphones. In this paper we investigate some issues in this direction: we employed twelve algorithms for optimization and classification to profile the computational demands they place on current smartphones. For this purpose, we chose twelve devices that go from low to high-end models, from six different makers, and measured execution time, CPU and RAM usage while the devices were running the algorithms.

Keywords: Mobile computing · Optimization algorithms · Classification algorithms · Profiling

1 Introduction

The number of mobile devices in operation has continuously increased during the last decade. According to International Data Corporation (IDC), 337.2 million smartphones were shipped worldwide during the second quarter of 2015. This figure is 11.6% higher than the sales from the second quarter of 2014 [1], and high sales are expected in the future, since short smartphone replacement cycles have been observed worldwide among consumers [2]. United States citizens have short replacement cycles, replacing their smartphones after one year and nine months (while Mexican citizens have longer replacement cycles, replacing their smartphones after three years and three months). Shorter replacement cycles allow consumers to have access to better and faster storage, computing, and sensing capabilities with this type of devices, frequently upgraded by manufacturers. Therefore, these shorter cycles have increased the available computing and sensing capabilities for the mobile device market.

O. Pichardo-Lagunas and S. Miranda-Jiménez (Eds.): MICAI 2016, Part II, LNAI 10062, pp. 15–25, 2017.
DOI: 10.1007/978-3-319-62428-0_2

The integration of sensing technologies (with accelerometers, gyroscopes, magnetometers, GPS, light, and proximity sensors, among others) enables smartphones to see, hear, and feel the user's environment. Therefore, the availability of the information provided by these sensors creates opportunities to infer different types of events for context-awareness, and to create applications for end users, third parties, crowds and groups [3].

Recent research has explored learning algorithms running in smartphones to represent, to some extent, the end users' activities [4]. Inferring and representing the user's activity might have positive implications in different domains such as fitness tracking, health monitoring, fall detection for vulnerable populations, context-aware behavior, and home and work automation [3]. Furthermore, activity recognition has been also explored in businesses and organizations where it is a valuable source of information for decision making processes. Companies have used activity recognition to present targeted advertisements, which are more relevant to their customers' preferences and activities. Similarly, activity recognition can assist with employee management and accounting for employee time. For example, health care companies can track the activities of their employees and improve their critical health processes, based on conflicting or time consuming activities. In crowd and group contexts, activity recognition has been used to track and analyze big data sets to automate tagging events such as traffic, places, emergency events or disasters. Although activity recognition is widely implemented in these three major types of application, only end user applications implement some algorithms in smartphones to recognize activities. Businesses and crowd work applications rely on back-end systems to perform heavy computing algorithms to collect and analyze data.

Machine learning (ML) can be well suited to treat problems of context-awareness for businesses and crowd work. ML capabilities can automatically adapt to the changes in input data created by the user, the available sensors, or environmental factors. Machine learning can also be used to detect malware in smartphones [5], and protect these devices from intrusions. Employing algorithms of this nature has been discussed in the literature, and pre-trained models are already available and deployed [6], but their actual viability to be trained in mobile devices, under normal operating conditions, has not been addressed. Furthermore, it is unclear whether if context-aware mobile applications can be supported with ML algorithms with online processing.

In this paper we address the following relevant open question: are current Android smartphones capable of handling the computational demands of these algorithms?[1] It is our understanding that, frequently, it's taken for granted that smartphones cannot deal with demanding computing tasks, which are left for server side applications. Our contribution in this work is to show how mobile phones perform when classifiers and optimization algorithms are run, to have a better perspective of what smartphones' current capabilities are, and provide some guidance in the implementation of such computing tasks.

[1] We chose the Android platform because it represents over 80% of the market share, and this preponderance is expected to remain in the near future [7].

