Adaptation of Applications to Compare Development Frameworks in Deep Learning for Decentralized Android Applications

Beatriz Sainz-de-Abajo[1] *, Sergio Laso[2], Jose Garcia-Alonso[3], Javier Berrocal[3]

[1] Universidad de Valladolid, Valladolid (Spain)
[2] Global Process and Product Improvement S.L., Cáceres (Spain)
[3] Universidad de Extremadura, Cáceres (Spain)

Received 24 April 2022 | Accepted 14 March 2023 | Published 19 April 2023

Abstract

Not all frameworks used in machine learning and deep learning integrate with Android, which requires some prerequisites. The primary objective of this paper is to present the results of the analysis and a comparison of deep learning development frameworks, which can be adapted into fully decentralized Android apps from a cloud server. As a work methodology, we develop and/or modify the test applications that these frameworks offer us a priori in such a way that it allows an equitable comparison of the analysed characteristics of interest. These parameters are related to attributes that a user would consider, such as (1) percentage of success; (2) battery consumption; and (3) power consumption of the processor. After analysing numerical results, the proposed framework that best behaves in relation to the analysed characteristics for the development of an Android application is TensorFlow, which obtained the best score against Caffe2 and Snapdragon NPE in the percentage of correct answers, battery consumption, and device CPU power consumption. Data consumption was not considered because we focus on decentralized cloud storage applications in this study.

Keywords

Android Applications, Decentralized, Deep Learning, Framework, Images, TensorFlow.

DOI: 10.9781/ijimai.2023.04.006

I. Introduction

THE availability of large volumes of data allows the evolution of artificial intelligence (AI) [1], [2]. For the first time in the history of humankind, systems can analyse the information generated at exponentially faster speeds.

Machine learning (ML) is the practice of using algorithms to analyse data, learn from it and make a prediction about something [3]. ML and deep learning (DL) algorithms must manage a lot of information to produce results that accurately describe reality [4], [5]. That information can draw conclusions about what we think and feel.

These models have drawn ever-increasing research interest due to their intrinsic capability to overcome the drawbacks of traditional algorithms [6]. ML, DL and IA have grown in their use given their benefits in different contexts [7]. In recent years, many studies have shown that combining ML and DL techniques is especially useful in image analysis [8], [9]. DL have proven effectiveness in object and image recognition, natural language processing, speech recognition, robot navigation systems, self-driving cars and health care. [10], [11]. This allows the precise detection of a disease, locating stolen and sold objects via the Internet, searching for missing persons, etc. Due to its great potential, this technique is applied in a large number of sectors such as security, health, finance, automotive and agriculture. However, DL takes ML to a more detailed level, reducing the margin of error and increasing the accuracy of the conclusions it reaches [12], [13]. In this case, the system goes through layers or neuronal units. While in ML, to perform a classification, it is necessary to indicate the characteristics; in DL, the algorithm will perform the classification during the training by itself.

Each layer processes the information and returns a result in the form of weighting. The second layer that analyses the image will combine the result obtained by the first layer with its own judgement. As a result, the weighting will change. The third layer will use this new modified weighted result to perform its calculations, reducing the margin of error and thus increasing the accuracy of its results. The system trains itself due to a large amount of information being considered, improving its weighting.

Data storage and preparation tasks for further processing require the most time [14] but are essential because AI algorithms develop complex processes of understanding and interpreting data and therefore need them to provide value.

Although existing ML and DL services use cloud computing and servers to run, and therefore require an Internet connection, there is a trend towards decentralization [15], [16]. [17] argues that decentralizing AI opens the door for more equitable development. Instead of connecting to data centre-based services, queried through mobile communications, AI capabilities will reside on the device itself.

* Corresponding author.
E-mail address: beasai@uva.es

ML and DL are the key technologies on which new functionalities, personalization and connectivity with other devices in the Internet of Things (IoT) will be based.

II. Tasks and Methods

This study has been structured into three tasks: (A) revision of the most well-known frameworks; (B) test application development; and (C) analysis and comparison. Finally, we show the results of this study. Fig. 1 shows the flow chart followed in this study.

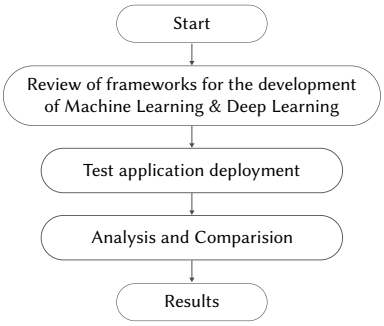


Fig. 1. Flow chart.

