# Distributed Computing System On a Smartphones-based Network

Hamza Salem

[1] Innopolis University– Master's student - Software Engineering
h.salem@innopolis.university

**Abstract.** The number of smartphone users in the world is expected to pass the five billion in 2019, these smartphones have huge processing power wasted without use year after year. The purpose of this paper is to provide a new architecture for a distributed computing system consists of a network of smartphones (nodes); by using their computation power to do some computation process such as Training Machine Learning models. The system contains three main layers; Database Layer, Distribution layer and the Execution layer. The database layer contains real-time database NoSQL to store data objects; the distribution layer contains business logic or (cloud functions) for the distribution algorithm and calculation after the nodes submit their results. Finally, the execution layer has a mobile application that installed on all nodes; inside this application, we have used a function to do computation on all values coming from the distribution layer and resend the results. As proof of concept, we have implemented a minimal viable product (MVP), by applying simple linear regression using a web-based library called JS-Regression on dataset consist JSON file. The proposed solution provides a new source for computation power is not used yet, by combining the power of hundreds, thousands or millions of smartphones this system can be the new layer of the new internet and provide new layer of execution for distributed systems.

**Keywords:** Distributed system, Computation power, JS-regression, JSON (JavaScript Object Notation), MVP (minimal viable product).

## 1 Introduction

latest industry forecasts indicate that the annual worldwide IP traffic consumption will reach 3.3 zettabytes (1015 MB) by 2021, smartphone traffic exceeding PC traffic by the same year [1][2]. Smartphones companies compete with each other by improving the features from processing power to memory and capacity; these features are improving exponentially every year. In the near future, smartphones will have better performance to compete with PCs and other devices such as PlayStation and Xbox. Also, the expansion of network scale and the diversification of services in the 5G era, the amount of data, related to applications, users and networks, will experience an explosive growth, which can help these devices to be more powerful [3] [4].

Systems with machine learning component need a lot of computations and to increase the accuracy of any machine learning model you should provide more data to train the

models. Nowadays Datasets are constantly increasing in size and complexity, the main challenge is the computational power that required to deal with data also should increase. One of the main solutions is the Distributed Computation; it is a universal approach to deal with large datasets, datasets are partitioned across several machines (or workers), the machines perform computations locally and communicate only small bits of information with each other, they coordinate to compute the desired quantity. This is the standard approach taken at large technology companies, which routinely deal with huge datasets spread over computer clusters [5].

Currently every software in the world is a kind of distributed system; for example, when we have website connected to APIs, maybe the code will be hosted on static server and the database on different server and the APIs also in a different server, we can say that Is a distributed system; but in this paper we will take the context of the distributed system in the computing process only [21]. A computer cluster is a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system [6]. The computing industry is one of the fastest growing industries and it is fueled by the rapid technological developments in the areas of computer hardware and software. The technological advances in hardware include chip development and fabrication technologies, fast and cheap microprocessors, as well as high bandwidth and low latency interconnection networks [7].

Creating distributed system can't depend only on the hardware; software is the other key for the process; for example, in personal computers and servers, we have Apache Spark as a popular open-source platform for large-scale data processing that is well-suited for iterative machine learning tasks. Spark is a fault-tolerant and general-purpose cluster computing system providing APIs in Java, Scala, Python, and R, along with an optimized engine that supports general execution graphs. Moreover, Spark is efficient at iterative computations and is thus well-suited for the development of large-scale machine learning applications [8].

On the other hand, software in smartphones or (Applications) have been improved to deal with complex computation; from the beginning of smartphones era, we have seen a lot of operating systems like Android, iOS, Blackberry, Symbian, Windows phone, web OS, Ubuntu, and Firefox. Smartphones provide a wide range of services such as sending or receiving SMS, music, billing, online shopping, online booking, playing games, web browsing, using different apps like WhatsApp, Facebook or Telegram, etc. [9].

The next wave in applications development; is application can provide complex computation that related to machine learning inside the device; for example, the filters by Snapchat are an amazing combination of augmented reality and machine learning algorithms for computer vision. Snapchat seriously considered the importance of Machine Learning only after their acquisition of Looksery, the Ukrainian computer vision company. Snapchat first detects a face, by looking at a photo as a set of data for the color value of each individual pixel. The program looks for areas of contrasts between light and dark parts of the image to determine which part of the picture is a face. Though the algorithm here doesn’t work when tilting your face acutely or face sideways but works really accurately for frontal faces. Also, Pinterest acquired a Machine Learning com-

pany Kosei, and that led the company to leverage the potential of this promising technology. Kosei specialized in the commercial application of Machine Learning particular content discovery and recommendation algorithm, which made Kosei & Pinterest an ideal collaboration. Since this collaboration, Machine Learning has managed to cast a positive impact on almost every aspect of the business operations of Pinterest, whether it is spam moderation, content discovery, advertising monetization, or reducing churn of email newsletter subscribers [10].