2 Related Work

Mobile phones have evolved extensively, in so manner that today these are not just telephones but multimedia devices and computing nodes with high capabilities. High-end mobile phones have system-on-a-chip (SoC) embedded systems with multi-core ARM processors, with clocks running in the order of GHz, and at least 2 GB of memory. With high performance, low-power, general purpose CPUs, users now expect near PC-like performance and a rich user experience, including high-definition audio and video, high-quality multimedia, dynamic web content, responsive user interfaces, and 3D graphics. In order to meet user expectations, mobile devices must smoothly perform processing tasks at different levels: processor, local I/O, video, memory, cache, communication, application, and interaction with the user. In this section we review some of the benchmarking approaches that target hardware and software systems in mobile phone platforms. Then we present a few works that suggest how applications based on machine learning improve the user experience that fueled our motivation to state the question: are smartphones ready to locally execute intelligent algorithms? As far as we are aware, no previous work has reported how these algorithms perform on mobile phone architectures.

One of the current challenges in social computing is the management of large amounts of data. Indeed, the continuous increase in volume, variety, and velocity of Big Data exposes data centers to an energy utilization problem. Loghin et al. [8] explore the usage of wimpy nodes, mobile phone-like architectures, to achieve some data pre-processing which might help alleviate the storage and processing stress faced by data centers. The authors ran experiments demanding memory, server level communication, and read/write operations on the ARM big.LITTLE technology. The benchmark included Big Data frameworks such as the Hadoop Distributed File System, I/O data processing with Hadoop Map Reduce, and the Parallel Memory Bandwidth Benchmark. Overall performance of mobile phone-like architectures is around four orders below Intel Xeon-based servers. In terms of energy costs Loghin et al. found that, under low processing demands, ARM-based platforms are four times cheaper than Xeon servers but, with peak computing profiles, these servers demand 50% less energy than ARM systems.

One of the changes with the users' lifestyles is that mobile phones allow multitasking. Users can play games while chatting or listening to music, and the concurrent execution of applications increases battery energy usage. To some extent, users consider their mobile device as a repository of running applications. Pathak et al. [9] present *eprof*, a tool to profile the energy demands of applications running on smartphones. Their energy profiler accounts for power draw at hardware components, at program entities, and at the source code level. Five different free applications that require internet connectivity were ran on Android devices. The authors found that running the five applications for about half a minute can invoke 29–47 threads, 200k-6M routine calls, and that it took 0.35%–0.75% of the battery charge. The web browser was also tested using a

Google search, and GPS was used to determine user location. This activity consumes 2000 μAH, distributed among CPU (53%), 3G (31%), and GPS (16%).

The academic interest in studying how energy stored in the batteries of mobile phones is used comes from the direct impact this has on the user's experience. In fact, as demonstrated by Carrol and Heiser [10], the hardware components and services the user interacts with the most are the ones that put the most pressure on battery life. Using micro-benchmarks, the authors characterize power consumption at the system level. The Openmoko Neo Freerunner open source platform allowed them to take physical measurements directly on testing points, available on the main board. Resistors were inserted to measure current, and both current and voltage measurements helped calculate the power demands of CPU, memory, touchscreen, graphics hardware, audio, storage, and various networking interfaces. Their tests integrate such diverse functionality as voice communication, audio and video playback, web browsing, SMS, and email communication, media downloads, and gaming. Their findings indicate that energy is spent the most in Video playback (453 mW), GPS (143 mW), GSM call (1054 mW), SMS (302 mW), email GPRS (610 mW), and email Wifi (432 mW). This level of rich functionality increases the pressure on battery life, and deepens the need for effective energy management.

It can be realized that mobile system designers and computer architects are aware of computing capabilities and processing resources that must allow users to use the smartphone as they wish. Gutierrez et al. [11] considered relevant to measure the performance of interactive applications that are commonly used by users. Representative applications for streaming HD video, gaming, playing MP3 files, and browsing the web, were selected for the benchmark and results were compared against some features of the SPEC CPU2006 benchmark. Generally speaking, the authors found that smartphone applications are far from the SPEC benchmarks. They observed issues with massive application code footprints, missing instruction cache, and poor management of paged memory. An explanation for this level of poor performance of interactive applications is that most mobile applications are developed relying on high level abstractions and calls to shared libraries. This impacts the user's experience.

