DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2018

Speed comparison of convolutional neural networks in TensorFlow

JUAN LUIS RUIZ-TAGLE

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Juan Luis Ruiz-Tagle <jlrto@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Anne Håkansson Software and Computer Systems ICT
KTH Royal Institute of Technology

Supervisor

Oskar Henriksson
Head of Recruitment
Slagkryssaren

Abstract

TensorFlow is an open-source library developed by Google for machine learning applications. It can be run in different environments such as desktop computers, servers or even browsers. For running TensorFlow on mobile devices there exist TensorFlow Mobile and the recently launched TensorFlow Lite. The latter is an improved version of the former, which claims to have several performance advantages over its predecessor and is meant to substitute it on the long run. This thesis presents an analysis of how the execution speed of a convolutional neural network varies depending on if it is being run in TensorFlow Lite or in TensorFlow Mobile. Such convolutional neural network is trained to analyze pictures of credit cards. In the thesis work the project is delimited by a set of requirements coming from the stakeholders and a system architecture which satisfies them is presented. The implementation of such system in the form of an application is described and the whole process is finally evaluated. This application is programmed to run the same network with both versions of TensorFlow and is installed on several devices to measure their performance by running some tests. The collected empirical data, which is presented and analysed, shows that TensorFlow Lite's performance is still lower than its counterpart's when running convolutional neural networks. This is due to the early stage of development of the library, which is not optimized enough yet.

Keywords

Machine Learning, TensorFlow Lite, Convolutional neural networks, Performance

Abstract

TensorFlow är en open-source programbibliotek utvecklat av Google för maskininlärningsapplikationer. Det kan användas i olika miljö såsom datorer, servrar eller webbläsare till och med. För att köra TensorFlow på mobilenheter finns det TensorFlow Mobile och den nyligen utsläppt TensorFlow Lite version som är en förbättrad version av den andra. Det påstås att ha flera prestationfördelar över dess företrädare och är menat att ersätta det. Denna uppsats presenterar en analys av hur exekveringshastighet av en convolutional neural network växlar beroende på om det körs med TensorFlow Lite eller TensorFlow Mobile. Sådant neuronät är tränat för att analysera kreditkort bilder. Examensarbetet är gränsat av några förutsättningar som kommer från delägarna och en systemarkitektur som motsvarar dem föreställes. Implementeringen av sådant system i form av en app beskrivs och hela processen värderas. Denna app programmerades för att köra samma nät med bägge TensorFlow versioner och installerades i flera enheter för att mäta dess prestation genom att utföra några tester. Den samlade datan, som presenteras, bevisar att TensorFlow Lites prestanda ligger fortfarande lägre än TensorFLow Mobile när de kör convolutional neural networks. Detta sker på grund av den tidiga utvecklingsståndet av programbiblioteket, som inte är optimerat ännu.

Nyckelord

Maskininlärning, TensorFlow Lite, Neuronät, Prestation

Contents

1 Introduction

Machine Learning has been a research field in Computer Science since the 1980's [1], but it has experienced a major breakthrough during the last 5 years. The emerging of novel machine learning techniques combined with the strength of the powerful processors present in modern computers has proven to give many positive results. Google is one of the many tech companies which has proven to have a real interest in the development of this branch of Computer Science.

1.1 Background

Google has become a pioneer company and an international reference in the field of machine learning. On November 2015 they released TensorFlow to the public, a software library specifically designed to build machine learning applications. TensorFlow is currently available in numerous platforms such as desktop computers, mobile devices and browsers [2]. To be able to run TensorFlow models on iOS and Android devices Google initially developed TensorFlow Mobile. Nevertheless, a reimplementation of the TensorFlow Mobile framework produced TensorFlow Lite, which has several advantages over its predecessor: it has higher execution speed, needs fewer dependencies and it is lighter in terms of memory usage. TensorFlow Lite is still in developer preview, so it is not optimal yet and does not cover all use cases [3].

1.2 Problem

When Google presented TensorFlow Lite in May 2017 [4], it attracted people's attention for its claimed promising performance advantages. It is a matter of interest to make a performance comparison between this framework and its predecessor TensorFlow Mobile in the specific ambit of convolutional neural networks (convolutional neural network abbreviated from now on as CNN). This is achieved by developing an Android test application which runs the same input data through two versions of the same graph, one exported with TensorFlow Lite and the other with TensorFlow Mobile. The problem statement of this thesis is: How does the execution speed of a CNN change when run as a TensorFlow Lite model compared to a TensorFlow Mobile model? Other factors, like the host operating system and the processing power of the device are analysed as well, since they are deeply related with the TensorFlow version run on the device.

1.3 Purpose

The purpose of this thesis is to present the variation of the execution speed of a CNN depending on if it is run as a TensorFlow Lite model or as a TensorFlow Mobile model. The following chapters of the thesis start with an explanation of some theoretical background, and then the investigation process which has been followed during the bachelor thesis is depicted. This consists, firstly, in providing an accurate description of what CNNs are, as well as some general information about TensorFlow and its available implementations for mobile devices. Secondly, it introduces the problem statement and outlines how the two models are compared to each other in terms of speed performance. Finally, it illustrates the results of this comparison and some conclusions are drawn.

1.4 Goals

The goal of the thesis is to provide with a solution in the form of an Android application which allows the comparison of the execution speed of a CNN implemented in both TensorFlow Lite and TensorFlow Mobile, for later performing some tests which permit drawing some conclusions. This general goal can be divided in several subgoals. An initial subgoal is to export the TensorFlow graph done at Slagkryssaren as both TensorFlow Lite and TensorFlow Mobile model files. The succeeding subgoal is the development of a deliverable test application in Android which runs the models over a sample of images. Finally, the third subgoal is to install such app in several devices to run the tests with sets of pictures of different dimensions, The deliverable of this project is the test application which can run and compare TensorFlow Lite and Mobile models.

1.4.1 Benefits, Ethics and Sustainability

The project has been beneficial for both the developers at Slagkryssaren and the author of the thesis since it has deepened the understanding of the TensorFlow framework and the possibilities it offers to mobile devices. The programmed app as a deliverable might be useful in the future to compare the performance of other CNN models running with TensorFlow Lite and TensorFlow Mobile since very few modifications in the code would need to be done to adapt the application to work with a new graph. It might be also beneficial to other students or academics which might have an interest in the relative performance of both versions of TensorFlow. An ethical issue appears regarding the management of the sensitive information which the models could extract from the credit card pictures. For now the CNN graph is trained to generate an output image where this sensitive information appears located and highlighted (a more thorough explanation will follow later on), but it is still only an image that is being produced, so there are no apprehensions to worry about. Of course, if the project is further developed to the point where the data is totally extracted from the image and rendered in form of text some measures should be taken to assure that the privacy of the users is not violated and there are no misuses of such sensitive information.

1.5 Methodologies

The methodology chosen to perform a scientific research is a crucial matter since it guarantees the accuracy and reliability of the results and conclusions drew [5]. S. Rajasekar provides an accurate theoretical explanation of the nature of methodology and research methods. ”Research methods are the various procedures, schemes, and algorithms used in research. All the methods used by a researcher during a research study are termed as research methods. They are essentially planned, scientific and value-neutral. They include theoretical procedures, experimental studies, numerical schemes, statistical approaches, etc [6]. There are two extense categories in which research methods fall, namely qualitative and quantitative methods [7]. Quantitative research consists on proving true or false certain hypotheses or theories by designing and performing experiments. In contrast, qualitative research methods cannot make use of experiments to confirm or discard hypotheses, since the results might not apply for all cases [7]. Therefore, they use other research methods like case studies, surveys or interviews. Strategies are the guidelines for carrying out research, which include organizing, planning, designing and conducting research [5]. Some of these strategies are Experimental Research, Ex post facto Research, Surveys (Longitudinal and Cross-sectional), Action Research, Exploratory Research, Grounded theory, and Ethnography.

After a research method and strategy are chosen, an approach has to be selected. Either inductive research, extrapolating a general conclusion from certain facta; deductive research, making specific conclusions based on a known general law; or abductive, which consists in a combination of the previous two [5]. Methods for data collection and analysis should be carefully picked as well. The former include the above-mentioned experiments, case studies, surveys and interviews among others. The latter are used to process the data accordingly to the chosen research methods and include statistics, computational mathematics, coding and analytic induction, etc. [6]. Finally, quality assurance methods should be applied to guarantee the validity, replicability and reliability of the research.

For this project, the selected methodology takes a deductive approach, based on a positivistic philosophical assumption. An experimental research design is chosen, elaborating a test which will be conducted in several devices and will make measurements of the inference time for the TensorFlow models. Quantitative methods will be used to collect the data, and all of it will be analysed statistically and presented in form of results. To be able to draw conclusions from the data its quality will need to be assured. The methodologies will be extensively elaborated later on in the report.

Quantitative methods have been chosen to compare the performance of the two TensorFlow frameworks for mobile devices. The measurement which will be collected is the inference time for the CNN models to run, given exactly the same input. This measurement will be done setting timers in the device's processor through the Java API before and after the execution of the CNN, to calculate the time interval subsequently. These timers are accurate enough to provide a realistic measure which lets a comparison between measurements be made. A deductive method will be utilized to reach these conclusions from the collected data.