A. Review of Frameworks for the Development of Machine Learning & Deep Learning

The eight most commonly used frameworks globally were reviewed [18]. Table I shows links to the official websites that list the best features of each framework and indicates whether integration with Android is allowed.

TABLE I. ML & DL Frameworks

	Name	Official Web	Android integration
1	**TensorFlow** [19]-[21]	https://www.tensorflow.org/	Yes
2	**Caffe** [22]	https://caffe.berkeleyvision.org/	No
3	**Caffe2** [a] [23]	https://pytorch.org/	Yes
4	**Amazon Machine Learning** [24]	https://aws.amazon.com/es/machine-learning/	No
5	**CNTK** [25]	https://docs.microsoft.com/en-us/cognitive-toolkit/	No
6	**Torch** [26]	http://torch.ch/	No
7	**Snapdragon NPE**	https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk	Yes
8	**DeepLearning4J** [27]	https://deeplearning4j.konduit.ai/	Yes

[a] Caffe2 and PyTorch projects are merging now [23].

After a first filter, we analyse the frameworks that are exportable to Android. Table II summarizes the requirements to be executed on a smartphone and its uses.

B. Test Application Deployment

We define the functional requirements (FR) and non-functional requirements (NFR) of applications to be developed or adapted.

All applications have the same requirements so that they behave in a similar way.

- FR-1. Through the trained model, the included images can be recognized.
- FR-2. The applications show the recognition result together with a percentage of success probability.
- FR-3. The recognition will be static from a photograph included in the application.
- NFR-1. The applications must achieve reasonable response times when executing the deep learning model.
- NFR-2. The applications will be functional for smartphones with an Android OS 6.0 or higher operating system.

For each framework, we implement the corresponding application. Whenever possible, we use the test applications from the official repositories because it would take a long time to implement the integration of these frameworks in Android from the beginning. If necessary, we make adjustments and developments, such as modifications to the code so that all applications have the same features.

The primary goal of this study is not to create commercial applications but rather to provide simple functionality to facilitate the objective of this study: to compare frameworks under equal conditions.

The applications include an image gallery. The user clicks on each image, and the application shows the result of the recognition and the probability of success of the clicked image.

Listed below are the changes implemented in each application and the problems found:

1. TensorFlow app.

   The TensorFlow app was retrieved from the official TensorFlow repository [28]. The most important change was to modify the primary functionality of the application. TensorFlow originally used a live camera to recognize objects, but this was changed to measure parameters correctly in the subsequent comparison. The functionality changed to a list of images where the user clicks an image, and the application returns the recognition result.

2. Caffe2 app.

   The AICamera application was obtained from the official repository [29]. The most important change was the same as that in the TensorFlow application. Caffe2 integrates with C++ to perform model recognition and execution.

3. Snapdragon NPE app.

   The SNPE Image Classifier application was obtained from the SDK, available on the official Qualcomm Developer Network repository [30]. No changes were necessary because the application provided the functions that were proposed in the requirements. Its development is only possible on Linux because it uses Snapdragon libraries that are included in the repository and only compatible on Linux.

4. Deeplearning4J app.

   The DL4JImageRecognitionDemo application was obtained from the official repository [31]. Due to its limited development to date, it has not been possible to use this application. Although it was modified, the result was unsuccessful. The application did not compile correctly, possibly due to bugs with the libraries or some type of incompatibility. Because implementing an application that integrates the framework from the beginning requires a long time, this application was discarded for the testing phase.

When defining the requirements of the model before training, the authors agreed to use a pre-trained DL model that was available in the frameworks because each framework has a different format.

The operation is the same for all three applications, which have a list of images that the user clicks on. The apps then display the recognition result along with the probability of success, which the model thinks is the clicked product. At this stage, we make a limited number of attempts.