As illustrated, hardware and software benchmarks have been studied. Battery energy, data processing, application and code performance have been evaluated and characterized in order to inform the designers and architects of future smartphones. This helps envisage robust and high computing performance mobile applications that assist users in their everyday activities. Intelligent applications can keep low resources usage when running location-based systems that make use of the hunger energy consumers like the GSM, WiFi, and GPS sensors [12].

To the best of our knowledge, the stress machine learning algorithms put on mobile phone architectures is not documented in the literature. The next sections describe the experiments and results of running optimization and classification heuristics on smartphones, and our insights from this experience, to address this situation.

3 Experimental Setup

In order to test the processing capabilities of common smartphones, the execution times, memory, and CPU usage were evaluated for seven optimization algorithms and five classifiers, in twelve Android smartphones. The optimization algorithms were run to solve common simple and multi-objective test problems, and the tested classifiers were trained under supervised learning. Each algorithm was run twelve times on each smartphone. Only one CPU core was employed to run the algorithms.

The Android platform was selected because it has represented about 80% of the smartphones in use in the last years [13]. We tested ten mid-range Android devices with up to three years of use, currently the primary phones of their respective owners, as a baseline for the smartphones currently in service. We also a included a more recent high-end model (Samsung S6) and a six years old device (T-Mobile MyTouch), to get an idea of the differences that could be expected for newer and older models. The main characteristics of these devices are summarized in Table 1.

To put in context the capabilities of the smartphones, the same suite of tests was run on a Toshiba P55W-B5224 laptop with 16 GB of RAM and an Intel i7-4510U CPU, using a Java 1.8.0 run-time on a Ubuntu Gnu/Linux x86_64 system (kernel 3.13.0-63-generic SMP), running the algorithms in only one core.

We acknowledge the lack of control in the software installed in the smartphones, and that it can have a significant influence when performing benchmarks. However, the heterogeneity of operating systems, runtimes, and installed applications are inherent to the Android platform. The smartphones had different applications installed, since they were the primary mobile phone for their owners. The manufacturer's kernel and run-time environment were kept in all the smartphones.

Table 1. Characteristics of the smartphones used.

Device	Model	OS	Runtime	CPU cores	RAM	Chipset
1	LG Nexus 4	5.1.1	ART 2.1	Quad 1.5 GHz	2 GB	Qualcomm APQ8064
2	Sony D5316	5.0.2	ART 2.1	Quad 1.4 GHz	1 GB	Qualcomm MSM8928
3	Motorola Moto G	4.4.4	Dalvik 1.6	Quad 1.2 GHz	1 GB	Qualcomm MSM8226
4	Zuum P60	4.2.2	Dalvik 1.6	Quad 1.3 GHz	1 GB	MediaTek MT6582
5	LG D680	4.4.2	Dalvik 1.6	Dual 1.0 GHz	1 GB	Mediatek MT6577
6	Samsung SM-G925I	5.0.2	ART 2.1	Octa 2.1 GHz	3 GB	Samsung Exynos 7420
7	Samsung SM-N900V	4.4.4	Dalvik 1.6	Quad 2.27 GHz	3 GB	Qualcomm MSM8974
8	Samsung SGH-I337M	5.0.1	ART 2.1	Quad 1.89 GHz	2 GB	Qualcomm APQ8064AB
9	Motorola Moto G	5.1.0	ART 2.1	Quad 1.2 GHz	2 GB	Qualcomm MSM8226
10	T-Mobile MyTouch	2.3.4	Dalvik 1.4	Single 1.0 Ghz	768 MB	Qualcomm MSM8255
11	ZTE Blade L3 Plus	4.4.2	Dalvik 1.6	Quad 1.3 Ghz	1 GB	Mediatek MT6582
12	LG Nexus 5	5.1.1	ART 2.1	Quad 2.27 Ghz	2 GB	Qualcomm MSM8974