1.6 Delimitations

The study has been delimited in the subsequent manner. TensorFlow Lite and TensorFlow Mobile, and no other frameworks for machine learning in mobile devices, were given in the requirements to conduct a performance test. To test the performance of the TensorFlow Lite framework in comparison with its former version TensorFlow Mobile, the specific context of convolutional neural networks has been chosen. No other types of networks or graphs are considered in this study. Also, there exist delimitations in the number of devices selected to run the performance tests. Four Android devices have been used for testing purposes, namely a Samsung Galaxy S4, a Samsung Galaxy S5, a Samsung Galaxy S8 and a Motorola Moto G5 Plus. These telephones were chosen because they represent different class categories. Some of them are modern flagships while others belong to a middle-class category. The direct influence that other factors might have such as memory consumption of the CNNs or the processing power of the host device is not considered in this study. The number of sizes of different images to test on the models was delimited to 3 sizes, of 160, 240 and 320 pixels-sided square images.

1.7 Stakeholders

Slagskryssaren is a tech agency based in Stockholm which provides software solutions to their clients by building digital services based on the most recent available technologies [8]. This thesis builds on top of one of their projects, codenamed SKCC (Slagkryssaren credit card). They shared the project repository and made available some devices used for testing purposes. I have been in contact with them regularly throughout the project and they have guided me and answered my questions at all times.

1.8 Outline

The structure of the next sections of the report is the following: in chapter 2 the theoretical background concerning CNNs and TensorFlow Lite and Mobile will be presented. Then the chosen methodologies for performing the research will be explained in chapter 3. After that, in chapters 4 and 5 the whole research process will be accurately described, pointing out the application of the methods when its relevant. Posteriorly, together with the results gathered some reasoning will be presented in chapter 6. Finally, in chapter 8 a formal conclusion will reflect upon the whole project.

2 Theoretical Background

After the concise background description given in the previous section, a more elaborate explanation of the theoretical background of this thesis project will be furnished in the following paragraphs. First, artificial neural networks will be briefly introduced and then, more specifically, convolutional neural networks will be explained with a higher level of detail. Then the TensorFlow framework will be presented, as well as both of its mobile versions TensorFlow Lite and TensorFlow Mobile. These will be compared and their relation described.

2.1 Artificial Neural Networks

Artificial neural networks are an extremely important branch in the Artificial Intelligence and Machine Learning research fields. Neural networks are systems of interconnected nodes which are programmed to find patterns in data by themselves after having gone through a training process, which consists of feeding the network with a large number of samples [9]. This process tries to mimic the way the brain operates, and that is where its name comes from: each node simulates a neuron which sends pulses to the other ones in the network. Neural networks have been a matter of research for more than 40 years now [10], therefore they exist in the form of countless types and classes. The main procedure followed by neural networks to make an output estimation for new input data is described below, and more specifically the functioning mechanism of CNNs will be presented in the following section more thoroughly [10].

In the images in Figure 1 we can see the basic structure of an ANN (artificial neural network). The left image depicts a node, the basic unit of the network. Each node performs a dot product between the input and weight values, that is, it takes the input values ($x_1 ... x_n$), multiplies each one by its respective weight ($w_1 ... w_n$) and sums everything up. The result of the dot product is passed to the activation function. If the result of this mathematical function is above a certain threshold, the node will fire and send an output value. In the right image can be seen a set of nodes structured in layers, with 4 in the input layer, 4 in the hidden layer and 2 in the output layer [11]. The input values are placed in the input nodes, which sim-

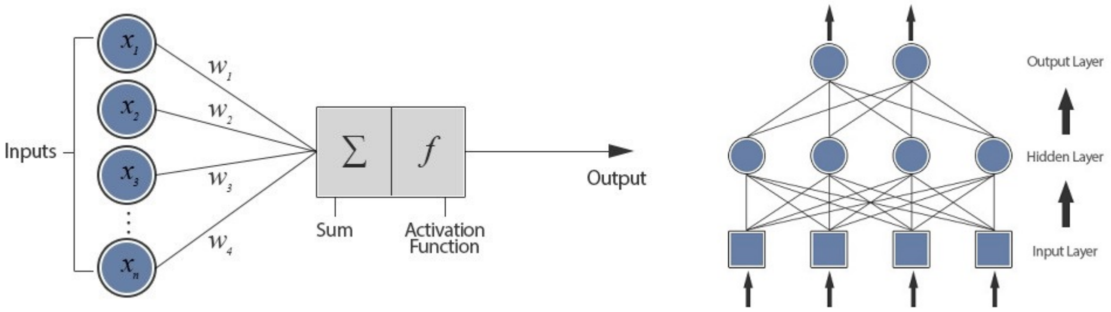


Figure 1: A perceptron and a neural network

Source: Pokharna, Harsh. For Dummies—The Introduction to Neural Networks we all need ! (Parts 1 and 2) URL: https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb. Consulted: 24/4/2018

ply refer them to the nodes at the hidden layer. Each hidden node computes an output and forwards it to the output layer nodes, which make a final computation and generate a result [12]. Training the network consists of finding the best values for all the weights which optimize the results for most given cases. The presented example is very simplified, however, ANNs can have many different structures, with several hidden layers and other elements, such as a bias value [11].

2.2 Convolutional Neural Networks

Convolutional neural networks are a specific kind of ANN usually used in computer vision and they usually take an image as an input. The general idea is that in each layer of the network the image is shrunken and processed by the activation functions in the neurons, and a new one is outputted which has partially abstracted any patterns found [13]. As the image goes from layer to layer, it becomes more abstract until it is finally tagged as belonging to some class.

Regular ANNs might work fine for small images, but do not scale well to bigger ones. "For example, an image of more respectable size, e.g. $200 \times 200 \times 3$ (3 stands for the RGB values of each pixel), would lead to neurons that have $200 \times 200 \times 3 = 120000$ weights. Moreover, we would almost certainly want to have several of such neurons, so the parameters would add up quickly. Clearly, this full connectivity is wasteful and the huge number of parameters would lead to overfitting." [11]

Since it is not computationally affordable to establish so many connections between nodes a new structure for the network has to be designed, with a new arrangement of the layers. Each layer has neurons disposed in 3 dimensions: width, height, and depth. Each neuron in a layer is only connected to a small group of neurons in the previous layer, which refers to adjacent pixels in the original image. What each layer does is to transform the 3D input volume to a 3D output volume of neuron activations [11]. [Figure 2]

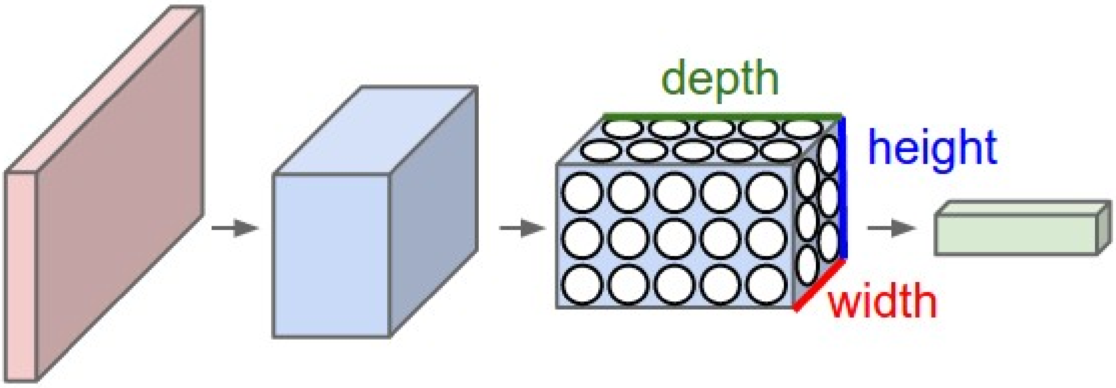


Figure 2: A CNN transforms the 3D input volume from layer to layer.

Source: Karpathy, Andrej. CS231n Convolutional Neural Networks for Visual Recognition [11]

2.2.1 Types of layers

- **Convolutional layers**: These layers are the principal building blocks of the CNN. The way they operate is by applying a set of filters to the input image trying to recognize patterns. These filters are quite smaller than the input image, (square matrices of 5×5 pixels, for example) so they have to slide or ”convolve” from left to right and top to bottom along the whole image moving usually one pixel at a time (stride of 1). The dot product is computed between each filter and the portion of the image that is currently on focus, so the more similar they are to each other the higher the result will be and the greater influence it will have in the next layer’s convolutions [11]. [Figure 3]

- **Pooling layers**: the function of the pooling layers is to gradually reduce the width and height of the input 3D volume as its depth keeps increasing due to the filtering process [12]. The most common pooling method is applying a filter of 2×2 pixels with a stride of 2 using the *MAX* operation. This means that the image is divided into squared groups of 4 pixels and the maximum value of each group is taken, discarding the other three pixels, so its area is reduced by 3/4 [11]. [Figure 4]

- **Fully connected layer**: All the nodes in the last layer have connections to all the activation inputs in the previous one. At this point, the image's dimensions have diminished and it has become small enough through the pooling process to feasibly compute a connection from each node to each pixel [13].

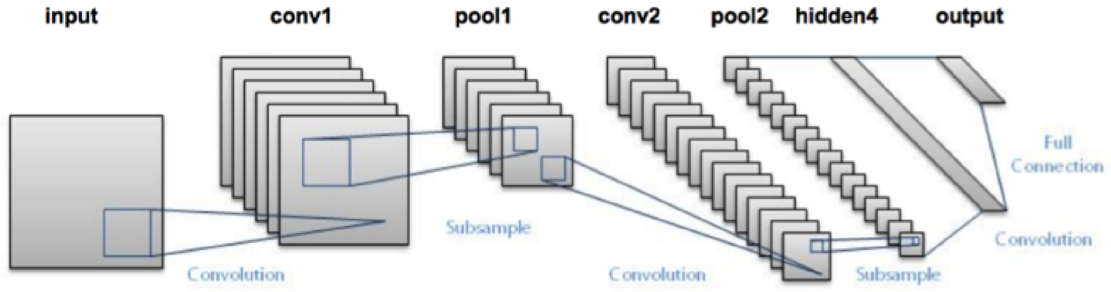


Figure 3: Filter sampling at convolutional layers

Source: Dishashree, Gupta. Architecture of Convolutional Neural Networks (CNNs) demystified. URL: https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/ Consulted: 27/5/2018

2.3 TensorFlow

TensorFlow is a machine learning library developed by Google to satisfy their needs of systems capable of building and training neural networks [14]. Today it is used both for investigation purposes as well as in many of Google's own products. TensorFlow has been continuously expanding since its very beginnings. It was opensourced in 2015 and now the library is implemented in several programming languages and optimized for its use in different domains [2]. To run TensorFlow applications in mobile devices Google released first TensorFlow Mobile, which later evolved into TensorFlow Lite. These two frameworks are described and contrasted, since the main goal of the project is to compare their performance at the task of running CNNs.