TABLE II. Frameworks Compatible With the Android Operating System

Name	Characteristics	Requirements	Uses
TensorFlow	• Execution of neural models. • Hardware acceleration thanks to the Android Neural Networks API.	• Android API 23 (Marshmallow) or later and NDK 12b or later.	• Computer vision. • Voice and image recognition. • Medical applications. • Intelligent searches. • Intelligent answers in emails.
Caffe2*	• Execution of neural models. • Hardware acceleration thanks to the Android Neural Networks API. • Offers conversion from Torch models to Caffe2.	• Android API 21 (Lollipop) or higher.	• Computer vision. • Voice and image recognition. • Translation. • Chatbots. • IoT. • Medical applications.
Snapdragon NPE	• Execution of neural models. • Compatibility with TensorFlow, Caffe and Caffe2. • Developed on Linux.	• For GPU: Qualcomm Snapdragon 845, 820, 835, 625, 626, 650, 652, 653, 660, 630, 636, and 450. • For Adreno GPU: libOpenCL.so	• Object classification. • Face detection. • Natural language understanding. • Speech recognition. • Security/authentication. • Resource management.
DeepLearning4J	• To create & train a neural network on an Android device.	• Android API 21 (Lollipop) or higher.	• Object and speech recognition. • Natural language processing. • Data prediction.

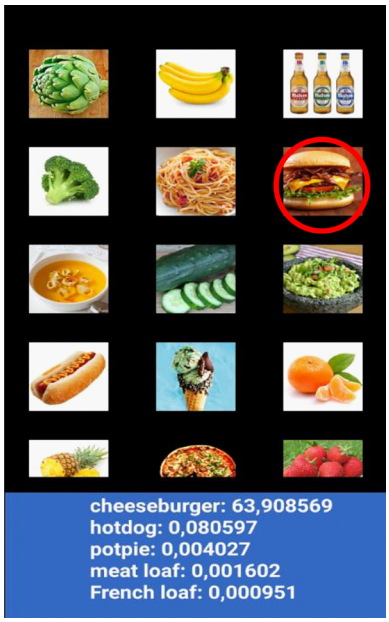


Fig. 2. Test in TensorFlow.

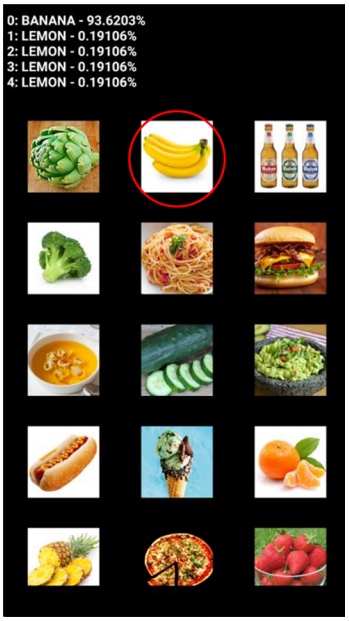


Fig. 3. Test in Caffe2.

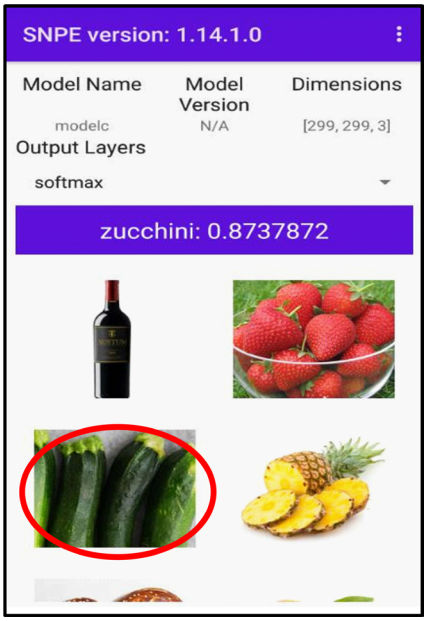


Fig. 4. Test in Snapdragon NPE.

Fig. 2, Fig. 3 and Fig. 4 show the image recognition of each application. We highlight with a red circle the image pressed during the preliminary evaluation tests.

Fig. 2 shows the test performed with TensorFlow when clicking on the photo of a cheeseburger. The percentage of success is 63.91%, and those of the other images were less than 10%.

In the test with Caffe2, the percentage is 93.62% after choosing the photo of bananas, with the other options near 0% (see Fig. 3)

In Fig. 4, after testing on the Snapdragon NPE, we also obtained a high success rate (87.38%) when selecting zucchini.

C. Analysis and Comparison

As basic applications, there are no differences in the aspects related to code optimization. The proposed analysis parameters were (1) success rate; (2) battery consumption; and (3) processor power consumption.