4 Optimization Algorithms

Four multi-objective algorithms, NSGA-II, a steady-state version of NSGA-II, SPEA2 and PAES, were used to solve the Kursawe test problem with three variables, employing a representation of the chromosome of type Real and a population of one hundred individuals. Crossover, mutation, and selection operations were performed randomly. The maximum number of evaluations was set to 25,000 for all multi-objective algorithms.

Three single-objective algorithms, PSO, Differential Evolution, and Evolution Strategy, were used to solve the Sphere problem with twenty variables. The default parameters used in the framework were kept for each these algorithms: for PSO, the swarm size was set to 50, mutation operations were performed, and was run for a maximum of 5,000 iterations; for Differential Evolution, the population size was 100, crossover and selection operations were performed, and the maximum of evaluations was set to 1,000,000; an elitist Evolution Strategy was evaluated for a maximum of 20,000 times, with $\mu = 1$ and $\lambda = 10$, using bit flip.

These algorithms and the test functions solved with them are frequently found in the literature, and were chosen as a representative sample of the type of calculations performed when working with optimization problems.

The algorithm implementations were provided by the 4.5 version of the jMetal framework, [14]. This framework was chosen because it offers readily available Java implementations of an array of both single and multi-objective optimization and test problems, with Java being the main programming language for the Android platform.

4.1 Results

The average execution times for each algorithm are summarized in Table 2.

The 2015 Samsung S6 was the fastest smartphone. During the execution of the algorithms, the OS reported for it an average CPU usage under 20%. As expected, the oldest phone (T-Mobile MyTouch from 2010), was the slowest (with execution times about 15 times longer than the S6) and had an average CPU usage close to 90%. For the other phones, the CPU usage was between 40% and 60%. The user-installed software running in the background in these devices might affect these results. However, we would expect the differences between devices to remain reasonably similar if we compare all phones with factory settings, due to the hardware specifications.

Hardware is not the only significant factor for smartphone performance, the OS and runtime environment seem to also be relevant for algorithm execution. The algorithms were executed on two Moto G phones with the same hardware, but the one with Android 5.1.0 and an ART 2.1.0 runtime solved the test problems in about half the time. The Samsung SM-N900V (Android 4.4.4, Dalvik 1.6.0) and the LG Nexus 5 (Android 5.1.1 and ART 2.1.0) have the same chipset, yet the first took almost twice the time of the latter to finish the tests. The smartphones with an ART runtime clearly outperformed those with Dalvik, even when the CPU architecture was comparable or equal.

Table 2. Rounded average execution times and standard deviations for genetic algorithms, in seconds.

Device	NSGAII		SSNSGAII		SPEA2		PAES		PSO		ES		DE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	*Avg*	Std	Avg	Std	Avg	Std
1	7.7	0.1	160.6	0.7	41.5	0.5	2.2	0.1	9.4	0.2	11.5	0.3	74.9	0.4
2	8.4	0.6	179.1	1.7	45.3	1.1	2.2	0.1	10.3	0.4	11.6	1.0	55.8	1.4
3	17.6	0.8	354.4	1.0	92.1	1.0	4.8	0.4	22.1	0.4	23.0	0.8	173.6	1.0
4	20.4	0.4	432.6	9.7	110.2	2.4	5.8	0.8	27.2	0.5	27.5	1.0	198.9	1.6
5	18.5	1.5	310.5	3.6	82.9	1.4	4.7	1.0	19.5	2.0	23.6	3.8	151.5	3.2
6	**2.0**	0.0	**35.9**	1.0	**10.7**	0.6	**0.7**	0.1	**2.3**	0.1	**4.0**	0.4	**17.2**	0.3
7	14.2	0.8	232.6	10.5	60.9	1.2	3.7	0.6	15.2	0.7	20.0	1.1	167.1	0.9
8	7.9	1.0	155.7	9.7	34.7	1.1	1.7	0.1	6.2	0.3	8.7	0.5	60.4	0.8
9	9.6	0.1	206.2	0.9	52.3	0.6	2.5	0.1	12.2	0.1	13.1	0.8	65.1	0.3
10	27.7	1.1	536.8	2.2	149.1	1.7	7.4	0.6	31.1	0.9	36.4	1.8	328.1	3.4
11	18.0	0.1	394.6	1.2	104.8	0.8	4.8	0.5	25.3	0.4	23.9	1.4	150.1	0.5
12	3.8	0.1	79.4	4.8	23.2	1.2	1.2	0.1	4.2	0.1	6.5	0.3	42.7	0.3