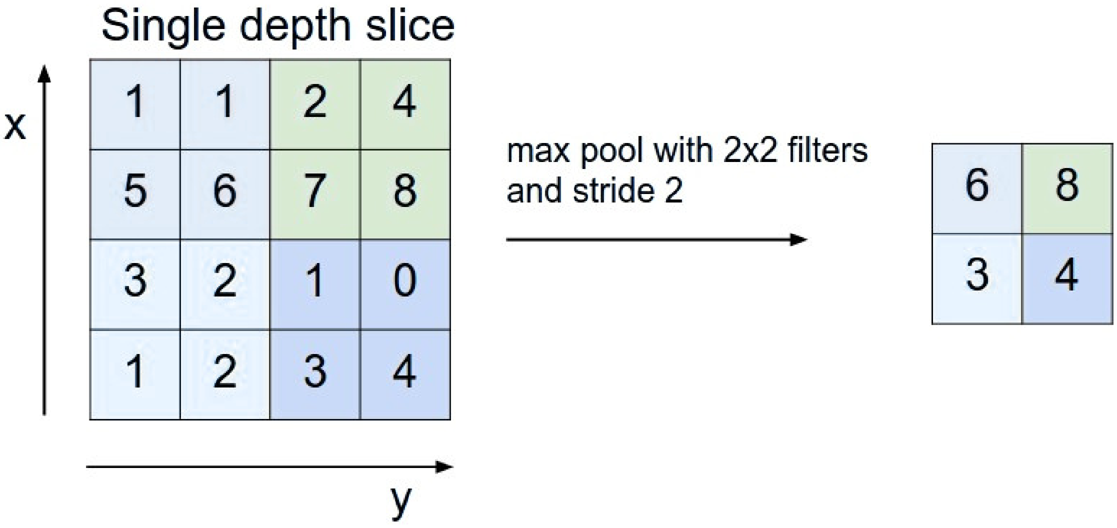


Figure 4: *MAX* operation usage at pooling layer.

Source: Karpathy, Andrej. CS231n Convolutional Neural Networks for Visual Recognition [11]

2.3.1 TensorFlow Mobile

Running ANNs is a very computationally demanding task that has always been entrusted to powerful computers, but Google planned to take a step further and make the less powerful mobile devices capable of running them as well, since this would open a door with lots of potential. TensorFlow Mobile was developed to be able to run TensorFlow graphs on iOS and Android devices [15]. When a TensorFlow graph is trained and ready for deployment it can be exported as a TensorFlow Mobile model with the protobuffer file format. A protobuffer is a file which contains all the relevant information about a graph [16]. TensorFlow graphs are defined by their overall structure and the values of each of the weights. These two parts of the graph are usually stored in different files for convenience reasons, but they need to be put together in a protobuffer file to be used by TensorFlow Mobile on a mobile device. There are two types of protobuffer files: text format (*file.pbtxt*) which is a human-readable form, and binary format (*file.pb*) which is not readable but much smaller file [15].

2.3.2 TensorFlow Lite

TensorFlow Lite is a natural evolution of TensorFlow Mobile, meant to substitute it in the long run. It was first released in November 2017, and it is still in a development stage [17]. It presents many advantages over its predecessor: it generally provides better performance and a smaller binary file, but it does not cover all use cases yet. ”TensorFlow Lite uses many techniques for achieving low latency such as optimizing the kernels for mobile apps, pre-fused activations, and quantized kernels that allow smaller and faster models” [17]. TensorFlow Lite’s file format is the FlatBuffer (*file.lite*), an ”open-sourced, efficient cross-platform serialization library. It is similar to protocol buffers, but the primary difference is that FlatBuffers does not need an unpacking step to a secondary representation before you can access data, often coupled with per-object memory allocation.” [17]. It has also the capability of interacting with the Android Neural Networks API, designed specifically to boost machine learning computations [18].

2.4 Related work

In the following section some examples of studies related to the topic of this thesis will be presented.

2.4.1 Image Classification, Deep Learning and Convolutional Neural Networks

In this comparative study of machine learning frameworks performed at Karlstad’s university the authors analyse the performance for training netural networks in both TensorFlow and Microsoft CNTK, two machine learning frameworks available to the general public [12]. For the TensorFlow part, they chose tu use the third-party API Keras for specific tasks at the front-end. Their results are that CNTK performed better at the benchmarking tests, but TensorFlow with Keras is more intuitive and easy to work with.
This study is relevant since it gives some insights about other existing available frameworks for machine learning besides TensorFlow. Knowing about different posible setting up configurations for running neural networks can be useful for the present thesis.

2.4.2 ImageNet Classification with Deep Convolutional Neural Networks

The ImageNet classification paper by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton represents one of the major breakthroughs in computer vision during 21st century. In 2012, a group of researchers designed deep convolutional neural network with five conv layers, some max pool layers and two globally connected layers [19]. The idea was to participate in the annual ImageNet Large-Scale Visual Recognition Challenge, where the participants present algorithms that have to classify input images over 1000 categories. With Alexnet (the name they gave to their CNN) they achieved a top 5 test error rate of 15.4%, considerably smaller than other solutions presented until that moment. This paper is relevant to the thesis since it suggest an efficient architecture to design CNNs.

2.4.3 TensorFlow: A System for Large-Scale Machine Learning

In this paper, the authors describe the TensorFlow dataflow model and demonstrate the compelling performance that TensorFlow achieves for several real-world applications [2]. The applicability of TensorFlow models to large scale systems requires a framework architecture which is optimal by every mean, since every single operation can become very computationally expensive in a large scale context. TPU (TensorFlow Processing Units) are presented with detail in the paper. They are a type of GPU specifically designed to train and execute neural networks, which saves enormous amounts of time in training graphs. This paper is relevant to the thesis since it informs about the possible approaches to handle massive inference requests for convolutional neural network systems.

3 Methodologies

The upcoming chapter aims to describe the scientific and software development methods used for the realisation of this degree project. Some insight on the strategies followed to do research, as well as the methods selected to collect and analyse the data will be exposed. The software development methods chosen to create the deliverable are presented also.

3.1 Research methods

”Research methods and methodologies are processes, or particular courses of actions, that assure the quality of results of the research” [5]. As mentioned in the introductory chapter, their election is of a vital importance since it can affect the results and drewed conclusions of the research process.

The kind of research being done within this project is clearly quantitative since time is being measured, so its research method is ”experimental”. This research method is often used when investigating systems's performances [5]. The relative performance of two frameworks (TensorFlow Lite and TensorFlow Mobile) is being studied, and this one is determined by measuring the time taken for some computations to take place, which is a quantitative measurement. For this project, Positivism is chosen as the driving philosophical assumption, since it suits situations where a phenomenon needs to be probed in order to deepen in its understanding (such as in the present case).

3.2 Research approaches

The research approach chosen is mainly deductive. Deductive reasoning is the process of reaching a set of certain conclusions starting from a set of premises. If the premises are true and the logical rules are rigorously followed, then the arriving conclusions are necessarily true as well [20]. A classical example of a deductive reasoning is the following:

All men are mortal. (First premise)
Socrates is a man. (Second premise)

Therefore, Socrates is mortal. (Conclusion) [21]

Deductive reasoning suits this research since by measuring how much time do the measurements take a logical conclusion can be drawn. An initial hypothesis is stated, TensorFlow Lite has a better performance in terms of time than its counterpart TensorFlow Mobile, since it has been developed to subsitute the latter. This theory needs to be severely tested using quantitative methods, and it will be accepted or discarded depending on the results of the tests.

3.3 Research design

Selecting the correct research design or methodology to any study is an all-important task. This design will set the guidelines for organizing, planning, designing and conducting the research. The most common ones in quantitative studies are ”ex post facto” research, experimental research, surveys and case studies. Furthermore, qualitative studies share as well some of these (surveys or case studies) but have as well some particular ones, such as action research, exploratory research, grounded theory and even etnography [5].

After taking all research strategies into consideration, experimental research is opted as research design in this project. Experimental research concerns control over all factors that may affect the results of an experiment [20]. As the research method, Experimental design verifies or falsifies hypotheses and provides cause-and-effect relationships between variables, i.e., correlations between independent and dependent variables [5]. In the present case there is control over the different factors that can influence the results of the experiment, namely the size of the image to be infered by the models, the device's operating system and its processor's computing power (the tests are performed in several devices with different characteristics).