On-Device Machine Learning models can be a challenge for some specific OS like IOS; because Apple has some restriction on permission's too access some components in the device. In Android OS the process is more easy for example, we have Talos App; it's an application to uses On-Device Machine Learning using Tensor Flow to detect Android Malware. Talos App aims to solve the problem of malware detection using "Requested Permissions" as the input parameters. The entire detection process takes place on the mobile device, and it doesn't require an Internet connection for its working. The machine learning model is created using Tensor Flow. The model's graph is frozen in the protocol buffer format and then exported for deployment on the mobile device. In the experiments, Talos has demonstrated an accuracy of 93.2%. It could analyze hundreds of apps within a second, even on low-end Android devices [11].

As we saw in the previous paragraph; implementing a distributed system to compute data are exclusive on PC and Servers; such as Apache Spark using ML kit. On the other hand, we have a lot of applications implement machine learning on devices such as Snapchat, Pinterest and Talos App as already trained models and ready to use. This paper proposes and demonstrates new architecture for a distrusted system consists network of smartphones working together to achieve one mission; training a specific type of machine learning algorithm. The paper shows an architecture for mobile application in Android Operating System and focuses on implementing a minimal viable product for the process and how it works. After summing up the process through the architecture; the conclusion discusses which machine algorithm can fit with the proposed solution and what is the main conditions should be provided in the data to make sure the accuracy of the results. The paper is organized into two main part as follows: part one present the related works and demonstrate the architecture for the previous system and the layers inside android architecture diagram. Part two present the proposed solution with architecture and the main components of the system defined every layer and shows how it works, and provide the advantage of the system over the previous systems.

In the next section, we will start with part one and we will demonstrate related works and what is the current solutions in the market and how it works?

## 2 Related work

In this section we will analyze the available solutions and show use cases paved the way to our proposed architecture; the cases can be categories to three main section; first section is to prove the power of smartphone dealing with machine learning; second section we will discuss the type of algorithms can work with distribution systems; the

third section will demonstrate the same system with the internet of things (IoT) environment and see how the system will work?

### 2.1 Section One: On-Device Machine Learning Using TensorFlow to Detect Android Malware

The paper proposing a lightweight method of malware analysis, the Talos application, that uses on-device machine learning and TensorFlow. It aims to solve the problem of malware detection using 'Requested Permissions' as the input parameters. The entire detection process takes place on the mobile device, and it doesn't require an Internet connection to work. The machine learning model is created using TensorFlow. The model's graph is frozen in the protocol buffer format and then exported for deployment on the mobile device. Talos has demonstrated an accuracy of 93.2%. It could analyze apps within a second, even on low-end Android devices [11].

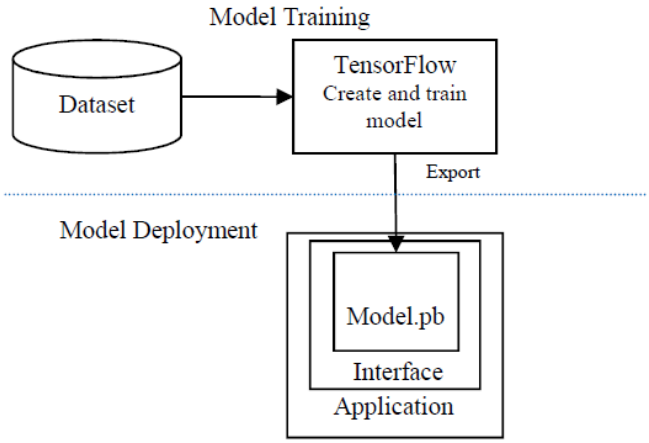



**Figure 1.1 (Model Architecture in Talos)**

As we see in figure 1.1 this architecture performs the analysis as the entire model is present on the Android phone inside the application. Talos use TensorFlow for creating the machine learning model. However, the training part is done on a computer as training a model is very computed intensive. Mobile devices are very low powered for accomplishing this task. Therefore, a model is trained and then the weights and biases are frozen. This frozen model is exported on to the device. It is then used on the mobile device to deliver predictions [11].