1. *Success rate:* The result is determined by the success or failure of each food. When all frameworks obtain the same successes or failures, by of the highest percentage of probability, i.e., the one with the highest probability of success for each detection. For this purpose, the same gallery with 20 food images was included in each application. After this, a round was performed in which each image was clicked on, and the result of the prediction was saved in a table. We checked the results in all cases along with the percentage of the probability of success (see Table III).
2. *Battery consumption:* Nowadays, we are always on the lookout for our smartphone's battery to reach the end of the day without running out, because with the multimedia content we watch and the hours of use we give it, few smartphones have large battery capacities. For this reason, it is essential to choose the framework that consumes the least so that the user notices it as little as possible. In order to evaluate this parameter, the new versions of Android,

TABLE III. PERCENTAGE OF CORRECT ANSWERS

Food	Image	TensorFlow	Caffe2	Snapdragon NPE (CPU)	Snapdragon NPE (GPU)
Artichoke		Artichoke 83.24%	Artichoke 99.99%	Artichoke 99.74%	Artichoke 99.70%
Banana		Banana 80.10%	Banana 93.91%	Banana 51.02%	Banana 51.66%
Beer Bottle		Beer Bottle 67.65%	Scale 55.14%	Beer Bottle 54.38%	Beer Bottle 54.68%
Broccoli		Broccoli 83.18%	Broccoli 99.74%	Broccoli 99.56%	Broccoli 99.51%
Burrito		Burrito 82.31%	Burrito 99.93%	Pinwheel 19.58%	Pinwheel 15.36%
Carbonara		Carbonara 83.17%	Carbonara 99.07%	Swab 41.59%	Swab 36.49%
Cheeseburger		Cheeseburger 83.17%	Hot Dog 47.40%	Cheeseburger 72.18%	Cheeseburger 70.55%
Consommé		Consommé 78.59%	Consommé 98.19%	Washbasin 60.95%	Washbasin 69.87%
Cucumber		Cucumber 83.20%	Cucumber 98.95%	Cucumber 99.85%	Cucumber 99.80%
Guacamole		Guacamole 82.98%	Guacamole 98.86%	Mortar 37.96%	Mortar 40.62%
Hotdog		Hot Dog 82.86%	Hot Dog 99.87%	Jellyfish 3.62%	Jellyfish 3.32%
Ice Cream		Ice Cream 24.05%	Honeycomb 33.80%	Pedestal 12.36%	Pedestal 15.29%
Meat Loaf		Meat Loaf 83.13%	Meat Loaf 67.83%	Ice Lolly 22.66%	Ice Lolly 28.56%
Orange		Orange 77.62%	Orange 77.56%	Orange 43.36%	Orange 41.82%
Pineapple		Pineapple 77.76%	Spaghetti squash 51.51%	Necklace 23.38%	Necklace 20.78%
Pizza		Pizza 80.11%	Pizza 99.19%	Wall clock 13.53%	Wall clock 13.64%
Pretzel		Pretzel 83.27%	Pretzel 99.40%	Pretzel 90.92%	Pretzel 89.79%
Strawberry		Strawberry 83.22%	Strawberry 94.68%	Golf Ball 82.70%	Golf Ball 82.47%
Wine bottle		Red Wine 69.69%	Whistle 30.39%	Wine Bottle 82.07%	Wine Bottle 80.71%
Zucchini		Zucchini 79.95%	Cucumber 78.24%	Zucchini 88.22%	Zucchini 87.30%
Correct		**20**	**14**	**11**	**11**
Wrong		**0**	**6**	**9**	**9**
Percentage of correct answers		**100%**	**70%**	**55%**	**55%**

Google offers a web service called Battery Historian [32], in which we enter a log obtained from Batterystats. Batterystats.bin is a file that works as a registry, where Android saves the consumption data of the mobile device either via hardware or software services. The operating system uses this file to monitor consumption and battery level, and to display consumption statistics. The operating system is programmed to reset the file when the battery is fully recharged. When the battery is discharged, we record new data about battery use and charging. With the tool, we will evaluate (1) device estimated power use; (2) device estimated power use due to CPU usage; and (3) CPU user time.

3. *Processor power consumption:* One of the factors affecting the battery is CPU consumption. For this reason, we will measure what percentage is consumed each time an image recognition is performed. Another factor is that the lower the CPU power consumed, the faster and smoother the app will experience on lower-end devices with a more moderate processor, so the app will cover more of the market. To measure the power consumed, the Android Profiler tool from Android Studio was used [33]. This tool provides real-time data related to the CPU, memory and network activity of an application. You can perform sample-based method tracing for time code execution, capture dumps, view memory allocation, and inspect information from files transmitted over the network. In this study, we focus on the CPU Profiler, which shows the power that the CPU is consuming on any interaction that we make in the system or after selecting a specific application in real time.