3.4 Data collection

The methods for data collection are the ones used to gather all the information that will be posteriorly analyzed and from which conclusions will be drew. Therefore it is vital to be able to guarantee precision and accuracy in the data collection, so that the results of the experiment are not biased. The most commonly used methods for quantitative research are experiments, questionnaire case study, and observation [5]. The chosen method for data collection is the experiment, a quantitative method used to gather samplings of large sets of variables. Experimental research often handles huge ammounts of data [20], so this method suited this thesis since many datapoints are collected. The order of magnitude of the dataset's size is adapted to the scope of the project. 150 measurements of time inference for each TensorFlow model are performed in each device, with images of three different pixel sizes (50 measurements each). Measuring the inference time of both models gives an idea of their real performance. Besides, it is worth mentioning that the interview, a qualitative method for data collection, is briefly used to collect the information about the requirements from the stakeholders.

3.5 Data analysis

The data analysis methods are used to analyse the collected material. It is the process of inspecting, cleaning, transforming and modelling data. It supports decision-making and drawing conclusions [5]. There exist several methods used for analysing quantitative data, such as statistics or computational mathematics. Statistics is the appropiate method to use in the present investigation since it summarizes the dataset into some values which are easier to work with [20]. Parameters like the media or the standard desviation for each of the tests run will be calculated in order to have a degree of confidence in the conclusions drew.

3.6 Quality assurance

Assuring the quality of an investigation means to validate and verify the research material. The quantiative research, with a deductive, approach, must apply and discuss validity, reliability, replicability and ethics [5] [22] [23].

3.6.1 Validity

The validity of the research confirms that the measurements are correct and accurate. As it will be later described, this validity is achieved by looking at the Android

Profiler in Android Studio, a tool that lets us check the real execution time of each invoked function in an Android app. The measurement which is collected is the inference time for the CNN models to run, given exactly the same input. This measurement is done by setting timers in the device's processor through the Java API before and after the execution of the CNN, to calculate the time interval subsequently. These timers are accurate enough to make a realistic measure which lets a comparison between measurements be made.

3.6.2 Reliability

Reliability means the stability of the measurements and the consistency of the results for every testing [5]. The repeated measurements obtained throughout this project are reliable since they remain moreless stable around fixed values (the variance is very low).

3.6.3 Replicability

Replicability is the possibility of another researcher to replicate the experiment obtaining the same results. This is an important property of a project's methodology as it strengthens the integrity of the conclusions drawn in the research. The experiment is totally replicable, since the only things needed are a set of Android devices, installing the already implemented application on them and running the tests.

3.6.4 Ethics and sustainability

Ethics and sustainability are the moral principles in planning, conducting and reporting results of research studies [24]. Ethics covers protection of participants, maintenance of privacy, avoiding coercion and having consent in written form, and treating material with confidentiality [5]. Sustainability extends to the consequences and impacts that the project might have on the enviroment and society. Ethics and sustainability in this project were introduced in section 1.4.1. The management the sensible information which can be found on credit card pictures (card number, owner's name, expiration date) should be taken into consideration. However, its is important to understand that this project never extracts the information on credit cards in a text format (it just generates a picture) and the credit card images used to train and test the network are computer generated.

3.7 Software development methods

There exist multiple software development methods for software project to effectively organize the work, implement all the requirements and reach the goals in time. All these methodological approaches can be classified in several types of models, like V-models, waterfall models or agile methods [25], like Extreme Programming [26] or Scrum [27]. Several methodological approaches were considered for the execution of this project. Some of them were discarded, like the waterfall or V-models, since they are designed to be used in large scale projects and take into consideration modifications of the requirements in the middle of the project. An important point to be taken into consideration is that only one person is in charge of this project, even if individuals from Slagkryssaren provided with counsil and help. This makes the whole development process much more flexible, since there is not a dependency of other people and work can be done at any place and at any time. For this reason, Scrum was chosen as software development method, because it has an agile approach, but it had to be adapted to a 1 person team [27]. Regarding the development enviroment and version control system, Android Studio together with Git were chosen.

3.7.1 Scrum

Scrum is an Agile framework for completing complex projects. Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work [28]. It is designed to break down the project into a set of actions which are iteratively run over through an specific time period, called sprint (very flexible, from 1 week to 1 month, but the duration remains constant along the project). The progress is tracked in short meetings scheduled at certain fixed moments along the day. These called daily scrums, and their purpose is to reflect upon the work done since the previous meeting and to readjust each participant’s tasks and goals. In this way, what is important is always in mind and the team is aware of its performance [29]. The participants of the project are called the Scrum team. In this team, several roles exist: the product owner, which represents the stakeholders and is in charge of assuring that all the requirements are fullfilled; the scrum master, which has the responsability of following the Scrum rules, and makes sure that there are no impediments or distractions which can affect the accomplishment of the goals. Finally, the development team, in charge of developing and delivering the potential product.

3.7.2 Android and Android Studio IDE

TensorFlow Mobile and TensorFlow Lite are available on two mobile platforms, iOS and Android. Android presents several advantages for the execution of this thesis and has been therefore chosen as host operative system [30]. This advantages consist overall in the author's previous experience in the Android development framework, as well as the better integration capabilites that TensorFlow has with Android since they are both developed by Google. Some special functions such as the usage of the Neural Networks API were exclusively available on Android. The IDE chosen to develope the application with is Android Studio, which is built on JetBrains' IntelliJ IDEA software and designed specifically for developing Android applications [31].

3.7.3 Git and Gitflow

Git is the version control system used through out the project, together with the Gitflow workflow, being the repository with the code hosted in Bitbucket. Git tracks changes in the files stored in a given repository and coordinates the work done of them across teams of several people. It is most commonly used to track code files, but in practice it can be used to keep track of the changes in any kinds of files. Git allows to go back to individual milestones called commits, which present the different versions of the code. This commits do not necessarily follow a line in time, they can aswell divide in different branches with their own changes [32]. Gitflow is a workflow used with the Git version control system. It consists in a conventional set of rules for using Git in order to smooth and optimize the development process. It defines a branching model which focuses on product releases, with two principal branches: master (where the official released version of the program lies) and develop (where new functionality are implemented and tested).

For each individual new feature a new branch is created which keeps the track of its development. When the new feature is ready it can be then merged back on to the develop branch, and when a set of new features are properly tested and seem to be working a new release is published in the master branch [33]. Using these version control tools was required by Slagkryssaren since they are the ones they use for all their other projects at the company.

4 Project design

In order to manage the project properly, the Scrum method was used, dividing the whole work in a series of 10 sprints, each of them lasting 9 days. For each sprint a set of goals were defined, usually including milestones that had to be surpassed in the development of the project. The first 8 sprints involved the main research and development of the application in the following way: 1 sprint invested in literature study and adaption to the new technologies, 1 sprint in exporting the TensorFlow models, 4 sprints in the development of the application and 1 sprint in testing the application in different devices and gathering the data. The last two sprints were used for writing this report. When a sprint came to an end, its results were evaluated and the tasks which had not been accomplished were posponed to the coming sprint.

4.1 Project requirements

At Slagkryssaren there is a research project, SKCC, which aims to build an API that receives pictures of credit cards and returns the data printed on them (i.e. the card number, expiry date, etc). To extract the information they use a convolutional neural network running in TensorFlow, but they are interested in making the API available for mobile devices, so the natural action is to reimplement the model with the mobile versions of TensorFlow. The recently launched framework TensorFlow Lite appears to be an attractive solution to them.

To understand which are the requirements demanded by Slagkryssaren, the interview is used as a data collection method. An interview with them was arranged, where they expressed their demands. The requirements consist in exporting the CNN to both TensorFlow Lite and TensorFlow Mobile file formats for later embedding them in an application which can compare their speed running tests. Another requirement is that such application is to be tested in different devices with different operating systems, in order to see how the performance of the two models varies. Images of different size should be fed into the system to see if the performance varies depending on this. Other requirements imposed by Slagkryssaren are that no part of the project's repository (including the dataset generator and other scripts) can be shared with third-party individuals alien to the company. Some software development requirements were stated as well: the version control system to keep track of the progress would be Git together with the Gitflow workflow, setting the parent repository in Bitbucket.

4.2 System architecture

After stating the project's requirements pointed out by Slagkryssaren the system is ready to be designed in the form of an experiment (it is the research design chosen in the methodology). The system has several interconnected elements which can be seen in Figure 5. First, the TensorFlow graph is exported in the two formats and compiled in an Android app using TensorFlow's APIs. Then, the credit card image generator already implemented by Slagkryssaren is used to create a set of new images, and these are fed to the two versions of the graph. The inference time taken by each of them to produce an output is measured and stored in the device. To achieve reliability in the measurements the sets of images had to be big enough, and of different sizes to check how the input size affected the performance of the models (see requirements above). Three sets of 50 images, with dimensions 160×160, 240×240, and 320×320 pixels would be generated for this purpose. Other variables to be considered are the device's computational power and the operating system being run on it. For this purpose, four devices which represent different areas of these variable's spectrum were selected for running the experiment, namely a Samsung Galaxy S8 (Android 8.0), a Samsung Galaxy S5 (Android 5.0.1), a Samsung Galaxy S4 (Android 4.4.2) and a Motorola Moto G5 Plus (Android 7.0).