### 2.2 Section two: Distributed linear regression by averaging

The paper study one-step parameter averaging in statistical linear models under data parallelism. By doing linear regression on each machine, and take a weighted average of the parameters by studying the performance loss in estimation error, test error, and

confidence interval length in high dimensions, where the number of parameters is comparable to the training data size. The paper provides several key phenomena. First, averaging is not optimal, but the results are simple to use in practice and compare. Second, different problems area affected differently by the distributed framework. Estimation error and confidence interval length increases a lot, while the prediction error increases much less. Third, by proposing a general framework for evaluating the relative efficiency of predicting linear functional of the regression coefficients [5].

### 2.3 Section three: Federated Learning via Over-the-Air Computation

Typical machine learning procedure including the training process and the inference process is supported by cloud computing, i.e., a centralized cloud data center with the broad accessibility of computation, storage, and the whole dataset. However, the emerging intelligent mobile devices and high-stake applications such as drones, smart vehicles and augmented reality, call for the critical requirements of low-latency and privacy. This makes cloud computing based ML methodologies inapplicable [11]. Therefore, it becomes increasingly attractive to possess data locally at the edge devices and then performing training/inference directly at the edge, instead of sending data to the cloud or networks.

The paper focuses on designing the fast model ag-aggregation approach for the Federation Average algorithm to improve communication efficiency and speed up the federated learning system. We observe that the global model aggregation procedure consists of the transmission of locally computed updates from each device, followed by the computation of their weighted average at a central node. You can see in figure 1.2 how the federated learning system works [12].

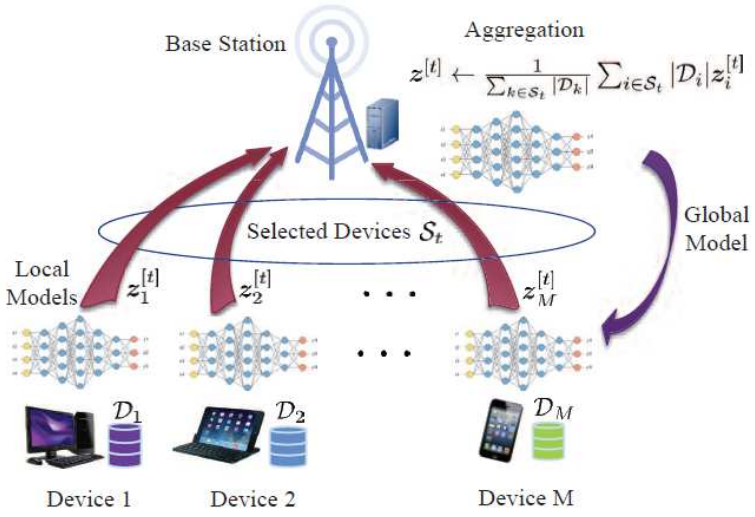



**Figure 1.2: On-device Distributed Federated Learning system.**

By proposing over-the-air computation approach to enable fast global model aggregation for on-device distributed federated learning via harnessing the signal superposition property of a wireless multiple-access channel. However, this architecture focus on working with a wireless local network and the view from the networking side.

In the next section, we will demonstrate our purpose architecture and the main component of our system; describing all the difference between the previous models and the proposed model from the architecture view, performance, and applicability of the system.

# 3 System Design and Development

## 3.1 System Design

The idea of the paper came when I have been assigned to building a project for company working with data scientist; the architecture for the system shown in figure 1.3.