One of the premises of this study is that the application is integrated into the device and, therefore, decentralized with respect to any server. Thus, it is not necessary to measure data traffic.

D. Resources Used

The hardware resources used in this study included the following:

- PC Asus X54HR
- Smartphone Xiaomi Mi3
  - To test the test applications.
  - To measure the parameters to compare the test applications.
- Smartphone Xiaomi Redmi Note 4
  - To test the analysis with the GPU offered by the Snapdragon Neural Processing Engine (Snapdragon NPE), because its processor and GPU are compatible with the framework for that function.

The tools for the implementation and development of the applications are as follows:

- Android Studio.
- Inception V3 model.
  - Trained by ImageNet content (https://image-net.org/) with data from 2012.
  - This model is composed of more than 1000 different classes: objects, animals, food, etc.

III. Results

The framework that stands out in a single parameter is not the best but the one that is more balanced considering all parameters. Next, we show the results for each framework.

A. Success Rate

To perform tests whose results are comparable, every time the test on each application was performed, all system applications were closed; having applications open in the background may influence the measurements. Regarding the Snapdragon NPE application, a smartphone compatible with GPU analysis was used to verify the differences between frameworks. Table III shows the results obtained.

Tensorflow has a 100% success rate. It also shows a stable behaviour, i.e., it obtains a high probability percentage in the cases it gets right, with a small exception. In the detection of the Ice Cream image, the prediction is correct but the probability percentage is low (24.05%).

Caffe2 is in second place with a success rate of 70%. Despite not achieving a 100% hit rate, the behaviour is also stable. In the cases it succeeds it obtains high probability percentages, and in the cases it fails it gets low probability percentages no higher than 55%. Therefore, we can easily detect whether a detection is wrong based on the probability percentage. Regarding the wrong predictions, in most cases the result is not similar or interpretable with the original. For example, with Pineapple the framework has detected Spaghetti squash, or with Wine bottle the detection obtained has been Whistle.

Snapdragon NPE is in third and last place with a success rate of 55%. Its results have been fair/poor, even with the GPU analysis that supposedly increases performance. The behaviour has not been as stable as in the previous frameworks. In certain successful results it gets a low percentage of probability, for example Banana (51.02%) or Orange (43.36%). The opposite also occurs. With Strawberry, it detects Golf Ball with 82.70%. Regarding the wrong predictions, some of the wrong results obtained, if they can resemble with the original, e.g. Guacamole and Mortar or Strawberry and Golf Ball.

Although the three applications use the same image recognition model (Inception V3), the model has to be adapted to each framework, so performance may change [34]. It can also affect the software optimization of each framework in the operating system (OS). In this case, Tensorflow is developed by Google, the same developer as the Android OS, so it could be better optimized and therefore get better results [35].

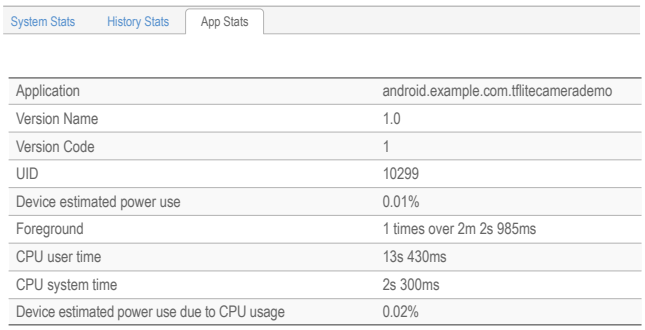
B. Battery Consumption

A 2-minute execution test was performed in which each image was analysed 2 times.

Before starting the test and running the applications, we reset the device's consumption log file and its history using the "adb shell dumpsys batterystats" command. Fig. 5, Fig. 6 and Fig. 7 show the captures made by the battery historian tool.

The frameworks that consume less power are Caffe2 and TensorFlow. Snapdragon NPE performs the worst, with a difference of 10% (estimated battery consumption) compared to TensorFlow.

The "CPU user time" in TensorFlow and Caffe2 is 13.430 s and 34.320 s, respectively, while in Snapdragon NPE, it is 112.530 s, indicating much higher consumption with Snapdragon NPE. While TensorFlow and Caffe2 seem to use the CPU only when they parse the image or update the display, Snapdragon NPE constantly consumes resources.