4.3 Initial state of the project

The project in which Slagkryssaren is working on consists, as aforementioned in the introduction, in developing a TensorFlow graph which could analyze a picture of a credit card and return the information printed on it. This project started in June 2016 but not many resources were invested into it, so it is still in an early phase when first shared with me. The TensorFlow graph which they had written

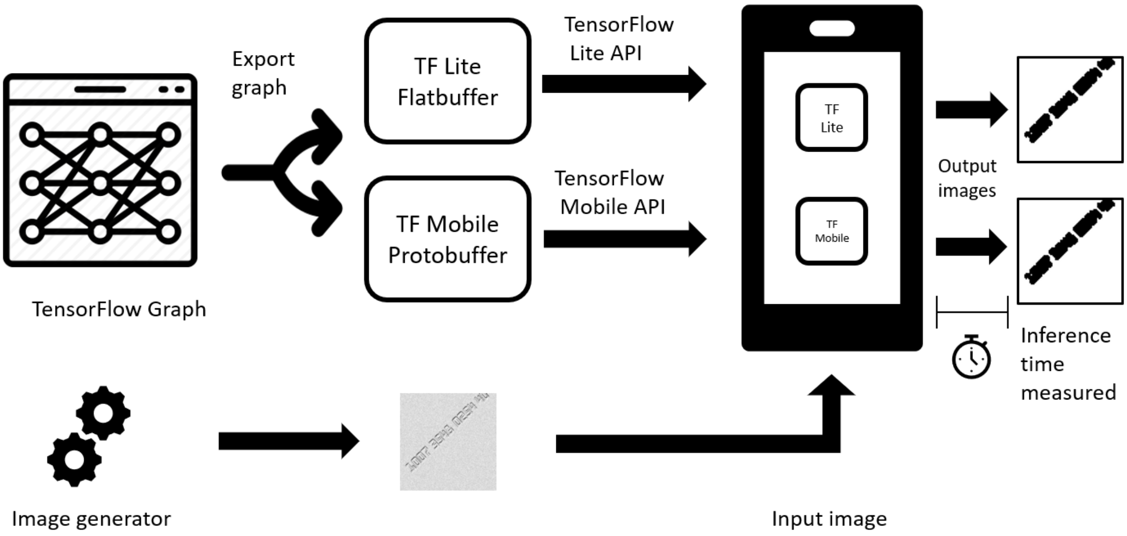


Figure 5: System architecture

Source: The TensorFlow Graph picture inside the diagram is obtained from https://d30y9cdsu7xlg0.cloudfront.net/png/1705433-200.png

at the time consists in a CNN which removes the background noise and returns a black and white image where only the credit card number can be seen. To create a dataset to train this graph they programmed the credit card image generator which produces fairly realistic pictures. Figure 6 is an example of one of the thousands of pairs of images generated to train the network. The left image is fed into the input and the right image is the target, that is, what the network is expected to produce. The graph had been trained on them and the results were satisfying.

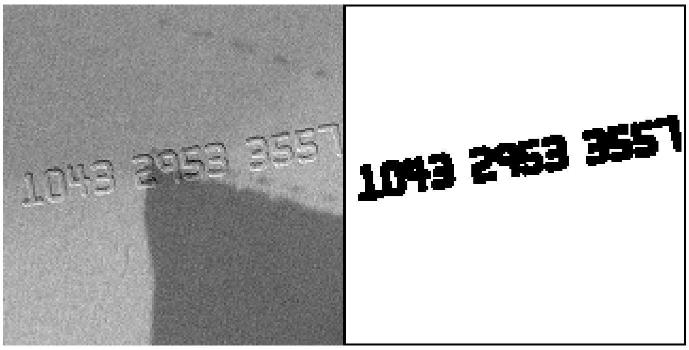


Figure 6: Example of input (left) and target (right) images used to train the CNN.

Source: Images produced with the credit card image generator developed at Slagkryssaren

Figure 7 reveals the inner structure of the CNN. *Conv2d* is the input layer, backed by *Conv2d_1* which adds a bias, then it comes a convolutional layer *Conv2d_2*, followed by a MAX pooling layer with a stride of 2 and a filter of 2x2 pixels. The image is now reduced to a quarter of its size. Afterwards, it goes through 6 more convolutional layers (*Conv2d_3*to *Conv2d_8* both included) with a stride of 1 and a filter of 5x5. Convolutional layers *Conv2d_2-8* use *relu* as activation function. In the last convolutional layer *Conv2d_9* the filter is changed to 1x1 and activation function to a *sigmoid*. The data is then sent to the Adam module which performs some optimizations which are beyond the scope of this thesis. The network has not a fully connected layer in the end, though, since the output of the graph is not yet meant to make a classification, but still generates a picture. All these values were chosen for the configuration of the network since they are optimal standard values in CNN architectures [11]. In some of the arrows which connect the different layers can be appreciated the size of the 4-dimensional float vector which represents the image. In other arrows, the number of tensors between layers is pointed out. This convolutional network developed by Slagkryssaren was adequate for studying the performance of the TensorFlow frameworks for mobile devices, so it was selected.

In this chapter the project requirements have been stated, together with the designed architecture of the system which fullfills all of them. The initial state of the project ”as is” is presented as well.

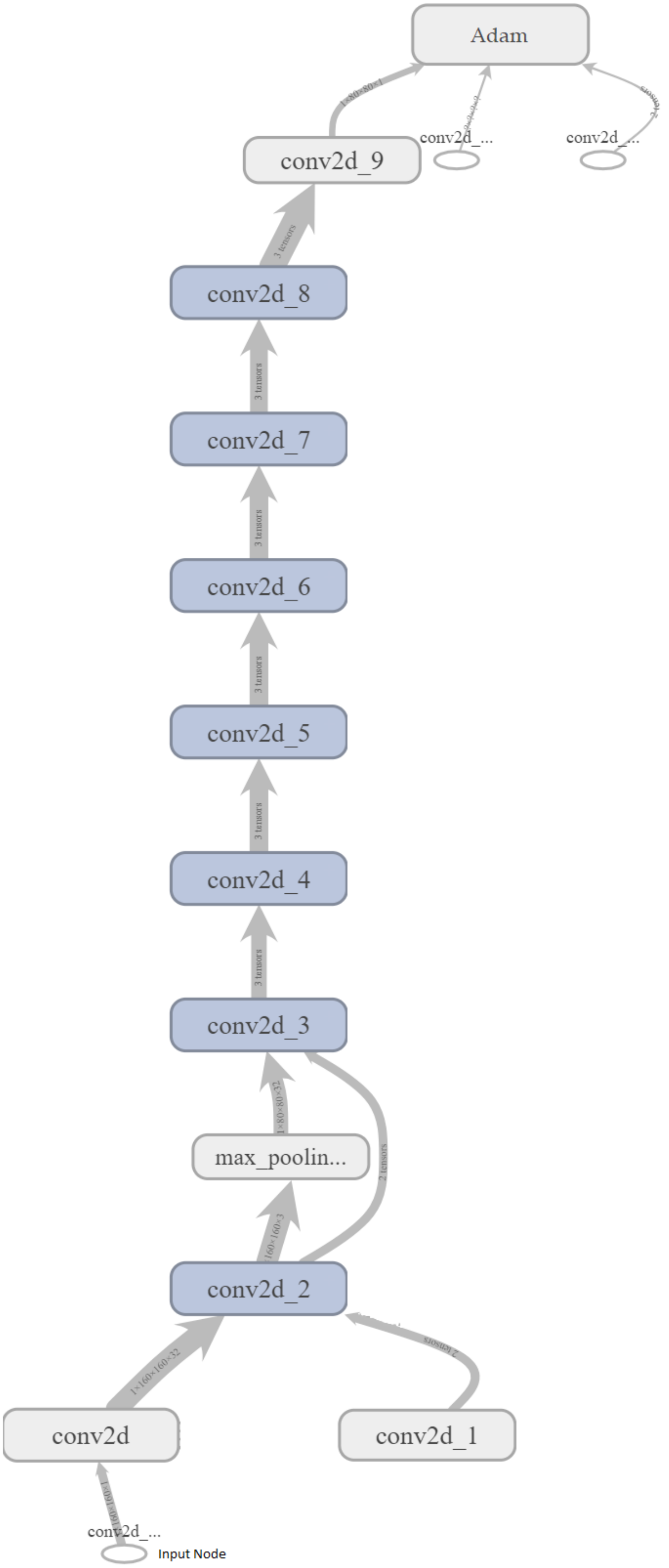


Figure 7: Graph of the CNN used on the credit card images

Source: Exported from TensorBoard

5 Implementation of the test application

The implementation of the test application consists of several steps. The first one is to export the graph from the TensorFlow framework in the computer as both TensorFlow Mobile and TensorFlow Lite models. The second is to develop the Android application which would iteratively feed the generated pictures into the models and register their inference time. These two steps are described in the following sections, whereas the final stage of testing it in the set of devices will be explained later on.

5.1 Exporting the graph

After the research was done it was time to export the CNN as both a TensorFlow Mobile model and a TensorFlow Lite model to be able to use them in an Android application. The graph had to be ”frozen” first, that is, all the variable values for the weights had to be converted into inline constants [34]. The function *freeze_session()* contained in *freeze_graph.py* in the TensorFlow library allows us to do this. Then it is time to write the frozen graph into a file, and this will be done in different ways depending on the TensorFlow version which is going to run it and on the output format desired for the file.

To get a model runnable by TensorFlow Mobile there were different possible output formats for the file, but the raw protobuffer format (with extension *.pb*) was preferred. The function *write_graph()* is invoked, specifying the path and the output format of the model, and it generates a file. For the TensorFlow Lite model, the process followed is slightly different, since there is an intermediate step between freezing and writing the graph to a file. The frozen graph has to be converted first to a TensorFlow Lite graph using TOCO (TensorFlow Lite Optimizing Converter). This is done by invoking the function *toco_convert()*. The resulting graph is then written to a file with flatbuffer format (*.lite*).