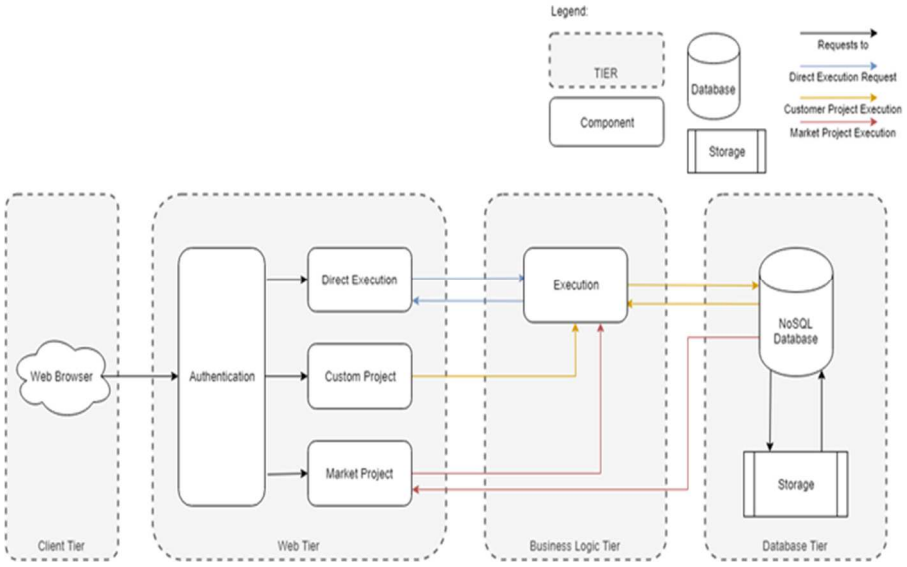



**Figure 1.3 (Four Tier Architecture View for Alogrithma)**

The system is a web application working as a platform between companies and data scientist; by providing an environment for both to execute machine learning algorithm written by data scientist dealing with data that submitted by companies and provide predictions. As you see in this architecture we have the bottleneck in the business logic tier; this tier contains the main component that executes and training models; when we created the risk plan for the project, we have highlighted this issue as the main risk can happen in the future; and the mitigation strategy to solve this risk is upgrading your server with a new processing unit to deal with multiple computations and scale up with the system. However, still, this is not the optimal solution; because more processing

unit means more money to scale up the system. the optimal solution is to scale up without increasing the cost; this paper proposed the solution for the company by creating a new layer for execution.

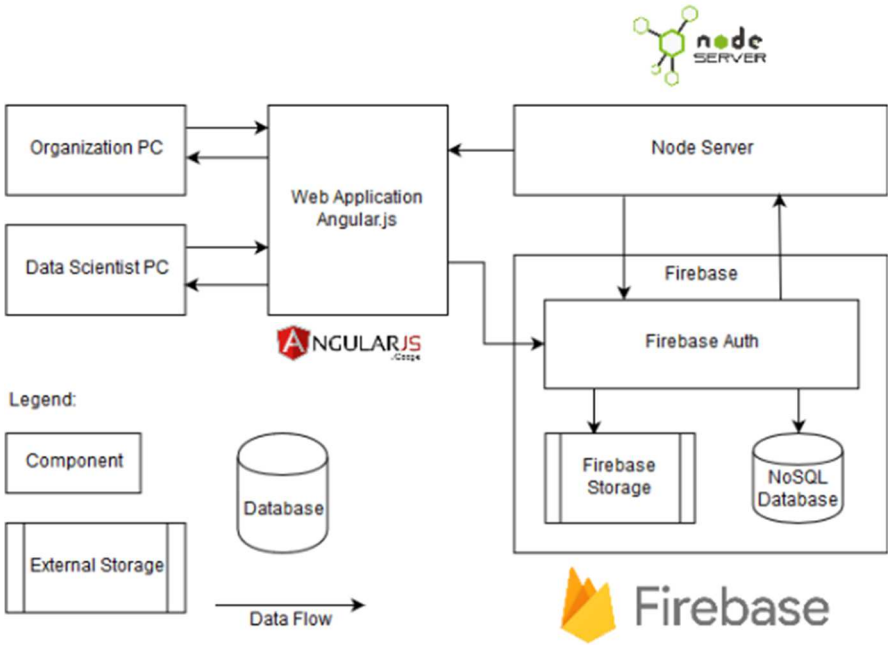



**Figure 1.4(Data Flow Architecture View for Alogrithma)**

Figure 1.4 shows the full architecture data flow view in the system; you can see that the Node server represent the execution layer or (business logic tier); Node server can be scaled by adding more instance inside the server, every instance needs more hardware and processing unit. In figure 1.5 we have replaced the node server with two components; the first one is the Network of smartphones; these Nodes have to have the ability to join the network to lend their processing power; second, function layer inside firebase called contains cloud functions; these function build to provide a mechanism to organize and manage nodes.