System Stats | History Stats | App Stats

Application	android.example.com.tflitecamerademo
Version Name	1.0
Version Code	1
UID	10299
Device estimated power use	0.01%
Foreground	1 times over 2m 2s 985ms
CPU user time	13s 430ms
CPU system time	2s 300ms
Device estimated power use due to CPU usage	0.02%

Fig. 5. Tersorflow battery test.

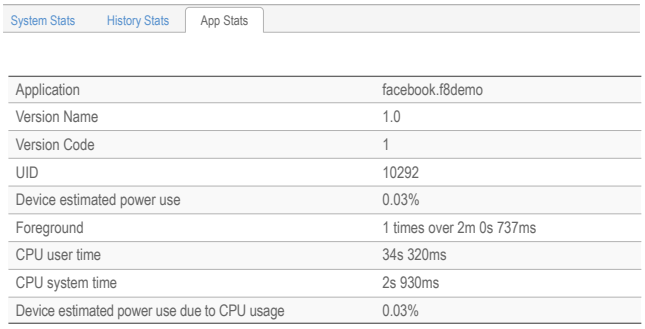


Fig. 6. Caffe2 battery test.

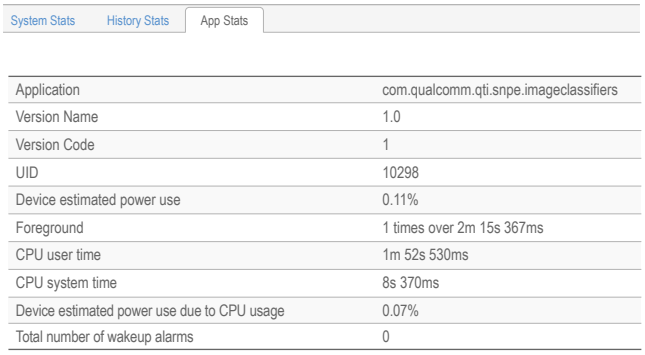


Fig. 7. Snapdragon NPE battery test.

TABLE IV. Battery Consumption

	Device estimated power use	Device estimated power use due to CPU usage	CPU user time
TensorFlow	0.01%	0.02%	13s 430ms
Caffe2	0.03%	0.03%	34s 320ms
Snapdragon NPE	0.11%	0.07%	1min 52s 530ms

C. CPU Power Consumption

In this test, we connect the mobile device to a PC to use an Android CPU Profiler. With this tool, we can measure the power consumed by the application when it analyses an image.

We report the average calculation in a given timeframe in which the application analyses ten images. After the calculations, we obtain an average consumption for the analysis of a single image (see Table V).

TABLE V. CPU Power Consumption

	Average consumption per image analysis
TensorFlow	38%
Caffe2	35%
Snapdragon NPE	43%

D. Summary

Although there are no noticeable differences, Snapdragon NPE consumes more CPU time to analyse an image than Caffe2 and TensorFlow.

After the tests, a summary of the results of the comparison is shown in Table VI.

TABLE VI. Comparison Summary

	Percentage of correct answers	Battery consumption	CPU power consumption
TensorFlow	100%	0.01%	38%
Caffe2	70%	0.03%	35%
Snapdragon NPE	55%	0.11%	43%

IV. Conclusion and Future Lines of Work

There are more and more developers in the application market and, therefore, more competition. For this reason it is necessary to choose the framework with the best results, so that the user does not feel disappointed.

TensorFlow and Caffe2 produce have much better results than Snapdragon NPE, which also exhibited highest battery consumption and a fair to poor response success rates.

The battery and CPU consumptions are similar for TensorFlow and Caffe2, but the response rate is better with TensorFlow. Additionally, TensorFlow is a Google framework, has a large community on both Github and Stack Overflow, and is well documented with questions, reviews and tutorials online. TensorFlow also keeps a close eye on these user communities to improve their platform. Thus, based on this study's results, TensorFlow is the most recommended for the implementation of an Android application.

There are different areas where image recognition is applied that could benefit from this type of development. The most interesting are the areas of health and wellness. First, through the diagnosis of diseases, after analysing the alterations in the X-rays. In the second, improving the management of food purchases. With a simple application on the mobile, the user could check the lack of products in the pantry.

Having selected this framework, we plan to develop an assistance application for food that will allow a user to take one or more photos with a smartphone camera and recognize food using the trained model. Next, we plan to transfer the list of products from a kitchen pantry to a smartphone. Using the same recognition function, we plan to generate a shopping list by checking which products are missing in the pantry and which we should buy.