Memory assignment was not consistent among the smartphones. The T-Mobile MyTouch had a maximum assigned memory under 6 MB, with an average around 3 MB. The S6 had a maximum of almost 50 MB, and an average around 30 MB. The values for the other phones are distributed between those two extremes, with big variations between models. These variations are attributed to differences in the Android OS and runtime versions run in each device.

On average, the laptop required only 12.63% of the time used by the fastest smartphone (with SSNSGAII having the biggest performance difference, requiring only 5.82% of the fastest time, and ES showing the smallest difference, 22.29% of the fastest time).

5 Classifiers

A similar experiment was performed with five classification algorithms. Execution times, memory and CPU usage were evaluated on the Android smartphones for twelve test runs.

The five classifiers employed were: a C4.5 decision tree; KNN, considering fifteen neighbours; a Random Forest with fifteen trees; a support vector machine with a linear kernel; and a multi-layer perceptron, with one hidden layer. The SVM implementation was provided by LIBSVM [15], JSAT [16] provided the other algorithms. These libraries were chosen because they provide Java implementations of the evaluated classifiers.

The data set employed for this experiment consisted of 500 items, labeled for five different categories. These items are histograms obtained from a bag-of-words methodology [17] to classify acceleration signals in one axis. Each item has 250 features, stored as integer values. The classifiers were trained with 60%

of the data and the other 40% was used to test the models. While the same data was employed to train and test all algorithms, it was re-tagged for two categories to be used for the MLP.

5.1 Results

The average execution times for the training of each algorithm on the tested smartphones are summarized in Table 3.

Table 3. Rounded average execution times and standard deviations for classification algorithms, in seconds.

Device	C45		KNN		RFOREST		SVM		MLP	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1	4.54	0.13	5.23	0.08	4.08	0.09	4.21	0.18	11.93	0.29
2	3.2	0.16	4.10	0.12	2.70	0.08	2.97	0.18	12.88	0.28
3	4.35	0.68	6.00	0.52	2.92	0.55	2.41	0.59	30.58	0.72
4	5.04	0.39	6.63	0.29	3.82	0.30	3.59	0.31	31.34	0.39
5	5.54	0.97	7.55	1.14	4.35	0.78	4.48	0.84	27.99	1.91
6	**1.31**	0.09	**1.58**	0.06	**1.21**	0.06	**1.17**	0.06	**6.02**	2.16
7	2.75	0.66	3.21	0.54	2.04	0.50	1.61	0.55	8.56	2.35
8	3.42	0.19	3.68	0.17	2.97	0.23	2.73	0.25	8.04	0.51
9	4.10	0.22	4.96	0.13	3.51	0.09	3.74	0.26	15.41	0.37
10	8.53	0.57	10.90	0.48	5.69	0.50	5.12	0.65	43.93	0.68
11	4.90	0.40	6.13	0.28	3.29	0.31	3.70	0.36	32.28	0.29
12	3.26	0.23	4.43	0.11	2.95	0.12	3.24	0.14	14.25	0.16

The ranking of the smartphones is similar to the one obtained for the optimization algorithms. The Galaxy Note 3 (SM-N900V) had the biggest change in ranking, going from the seventh place in the previous test to almost the top in this one. This big change might be attributed to the owner performing an emergency factory reset, removing some applications that might have been running in the background for our previous test. This noticeable change in performance puts in perspective that the impact of the diversity of conditions found in Android smartphones must not be underestimated.