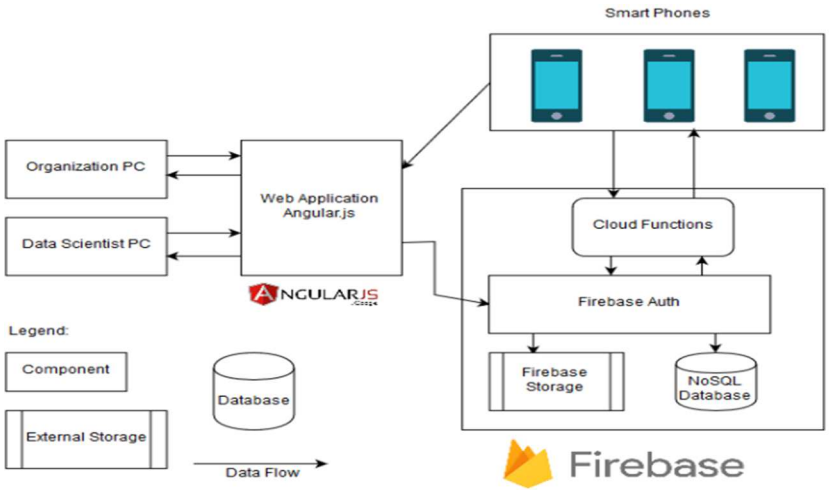



**Figure 1.5 (Proposed Data Flow Architecture for Alogrithma)**

By taking a closer look to both layers see figure 1.6; this figure represents the layered view architecture; as you see in distribution layer we have distributed algorithms to handle the distribution process between nodes, and we will discuss the process in details in the next section. On the other hand, the application layer provides the execution power from each node; and we will discuss the process and how it works and what is the limitations for it.

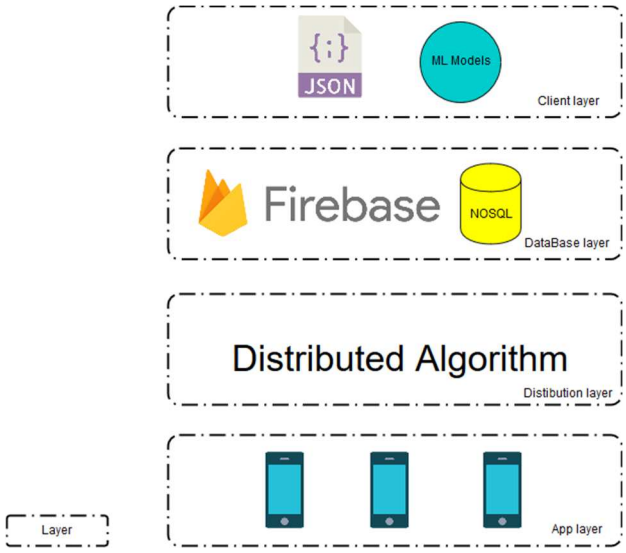



**Figure 1.6 (Layered View Architecture)**

### 3.2 Development

As we discuss in the previous section; we have replaced the execution layer the consists (Node-JS Server) with two main components or tiers (Distribution tier and Application tier). The first part of the implementation is the Distribution tier; this tier contains two main functions works together to serve all nodes; the first function is the "**Evaluator**"; it is responsible to evaluate every node wants to join the network; for example, when we have Smartphone want to join the network, the function evaluates the device based on some characteristics such as Manufacturer, Model, Cores, Threads, Process (nm), Graphics Card, and CPU. And we have characteristics depend on the user such as Time to Join (Period in Second) and percentage of usage during the process. From these characteristics the function return a value called **Evaluation number(EV)** represent how much this node have processing power (min = 0, max= 10); this number always changes when you will join the network based on the current task. For example, we can have two nodes with smartphone type (Samsung NOTE8) same features and same characteristics [13]; but both of the users change the period of joining the network and the percentage of the usage of the device, both of node will have different EV also, in other cases, some node will have fewer features than other nodes; but their EV more than other with best features; because of the time and the percentage of usage are different.

The second function called the "**Distributor**"; it is responsible to distribute the data on all node based on the EV for each node will patriciate with the process and provide the name of the algorithm that will train the data. For example, in our Minimal Viable Product (MVP) was "Linear Regression" based on a library that working on a mobile device and web application called (JS-Regression). This function also has access to the database to calculate the final results from all node that participating in the process.

We have used two main services from Google Firebase, the first service is NoSQL database [14] as the main database for all data inside the software, and the second service is the "cloud functions " that contains the distribution tier functions (**Evaluator, Distributor**); we have chosen a NOSQL firebase database; because we need real-time database to achieve the synchronization between nodes and take the results when it is ready from each node. For example, the system has to provide the read and write form the database for huge number of node at the same time; to fulfill this requirement we have to choose a NoSQL database and the firebase was the first choice because it contains both of the logic to execute (Cloud Functions) and database (real-time Database).

The second part of the implementation is the application tier; this contains the application that will be installed on the Smartphones or Nodes. In our case now; the application is for android devices with extension APK (Application Package). It contains all the libraries, dex files, manifest file, assets and resources that an application requires [11]. The Android architecture provides more control on the permission of using the devices features and hardware part. In figure 1.7 you can see the architecture of android software stack [11].