5.2 Development of testing application

Once the two models had been retrieved the development of a test application could start. The application's purpose is to be able to quickly test the relative performance of both models by running them over a set of images on different devices. The time it takes to run the models for each image is stored and displayed in a nice graph to get an overview of the performance of each model. All this inference data can be easily exported from the app to a text file for posterior analysis. The development of the Android application took a considerable portion of the work time used for this project. My personal experience with the Android environment was somehow limited, and the TensorFlow Lite and Mobile dependency libraries for Android were totally new to me. TensorFlow Lite 0.1.7 and TensorFlow Mobile 1.8.0 are the versions used in this project. The app was divided into different activities:

- **MainActivity**: runs the models and displays a grid with the pictures to be analyzed. When any of the pictures are clicked, the models are run on that picture and the result is displayed in DisplayActivity. [Figure 8a].
- **DisplayActivity**: after running the models it plots the picture, the target (expected result), and both of the predictions of the TFLite and TFMobile models, together with their respective inference times. [Figure 8b]
- **StatsActivity**: plots a chart where it is represented the inference run time of both models for each picture. Each time a model is run on a picture, a new point is added to the chart. [Figure 8d]
- **SettingsActivity**: used to configure the settings for the test, such as the number of images selected to run the model on, which model or models to use and the usage of the Neural Networks API if available. The data on the chart can be exported from here to a text file (Export button), or totally erased as well (Erase button). [Figure 8c]
- **CameraActivity**: lets the user take a picture with the device camera and run the models on it.

A parent Model class has been also implemented with some abstract methods, from which specific TfLiteModel and TfMobileModel classes are extended. An Adapter class for the picture grid and a BaseActivity have been implemented as well. TfMobileModel and TfLiteModel classes have similar implementations of the *predictImage()* method, which is used to actually run the model on an image. This function converts the image in Bitmap format to a float array, starts then a time counter, feeds the float array to the model, stops the timer when the computation has finalized and reconstructs a Bitmap image from the output float array. Both implementations follow.

Listing 1: code in *predictImage()* used for TfMobileModel

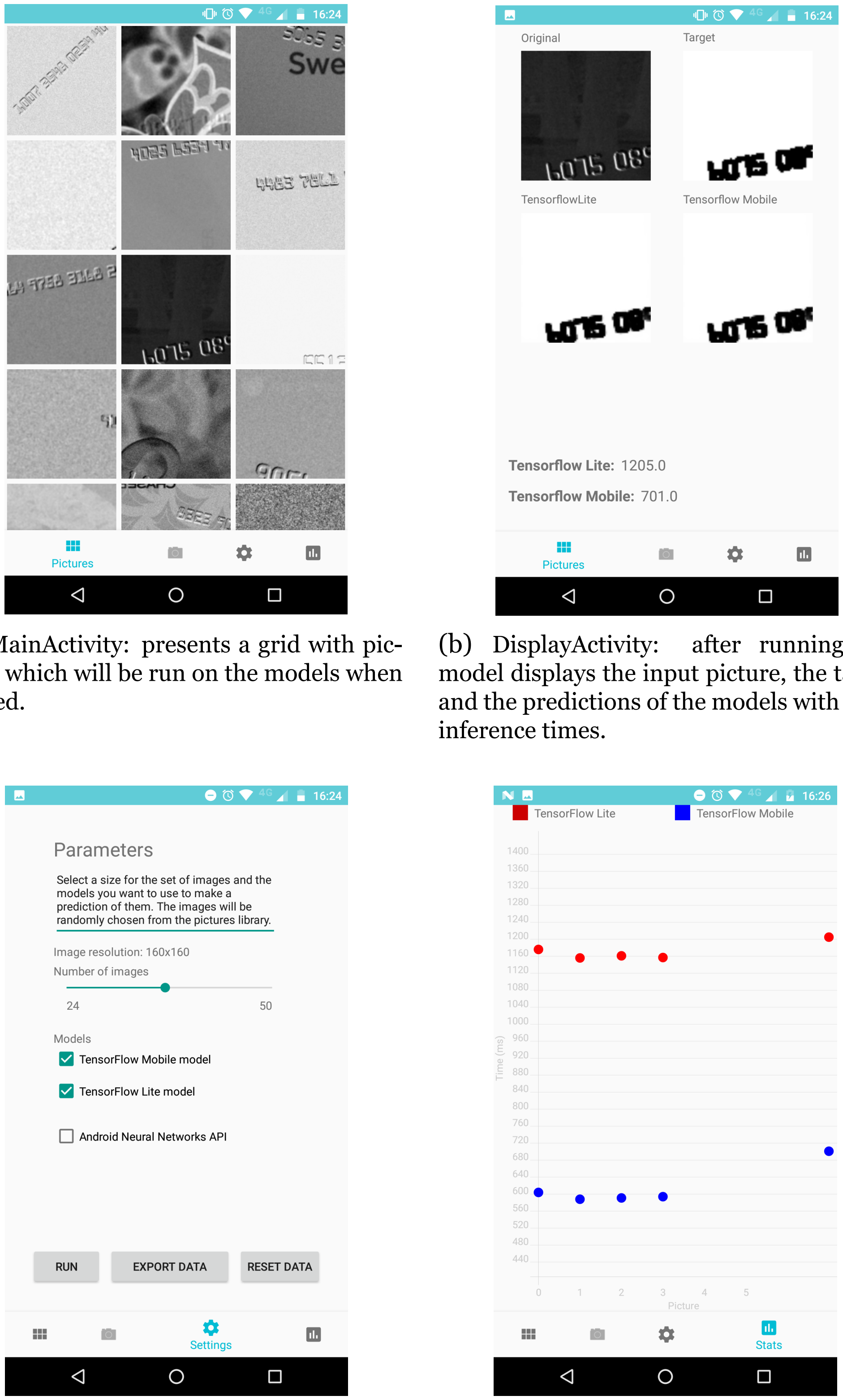
```java
inputData = convertBitmapToFloatArray(bitmap);
long startTime = SystemClock.uptimeMillis();
infInterface.feed(inputName, (float[]) inputData, 1, IMAGE_DIM_X, IMAGE_DIM_Y, 1);
infInterface.run(new String[]{outputName}, true);
infInterface.fetch(outputName, (float[]) outputData);
long endTime = SystemClock.uptimeMillis();
Bitmap outputImage = convertFloatArrayToBitmap(outputData);
```

Listing 2: code in *predictImage()* used for TfLiteModel

```java
inputData = convertBitmapToFloatArray(bitmap);
tflite.setUseNNAPI(neuralAPI);
long startTime = SystemClock.uptimeMillis();
tflite.run(inputData, outputData);
long endTime = SystemClock.uptimeMillis();
Bitmap outputImage = convertFloatArrayToBitmap(outputData);
```

The principal difference between both approaches is that the TensorFlow Lite execution of the model (listing 2) is atomical and happens within one function call while TensorFlow Mobile (listing 1) subdivides the whole operation in feeding the model with data, running it, and fetching the results. The input vector differs between both implementations, being a one dimensional float array (float[]) for TensorFlow Mobile and a four-dimensional float array (float [][][][]) for TensorFlow Lite. When the Android application was completed and fully operational several tests were run in a handful of devices. The results of these tests are described in the following section.

The process followed to implement the system has been thoroughly described,



(a) MainActivity: presents a grid with pictures which will be run on the models when clicked.

Tensorflow Lite: 1205.0

Tensorflow Mobile: 701.0

(b) DisplayActivity: after running the model displays the input picture, the target and the predictions of the models with their inference times.

Parameters

Select a size for the set of images and the models you want to use to make a prediction of them. The images will be randomly chosen from the pictures library.

Image resolution: 160x160

Number of images

24 50

Models

☑ TensorFlow Mobile model

☑ TensorFlow Lite model

☐ Android Neural Networks API

RUN EXPORT DATA RESET DATA

(c) SettingsActivity: Menu to run the models on multiple images (number selected with the slider). The collected data points can be written to a file (export button) or erased (reset button).

(d) StatsActivity: displays a chart which compares the inference time between the two models for each image.

Figure 8: Principal activities in test application

first explaining the export of the model in to the two different file formats and then illustrating the inner architecture of the Android application. The upcoming section presents the testing.

6 Test execution

When the application development phase was satisfactorily completed the project could carry on with the test phase. The application was exported from Android Studio (the IDE platform used to program it) as an apk file, the standard Android application package format, and installed on the selected devices. The tests were executed according to the design and the measured data was exported. Tables 1-4 present several statistical numerals about the inference times of the three different sets of images in each device, namely the average time of all images in a set $\overline{x}$, the standard deviation σ and the relative standard deviation *RSD*. Figures 9-12 display in a graph the values of the average inference times, to give the reader an intuitive impression of their relative magnitudes. The information contained in these tables and figures helps to arrive to some conclusions later on.

6.1 Test 1: Samsung Galaxy S8

The Samsung Galaxy S8 performed very well in the test. This does not surprise since it is Samsung's flagship device for the year 2017 and by far the most powerful of all the devices being tested. The relative standard deviation is very low for the different sets of images, so the measurements do not vary much. Table 1 and Figure 9 reflect the resulted values of processing the collected data with statistics, the data analysis tool selected in the methodology.

Samsung Galaxy S8	**TF Mobile**			**TensorFlow Lite**		
	$\overline{x}$ (ms)	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*	*$\overline{x}$ (ms)*	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*
160x160 px	*182,44*	5,8	3,2	371,52	24,3	6,5
240x240 px	*410,12*	12,7	3,1	814,16	30,3	3,7
320x320 px	*757,34*	19,4	2,6	1494,28	49,9	3,3

Table 1: Samsung Galaxy S8 test results

6.2 Test 2: Samsung Galaxy S5

The Samsung Galaxy S5 is also a powerful device, and it headed the top line of Samsung smartphones in its time, but now it is getting outdated and we can see that its performance is not as good as the S8. It runs an older version of the Android OS (4.4), even though a new one is available for this model. Still, the relative standard deviation is low as well. Table 2 and Figure 10 reflect the resulted values of processing the collected data with statistics, the data analysis tool selected in the methodology.