Except for the SM-N900V, the devices running ART also outperformed those running Dalvik for this test. This finding, together with the performance of the newest device, shows the big advances that are being made in mobile platforms, both in terms of hardware and software.

In this test, unsurprisingly, the laptop also outperformed the smartphones. It required 14.69% of the time used by the fastest smartphone (with MLP having the biggest time difference, requiring 4.32% of the fastest smartphone time).

6 Discussion

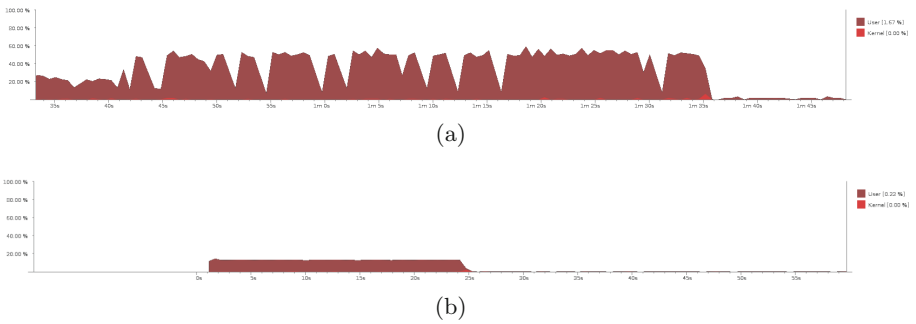
Our results show that even modest smartphones can handle low scale applications of optimization and classification tasks.

If we extrapolate execution times to big datasets, or more complex optimization problems, we might quickly run into unacceptable time scales. The current workflow of running the intensive calculations in more powerful machines and just deploying the pre-trained models in smartphones is clearly justified.

The improvement in computing power is clear when comparing the older phones with the more recent ones. If this trend continues, the possibility of smartphones running more complex problems should not be discarded.

We found very noticeable variations in resource usage, which might be caused by both hardware and software differences. For example, CPU utilization was more consistent in some devices (see Fig. 1). This is suspected to be caused by the operating system having to interrupt our process in order to perform some background task. A common situation probably caused by hardware limitations, the number of applications running in the background, the runtime, or a combination of these.

The differences in standard deviation for the execution times are probably related to the previously mentioned situation. And these variations in execution times might have repercussions in the usability of the device, which in turn could play a significant role in the acceptance of the users for applications that run these kinds of algorithms.



(a)

(b)

Fig. 1. CPU usage for an execution of NSGAII, for (a) Device 8, and (b) Device 6.

While running the tests, an increase in temperature was felt in most devices, enough to be noticeable when holding the smartphone or having it at rest in the front trouser pocket. This situation, and the impact in battery life, will probably be influential in the choice of the users of using, or not using, applications relying on this kind of processing. A possible solution to these objections could be to schedule the intensive processing to be performed when the smartphone is being charged and is not being actively used by its owner (for example, at night).

7 Conclusions and Future Work

We performed an exploratory evaluation of execution times and resource usage for java implementations of optimization and classification algorithms in Android smartphones. The fastest device took, on average, 6–8 times longer to run the evaluated algorithms than a laptop computer. Considering that the other smartphones were about 7 times slower for optimization tasks, and three times slower for supervised learning, it is clear that the current workflow of deploying only pre-trained models in smartphones is justified if we are working with big datasets or complex problems.