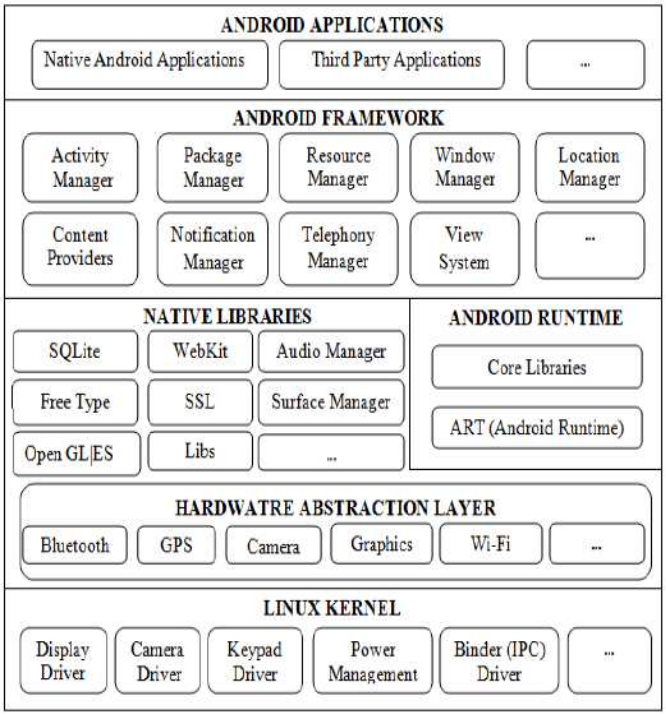



**Figure 1.7 Architecture Of Android Software Stack**

At the bottom of the Software stack of Android, there is a Linux kernel. It acts as the heart of the whole system. It provides various functionalities like memory management, process management, device management, security settings, etc. Also, On the top of the kernel layer there is a set of libraries including surface manager that composes windows on the screen, these libraries called the Native libraries such as: Web Kit Browser Engine, SQLite relational database for storage, Open SSL internet security library etc. in our MVP we have used Cross-platform engine called **IONIC-FRAMEWORK** [15] to access the all native libraries by using plugins Called (**IONIC-NATIVES**) to train data.

Our application can be categories in third-party applications; because native applications provide the basic Android implementation such as Dialer client app, messaging, Web browser and Contact manager [11]. On the other hand, every app made by developers installed on the device is a third-party application.

The basic functionality for the application in MVP stage was to execute and train the data and provide results and insert it is to the Database; in this stage, we didn't use any new special kind of permission to access more features in the device.

As you see in ionic architecture in figure 1.8, the framework provides web-view inside the application and they have created JavaScript bridge to handle the written code and works with native APIs for the device [16].

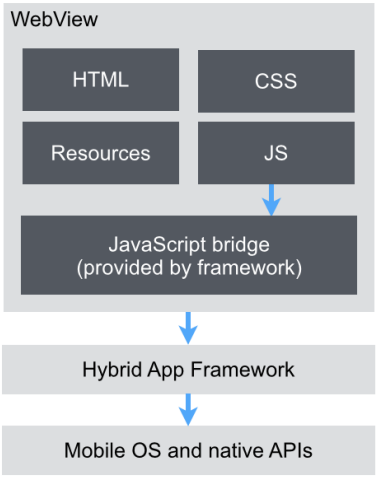



**Figure 1.8 IONIC Framework Architecture**

The application use (JS-Regression) library to train JSON data that comes from the distributor function, the result will be written directly to the database when it is ready. We have used cross-platform to make the MVP more general and try to access more devices; because using **IONIC framework** we can export the application to work with android and IOS.

### 3.3 Minimal Viable Product

We have created this MVP to prove the idea can be implemented and compare the results between the process before and the new distributed system; for example, we have JSON data in JSON Example 1.1:

[{"height":181,"weight":80,"shoe_size":44,"Gender": male},
{"height":177, "weight":70, "shoe_size":43,"Gender": female},

**JSON Example 1.1**

This data will be trained on different devices(nodes) using (JS-Regression) with Multi-Class Logistic function; also we split the data inside the device to (X, Y). X represents {height, weight, shoe size} and Y represent {Gender}; The distributor function sends (X_predict) value to all devices and wait for every device to send (Y_predicate); the final result will be a calculated for all devices, by taking (Y_predicate) from all nodes the distributor will find the average of the results and deliver the requested value.

As you see in the MVP we tried to make it more simple to show that the implementation can be achieved easily; but if we want to make this product profitable and replaceable for current distributed systems and achieve the performance that promised, we should work with different use cases and apply this architecture to one of the current use cases in the market.