Acknowledgment

This study was supported in part by the DIN2020-011586 Grant, funded by MCIN/AEI/10.13039/501100011033 and the European Union "NextGenerationEU/PRTR," and by the Interreg V-A España-Portugal 2014-2020, under Project 0786_CAP4ie_4_.

It was also supported by the "movilidad investigadores e investigadoras UVa-BANCO SANTANDER 2022" grant.

References

[1] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of big data – evolution, challenges and research agenda," *International Journal of Information Management*, vol. 48, pp. 63–71, 2019, doi: 10.1016/j.ijinfomgt.2019.01.021.

[2] L. Spector, "Evolution of artificial intelligence," *Artificial Intelligence*, vol. 170, no. 18. pp. 1251–1253, Dec. 2006, doi: 10.1016/j.artint.2006.10.009.

[3] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3. pp. 161–175, 2018, doi: 10.1016/j.dcan.2017.10.002.

[4] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7. pp. 53040–53065, 2019, doi: 10.1109/

ACCESS.2019.2912200.
[5] W. Samek and K. R. Müller, "Towards explainable artificial intelligence," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11700 LNCS, 2019, pp. 5–22.
[6] N. Bouchra, A. Aouatif, N. Mohammed, and H. Nabil, "Deep belief network and auto-encoder for face classification," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 5, p. 22, 2019, doi: 10.9781/ijimai.2018.06.004.
[7] F. J. García-Peñalvo *et al.*, "Application of artificial intelligence algorithms within the medical context for non-specialized users: The cartier-ia platform," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 6, 2021, doi: 10.9781/ijimai.2021.05.005.
[8] S. H. Chen, C. W. Wang, I. H. Tai, K. P. Weng, Y. H. Chen, and K. S. Hsieh, "Modified yolov4-densenet algorithm for detection of ventricular septal defects in ultrasound images," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 7, 2021, doi: 10.9781/ijimai.2021.06.001.
[9] M. I. Khattak, M. Al-Hasan, A. Jan, N. Saleem, E. Verdú, and N. Khurshid, "Automated detection of covid-19 using chest x-ray images and ct scans through multilayer-spatial convolutional neural networks," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 6, 2021, doi: 10.9781/ijimai.2021.04.002.
[10] A. Venkat, T. Rusira, R. Barik, M. Hall, and L. Truong, "SWIRL: high-performance many-core CPU code generation for deep neural networks," *International Journal of High Performance Computing Applications*, vol. 33, no. 6, 2019, doi: 10.1177/1094342019866247.
[11] S. S. Nisha, M. M. Sathik, and M. N. Meeral, "Application, algorithm, tools directly related to deep learning," in *Handbook of Deep Learning in Biomedical Engineering: Techniques and Applications*, 2020.
[12] Y. Xin *et al.*, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.
[13] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, 2020, doi: 10.1007/s11831-019-09344-w.
[14] N. El Aboudi and L. Benhlima, "Big data management for healthcare systems: architecture, requirements, and implementation," *Advances in Bioinformatics*, vol. 2018. 2018, doi: 10.1155/2018/4059018.
[15] F. L. Koch, "Decentralized network management using distributed artificial intelligence," *Journal of Network and Systems Management*, vol. 9, no. 4, pp. 375–388, 2001, doi: 10.1023/A:1012976206591.
[16] I. Gupta, "Decentralization of artificial intelligence: analyzing developments in decentralized learning and distributed AI networks," *Researchgate.Net*, no. May, 2020, doi: 10.13140/RG.2.2.17018.93124.
[17] G. A. Montes and B. Goertzel, "Distributed, decentralized, and democratized artificial intelligence," *Technological Forecasting and Social Change*, vol. 141. pp. 354–358, 2019, doi: 10.1016/j.techfore.2018.11.010.
[18] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, "Various frameworks and libraries of machine learning and deep learning: a survey," *Archives of Computational Methods in Engineering*, 2019, doi: 10.1007/s11831-018-09312-w.
[19] Google, "TensorFlow Lite guide," *TensorFlow*, 2020.
[20] Tutorials Point, "TensorFlow tutorial," *Tutorials Point Pvt. Ltd.*, p. 90, 2019.
[21] M. Abadi *et al.*, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016,* Savannah, GA, USA, 2016, pp. 265–283, doi: 10.5555/3026877.3026899.
[22] Y. Jia *et al.*, "Caffe: convolutional architecture for fast feature embedding," in *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, Nov. 2014, pp. 675–678, doi: 10.1145/2647868.2654889.
[23] Facebook, "Caffe2 and PyTorch join forces to create a research + production platform PyTorch 1.0," *Caffe2 Documentation*, 2018.
[24] A. Mishra, "Amazon machine learning," in *Machine Learning in the AWS Cloud*, 2019, pp. 317–351.
[25] F. Seide and A. Agarwal, "CNTK : Microsoft's open-source deep-learning toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 2135-2135, doi: 10.1145/2939672.2945397.
[26] R. Collobert, C. Farabet, and K. Kavukcuoğlu, "Torch | Scientific computing for LuaJIT.," *NIPS Workshop on Machine Learning Open Source Software*, 2008.
[27] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, and E. Frank, "WekaDeeplearning4j: a deep learning package for Weka based on deeplearning4j," *Knowledge-Based Systems*, vol. 178, pp. 48–50, Aug. 2019, doi: 10.1016/j.knosys.2019.04.013.
[28] M. Abadi *et al.*, "TensorFlow, Large-scale machine learning on heterogeneous systems." 2015, doi: 10.5281/zenodo.4724125.
[29] Facebook, "AICamera application," 2017. https://github.com/facebookarchive/AICamera (accessed Jul. 28, 2022).
[30] ©2022 Qualcomm Technologies Inc. and/or its affiliated companies, "Qualcomm neural processing SDK for AI," 2022. https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk (accessed Jul. 28, 2022).
[31] Eclipse Foundation, "deeplearning4j," *github.com*, 2019. https://github.com/eclipse/deeplearning4j (accessed Jul. 28, 2022).
[32] Google Developers, "Analyze power use with battery historian," 2022. https://developer.android.com/topic/performance/power/battery-historian (accessed Jul. 28, 2022).
[33] Google Developers, "The Android profiler," 2022. https://developer.android.com/studio/profile/android-profiler (accessed Jul. 28, 2022).
[34] A. Chowanda and R. Sutoyo, "Convolutional neural network for face recognition in mobile phones," *ICIC Express Letters*, vol. 13, no. 7, pp. 569–574, 2019, doi: 10.24507/icicel.13.07.569.
[35] H. C. Takawale and A. Thakur, "Talos App: on-device machine learning using TensorFlow to detect Android malware," in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, Oct. 2018, pp. 250–255, doi: 10.1109/IoTSMS.2018.8554572.