However, the execution times for low scale applications are acceptable, suggesting smartphones might be underused for this kind of tasks. Potential use cases for this type of algorithms include the custom refinement of previously generated models for contextual applications, by performing optimization or training classifiers on data acquired in the specific context of the device's owner. Educational software that can show real life applications of these algorithms are another area of opportunity, with the possibility to put data acquisition and model training, literally, in the hands of students.

As for future work, the lower computational capacity of smartphones might be compensated by performing calculations when the device is charging and not in use. This strategy, and its energetic efficiency, remain to be explored. The viability of distributed computing using smartphones is another possible research topic.

References

1. IDC: Worldwide smartphone market posts 11.6% year-over-year growth in Q2 2015, the second highest shipment total for a single quarter, according to IDC (2015). http://www.idc.com/getdoc.jsp?containerId=prUS25804315
2. Entner, R.: International Comparisons: The Handset Replacement Cycle (2013). http://mobilefuture.org/resources/international-comparisons-the-handset-replacement-cycle-2
3. Lockhart, J.W., Pulickal, T., Weiss, G.M.: Applications of mobile activity recognition. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp 2012, pp. 1054–1058. ACM, New York (2012)
4. Reyes-Ortiz, J.L., Oneto, L., Samà, A., Parra, X., Anguita, D.: Transition-aware human activity recognition using smartphones. Neurocomputing **171**, 754–767 (2016)
5. Chen, S., Xue, M., Tang, Z., Xu, L., Zhu, H.: Stormdroid: a streaminglized machine learning-based system for detecting android malware. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 377–388. ACM (2016)
6. Alammar, J.: Supercharging android apps with tensorflow (2016). https://jalammar.github.io/Supercharging-android-apps-using-tensorflow/
7. IDC: Smartphone Growth Expected to Drop to Single Digits in 2016, Led by China's Transition from Developing to Mature Market, According to IDC (2016). http://www.idc.com/getdoc.jsp?containerId=prUS41061616

8. Loghin, D., Tudor, B.M., Zhang, H., Ooi, B.C., Teo, Y.M.: A performance study of big data on small nodes. Proc. VLDB Endow. **8**(7), 762–773 (2015)
9. Pathak, A., Hu, Y.C., Zhang, M.: Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof. In: Proceedings of the 7th ACM European Conference on Computer Systems, pp. 29–42. ACM (2012)
10. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: USENIX Annual Technical Conference, vol. 14 (2010)
11. Gutierrez, A., Dreslinski, R.G., Wenisch, T.F., Mudge, T., Saidi, A., Emmons, C., Paver, N.: Full-system analysis and characterization of interactive smartphone applications. In: 2011 IEEE International Symposium on Workload Characterization (IISWC), pp. 81–90. IEEE (2011)
12. Papandrea, M.: A smartphone-based energy efficient and intelligent multi-technology system for localization and movement prediction. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 554–555. IEEE (2012)
13. IDC: Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 1st quarter 2016 (2016). http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems
14. Durillo, J.J., Nebro, A.J., Alba, E.: The jMetal framework for multi-objective optimization: design and architecture. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
15. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011). Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm
16. Raff, E.: JSAT: Java statistical analysis tool (2015). https://github.com/EdwardRaff/JSAT
17. Wang, J., Liu, P., She, M.F., Nahavandi, S., Kouzani, A.: Bag-of-words representation for biomedical time series classification. Biomed. Sig. Process. Control **8**(6), 634–644 (2013)

http://www.springer.com/978-3-319-62427-3

Advances in Soft Computing
15th Mexican International Conference on Artificial Intelligence, MICAI 2016, Cancún, Mexico, October 23–28, 2016, Proceedings, Part II
Pichardo Lagunas, O.; Miranda-Jiménez, S. (Eds.)

2017, XXVII, 552 p. 195 illus., Softcover
ISBN: 978-3-319-62427-3