We have attached Screenshots for the MVP in the **appendix**; as you see we have our mobile application, after getting the object form the distributer function the application start the computations processing and here it is the training process; and retrieve the results for the distributer function.

## 4 Discussion & limitations

After the implementation of our MVP; the results were very positive; however, proposed the architecture; show us that we have some limitation can appear during the process of the implementation. Limitations can be categorizing to three main part (data limitation, Impact limitations, Study design limitations).

First; the data limitation in our study related to the data normalization; for example, in our MVP; when the distributor layer wants to distribute the data or (JSON objects) through the nodes, data should be normalized; normalization is the technique applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Input data normalization with certain criteria, prior to the training process, is crucial to obtain good results, as well as to fasten significantly the calculations. Because without normalization we could have the wrong prediction depend on Biased data sample that is not normalized with the main data. Data normalization affects the performance error of parameter estimators trained to predict the value of several variables [17].

Second; the study design limitations can be represented with the permissions inside mobile applications on different operating system; for example, the IOS operation system in iPhone or iPod have a lot of restrictions of using the system resources such as the processor or the RAM (memory); if we will publish the application to work only on android; the application will lose user base that uses only iPhone; In 2018, 44.6 percent of smartphone users in the United States used an iPhone, this share was expected to stay at around 45.2 percent in 2019 [18]. So more than 40% percent of our targeted users can't use the application because of the restrictions.

Third; the Impact limitations represented by the Limitation using the Distributed algorithm and the machine learning algorithm; For example, in this MVP we have implement simple linear regression; this algorithm can be implemented on distributed system in the new proposed architecture; because of the nature of the algorithm; it can be separated and it can be recollected using averaging, but some of the algorithm in machine learning need an instance feedback inside the processing unit itself, this can be challenging in the future of the study.

As you see, the data and impact limitations can be suppressed by choosing only the algorithm that friendly with distributed system, and out a condition to all input data in the system to be normalize before. On the other hand, the study design limitation can be overcome by using only android and focus on the opens source systems only.

# 5 Conclusion & Future Scope

One of the main fact now in twenty-one centuries is that "Smartphones will be improved every year", and the competitions between companies are focused on "how to improve the features more and more?". As a smartphone user, we will have in our pocket a machine with a huge amount of features that we will use less 30% during the period of owning the device. In this paper we have purposed new architecture that allows users to monetizes their own Smartphones by providing their processing power to huge network contains other smartphones to share that power and using it in execution and training machine learning model on device and provide the results for central server or provider; in this paper we have implemented an MVP contains mobile application (Android +IOS) that takes data (JSON) objects and do a simple linear regression and resend the date to a NOSQL (Real-time –Firebase database).

From the future implementation for the proposed architecture is to use it be the new layer for data training in IOT systems in the same network; for example, IOT devices generate huge amounts of data and transfer that data to the cloud for further processing. These data include structured data, such as temperature, vibration or multimedia information, such as video, images, and sounds. There is a study that proposed a new concept called "Edge computing" by moving computing ability from centralized cloud servers to edge nodes near the user end [19]. Also another research paper talking about the same concept in machine learning specific "Edge Learning " by deployment of machine-learning algorithms at the network edge. As you see in figure 1.9 the type of learning layers and the key motivation of pushing learning towards the edge is to allow rapid access to the enormous real-time data generated by the edge devices for fast AI-model training, which in turn endows on the devices human-like intelligence to respond to real-time events [22].


**Figure 1.9 (Types of Learning Layers)**

In this scope our future work can be proposed to provide a new layer for computing for IOT systems; for example: let's assume that we have a local network and have IOT sensors to generates data; these data must be sent to the main server in the network;

using our architecture we can provide an execution layer that can communicate with sensors and receive that data and provide the required training method for data and resend it to the main server in the network ready. This solution can reduce a huge amount of processing power in the main server and let the participant in the network (smartphones) share their execution power inside the network. Also, the process can be improved by adding payment form Machine to Machine using IOTA. IOTA is based on a new distributed ledger technology called the Tangle. An emerging technology being developed by the open source community, IOTA is a no-fee transaction and payment settlement platform. It is highly scalable, provides secure data transfer, is quantum immune (secure against quantum computer attack), and is designed for the low network resource requirements common in many machines [20]. So the Smartphones can receive payment using cryptocurrency from the IOT component direct after finishing the process of data training. The main server will provide permission for the smartphones to take the data through WIFI from IOT components and the process will be organized on the local network only.