Samsung Galaxy S5	**TF Mobile**			**TensorFlow Lite**		
	$\overline{x}$ (ms)	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*	*$\overline{x}$ (ms)*	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*
160x160 px	462,9	25,7	5,6	1006,38	69,3	6,9
240x240 px	931,6	42,6	4,6	2212,34	79,7	3,6
320x320 px	2587,76	105,3	4,1	5777,78	215,8	3,7

Table 2: Samsung Galaxy S5 test results

6.3 Test 3: Samsung Galaxy S4

For today's standards, the Samsung Galaxy S4 is an average phone. We see that the performance of both models clearly drops with this device, especially with the bigger images. The relative standard deviation increases with the image size up to 17%. The slightly more modern version of the OS (5.0.2) does not help in improving the results. Table 3 and Figure 11 reflect the resulted values of processing the collected data with statistics, the data analysis tool selected in the methodology.

Samsung Galaxy S4	**TF Mobile**			**TensorFlow Lite**		
	$\overline{x}$ (ms)	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*	*$\overline{x}$ (ms)*	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*
160x160 px	516,9	43,8	8,5	1397,58	135,1	9,7
240x240 px	1307,1	160,9	12,3	3562,82	428,7	12,0
320x320 px	3360,74	571,8	17,0	9679,18	1663,7	17,2

Table 3: Samsung Galaxy S4 test results

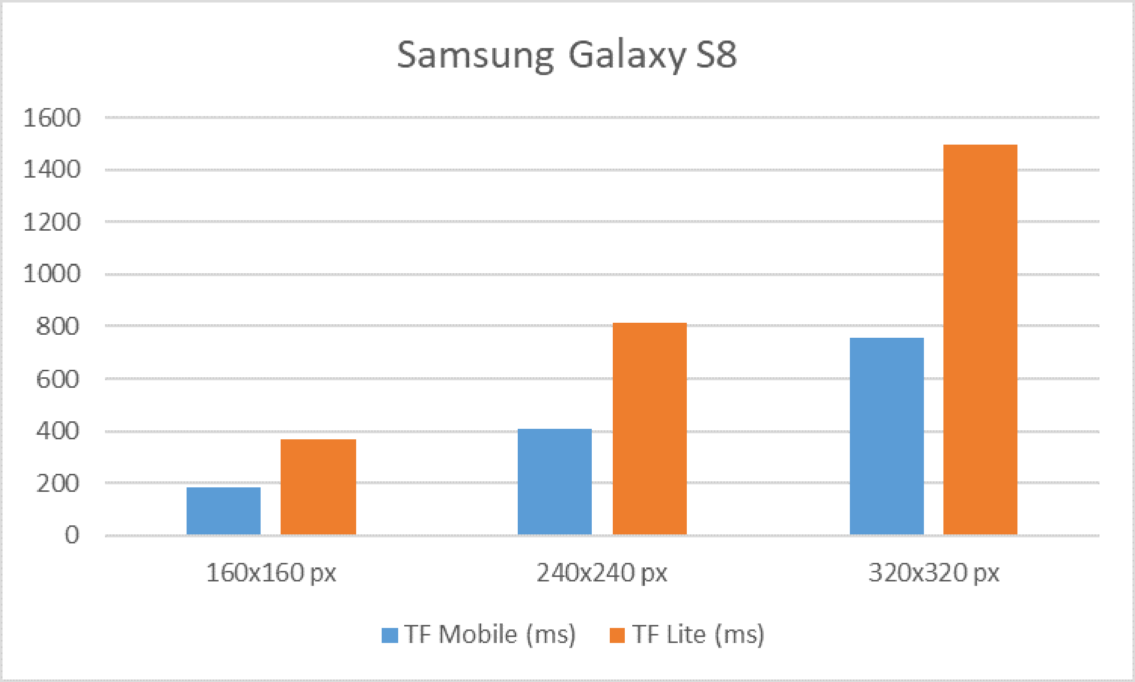


Figure 9: Samsung Galaxy S8 average inference times. This figure reflects the data from table 1

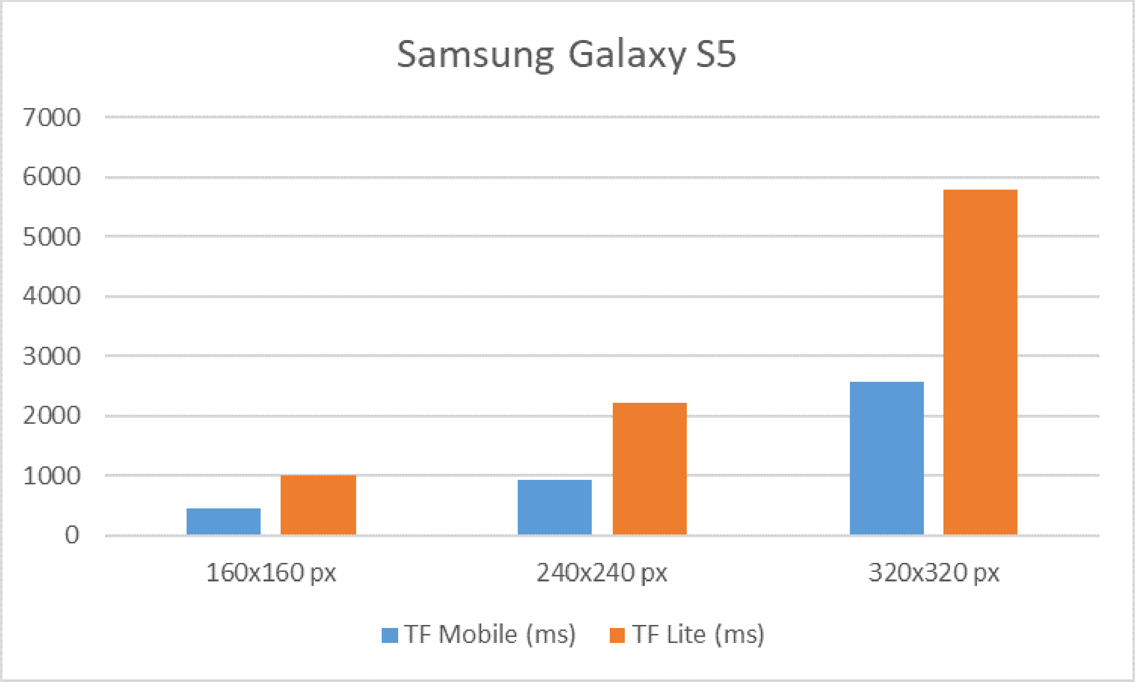


Figure 10: Samsung Galaxy S5 average inference times. This figure reflects the data from table 2

6.4 Test 4: Moto G 5 Plus

The Moto G 5 Plus belongs to a different family of devices, produced by Motorola. Since its initial release, the Moto G line has always offered middle-class devices with decent specifications at a low price. It is known of Motorola to generally supply its smartphones with the latest OS available, as well as sending regular updates. At the moment of the test, the device was running on Android 7.0, achieving acceptable results and even outperforming the Samsung Galaxy S4 and S5. Table 4 and Figure 12 reflect the resulted values of processing the collected data with statistics, the data analysis tool selected in the methodology.

Moto G5 Plus	**TF Mobile**			**TensorFlow Lite**		
	$\overline{x}$ (ms)	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*	*$\overline{x}$ (ms)*	*σ (ms)*	*RSD $\sigma/\overline{x}$ (%)*
160x160 px	577,66	8,6	1,5	1148,74	7,2	0,6
240x240 px	1276,94	7,4	0,6	2562,28	18,0	0,7
320x320 px	2259,64	8,7	0,4	4542,26	19,0	0,4

Table 4: Moto G5 Plus test results

In this section the raw data collected on the experimental tests performed in the four different devices has been thorougly presented. In the next section it will be crunched down and conclusions will be drawn from it.

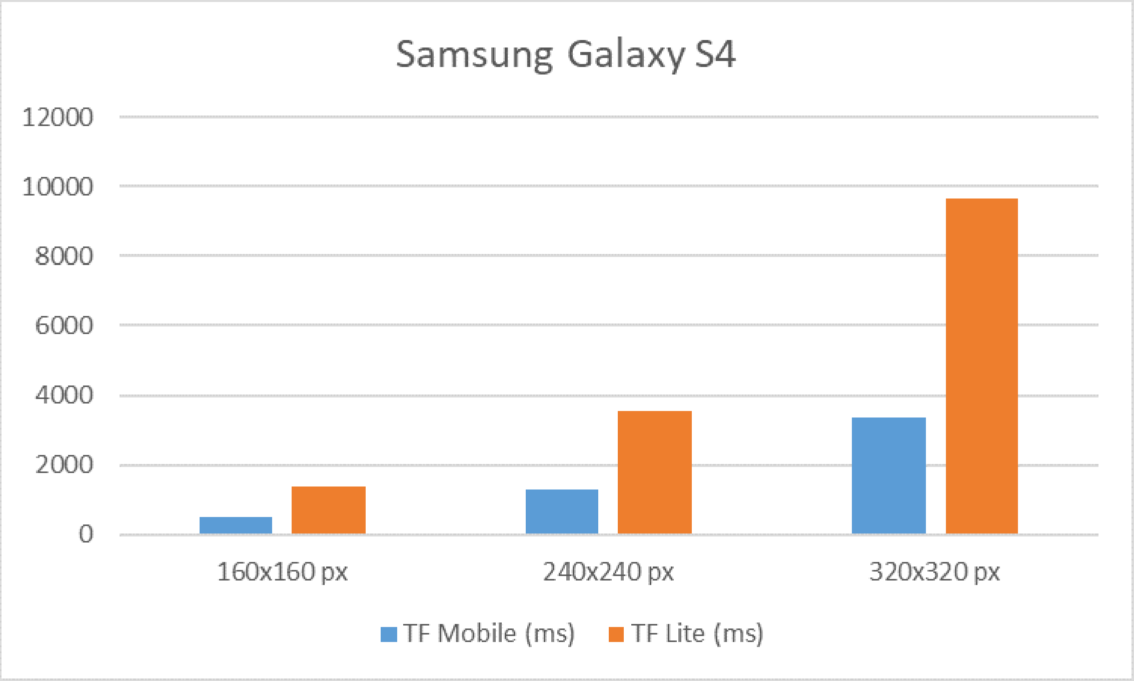


Figure 11: Samsung Galaxy S4 average inference times. This figure reflects the data from table 3