Beatriz Sainz-de-Abajo

She is currently an Associate Professor in Telecommunications Engineering at the University of Valladolid in Spain. She received the Ph.D. degree (summa cum laude) in the University of Cordoba in 2009 and has a master's degree in Data Networks and Transportation Networks from Lucent Technologies. Her fields of action are the development and evaluation of e-Health systems, m-Health, medicine 2.0., cloud computing, etc., focuses on topics related to electronic services for the information society. She belongs to the GTe Research Group, integrated within the UVa Recognized Research Group "Information Society". Actually, she also collaborates with the research "Quercus Software Engineering Group" of the University of Extremadura, Spain. Among the lines of research, the group works to develop innovative solutions in the field of health that help patients improve their quality of life and facilitate the work of health professionals.

Sergio Laso-Mangas

He received the Industrial Ph.D. degree in computer science from the University of Extremadura, Spain, in 2023. He is currently a researcher at the company Global Process and Product Improvement, Cáceres, Spain. His research interests include mobile computing, pervasive systems, the Cloud-to-thing continuum, Quality of Service and the Internet of Things

José Manuel García-Alonso

He is an Associate Professor at the University of Extremadura, Spain and co-founder of Gloin, a software-consulting company and Health and Aging Tech an eHealth company. He got his PhD on software engineering at the University of Extremadura in 2014. He is currently working in the department of Computer and Telematics Systems Engineering, in the area of Languages and Computer Systems. He is part of the research "Quercus Software Engineering Group" and his research interests include quantum software engineering, mobile computing, pervasive computing, eHealth, gerontechnology.

Javier Berrocal

He received the Ph.D. degree in computer science from the University of Extremadura, Spain, in 2014. In 2016, he obtained an Associate position at the University of Extremadura. He is currently working in the department of Computer and Telematics Systems Engineering in the University of Extremadura. He is part of the research "Quercus Software Engineering Group", and his main research interests are mobile computing, context awareness, pervasive systems, crowd sensing, the Internet of Things, and fog computing. He is a cofounder of the company Gloin, which is a software-consulting company, and Health and Aging Tech an eHealth company.