As you see the future implementation for the proposed architecture are wide; because we are providing a new layer to execute data and do the computation on a distributed system organized by a single entity; the same model can be implemented as is or it can be enhanced to adopt more use cases.

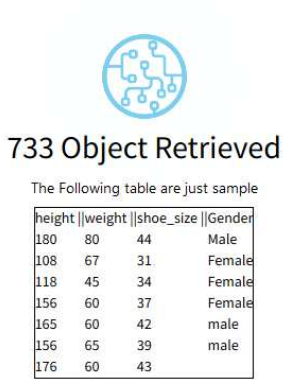
## References


1. Deep Learning in Mobile and Wireless Networking: A Survey Chaoyun Zhang, Paul Patras, and Hamed Haddadi
2. Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, June 2017.
3. Application of Machine Learning in WirelessNetworks: Key Techniques and Open IssuesYaohua Sun, Mugen Peng,Senior Member, IEEE, Yangcheng Zhou, Yuzhe Huang, and Shiwen Mao,Fellow,IEEE
4. S. Han, C. I, G. Li, S. Wang, and Q. Sun, “Big data enabled mobilenetwork design for 5G and beyond,”IEEE Commun. Mag., vol. 55, no.9, pp. 150–157, Jul. 2017.
5. Distributed linear regression by averaging Edgar Dobriban and Yue Shengy October 2, 2018
6. APPLICATIONS David A. Bader NEW MEXICO, USA Robert Pennington NCSA, USA
7. Cluster Computing at a Glance; Mark Bakery & Rajkumar Buyya ;Chapter one ;page 4.
8. Journal of Machine Learning Research 17 (2016) 1-7; MLlib: Machine Learning in Apache Spark; Xiangrui Mengy
9. A Study on Smartphone based Operating System; Kiran Bala; Sumit Sharma; Gharuan Kaur.
10. Top Machine Learning Mobile Apps • Appy Pie. [online] Appy Pie. Available at: https://www.appypie.com/top-machine-learning-mobile-apps [Accessed 16 Apr. 2019].
11. Talos App: On-device Machine Learning Using Tensor Flow to Detect Android Malware Harshvardhan C Takawale ; Abhishek Thakur.
12. Federated Learning via Over-the-Air ComputationKai Yang,Student Member, IEEE,Tao Jiang,Student Member, IEEE,Yuanming Shi,Member, IEEE,and Zhi Ding,Fellow, IEEE
13. Galaxy Note Features; Samsung Phones [https://www.samsung.com/ph/ smartphones/ galaxy-note8/ ].

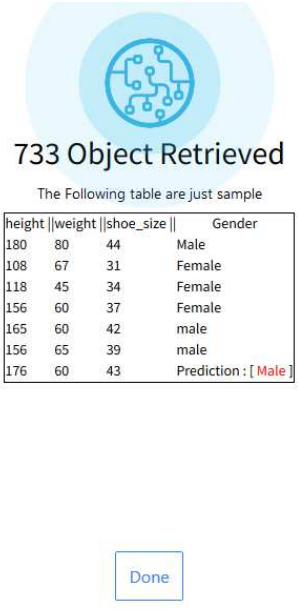
14. [GB+11] Greg Burd. (2011). NoSQL.
15. What is Ionic Framework?; Ionic framework; http://ionicframework.com/
16. Ionic framework angular JS on the rise [https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise/].
17. Importance of input data normalization for the application of neural networks to complex industrial problems; July 1997IEEE Transactions on Nuclear Science 44(3):1464 – 1468; SourceIEEE Xplore
18. Percentage of Us Population that own an iPhone smartphone. [https://www.statista.com/statistics/236550/percentage-of-us-population-that-own-a-iphone-smartphone/]2018
19. Flexible Deep Learning in Edge Computing for IoT ,1Sneha Sureddy, 2K. Rashmi, 3R. Gayathri and 4Archana S. Nadhan
20. D. Schiener. Introduction to iota cryptocurrency.https://blog.iota.org/a-primer-on-iota-with-presentation-e0a6eb2cc621, 2017.
21. Designing Distributed Systems; by Brendan Burns;Publisher: O'Reilly Media, Inc.;Release Date: February 2018; ISBN: 9781491983638
22. G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, “Towards an intelligent edge: Wireless communication meets machine learning,”arXiv preprint arXiv:1809.00343, 2018.

# 6 Appendix


1.1 Screenshot (Start)




1.2 Screenshot (Processing)




1.3 Screenshot (Done)