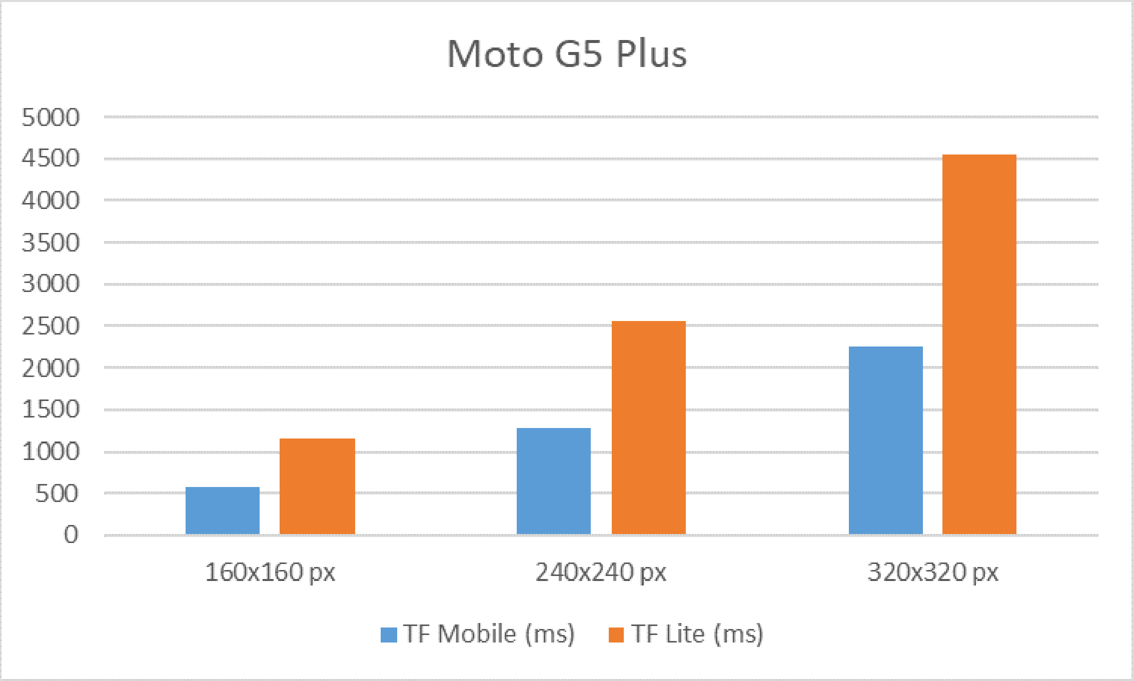


Figure 12: Moto G5 Plus . This figure reflects the data from table 4

7 Results

The following section presents the results of the thesis, comparing first the collected and analysed data between the different tests and then interpreting its implications.

7.1 Comparison of collected data across devices

As aforementioned, three sets of 50 images, with dimensions 160×160, 240×240, and 320×320 pixels were used to perform the tests. They were created with the credit card image generator that Slagkryssaren provided. The application was compiled and run with each set of images on four different Android devices, namely a Samsung Galaxy S8, a Samsung Galaxy S5, a Samsung Galaxy S4 and a Motorola Moto G5 Plus. The reason to test the models with several image sizes was to check how the inference time grows in relation to the volume of pixels that have to be processed. The processors on the selected devices to run the application represent appropriately the spectrum of processing power in nowaday's mobile phones. Some of them are high class devices whereas others are more outdated. They run as well different versions of the Android operating system.

Table 5 presents the average time taken for each model to run over the images of different sizes for each telephone. Surprisingly, the time taken for TensorFlow Lite to run is much longer than TensorFlow Lite. In Table 6 the proportion between both model's inference time is calculated. In the devices running more recent versions of Android, the proportion stays around 2.0, while in the ones with an older operating system it can go up to 2.9.

	160x160 px		**240x240 px**		**320x320 px**	
	TfMobile	TfLite	TfMobile	TfLite	TfMobile	TfLite
Moto G 5+	577,66	1148,74	1276,94	2562,28	2259,64	4542,26
Samsung S4	516,9	1397,58	1307,1	3562,82	3360,74	9679,18
Samsung S5	462,9	1006,38	931,6	2212,34	2587,76	5777,78
Samsung S8	182,44	371,52	410,12	814,16	757,34	1494,28

Table 5: Average inference time on each device for the three sets of images

	Android version	Ratio 160px	Ratio 240px	Ratio 320px
Moto G 5+	*7.0*	1,99	2,01	2,01
Samsung S4	*4.4.2*	2,70	2,73	2,88
Samsung S5	*5.0.1*	2,17	2,37	2,23
Samsung S8	*8.0*	2,04	1,99	1,97

Table 6: Ratio between TensorFlow Lite and TensorFlow Mobile average inference times.

7.2 Interpretation of results

The most outstanding result probably is that the TensorFlow Lite model is considerably slower than its counterpart on TensorFlow Mobile. Theoretically, TensorFlow Lite is claimed to be an improved version of its predecessor, built from the ground to be faster and more efficient. However, it has been empirically shown that this is not the case, at least for convolutional neural networks. As has been explained, the TensorFlow library is designed for a wide variety of machine learning graphs, and not only CNNs, so this lack of performance does not necessarily apply to other TensorFlow Lite operations.

As stated in the section 3.6.1 in the methodology to guarantee the validity of the collected data, that the inference time stamps were being correctly measured, the Android Profiler available in Android Studio was used. This tool monitors in real-time the activity of an application, and it helped to reassure that the *tflite.run(inputData, outputData)* function which can be seen in listing 2 was really being executed for an unexpected long time. After some research, the conclusion that the TensorFlow Lite library is in an early development stage was made. It was first released in November 2017 [4], and even though the engineers at TensorFlow are making great efforts to match the already existing TensorFlow Mobile the library is far from being completed and optimized. When running the app with various releases of TensorFlow Lite which have taken place during the first semester of 2018 substantial differences in its execution time have been found, so the efficiency problems present in the last TensorFlow Lite stable release will probably be solved in the near future.

Other interesting results come when contrasting the inference time between devices. Figure 13 and figure 14 plot the inference time of each device in relation to the amount of work (number of processed pixels) in TensorFlow Lite and Mobile respectively. Firstly, the Samsung Galaxy S8, the device with the most powerful processor, totally overthrows the rest, taking as little as 182 ms to run the TensorFlow Mobile model. A straightforward and obvious conclusion can be drawn: the more processing power a telephone has the faster it will run a TensorFlow model. But not everything depends exclusively on the hardware, the device's software has also an important impact to be considered. This can be seen by comparing the inference time taken by the Samsung Galaxy S5 and the Moto G5 Plus. The former has a better processor with a higher clock speed (2,5 GHz with 4 cores) but runs an older operating system (Android 4.4). With the latter exactly the opposite happens, its processor is not as fast (2,0 GHz with 8 cores) but has Android 7.0 installed. It can be seen in Figures 13 and 14 how with a smaller workload the Samsung S5 accomplishes lower inference time, but when the size of the image increases the Moto G5 Plus performs better since its software is more prepared.

In general, the devices with a more modern operating system suited for these kinds of tasks (Samsung S8 and Moto G5+) prove to have a linear evolution in terms of performance when the image size increases. In contrast, the older devices with a more outdated operating system show to have an exponential evolution of the inference time. They are unable to manage big workloads. A more modern OS allows the device to make measurements which are less disperse (Samsung S8 and Moto G5+ have considerably lower standard deviation than the rest of devices).

After obtaining and analyzing the results of this study in the form of empirical data some conclusions are ready to be drawn. First of all, as aforementioned, convolutional neural networks are not yet optimized for TensorFlow Lite, which makes them run slower than their equivalent in TensorFlow Mobile. This is due to the early development stage in which TensorFlow Lite is at the moment. The second conclusion is that both hardware and software affect the performance of the CNN execution. A more powerful processor will drop the inference time of the network, and a more recent version of Android will handle greater amounts of data with ease, making the execution time grow linearly with the network's workload. In contrast, the inference time increases exponentially with the workload in devices with an older operating system. The reliability of the measurements goes hand in hand with the version of the operating system. It has been show on the Test execution section that the devices running more modern versions of Android (Table 1 and Table 4) have a lower relative standard deviation than the devices with older operating systems (Table 2 and Table 3).

7.3 Project evaluation

The chosen methodologies and methods are from an overall perspective, suitable to the problem statement and the research done. A sufficiently large dataset is collected in the test running the experiment design. This design fulfils the demanded requirements of validity, reliability, replicability and ethics; so conclusions can be safely drawn from it. The project could have been performed from a different methodological approach, choosing other methods such as case study, for example. This would have presented several differences in the execution of the project, both in the form of advantages and disadvantages. However, the experimental analysis of the TensorFlow frameworks has been successful enough, and even a deliverable (the application itself) was produced, which easily lets other graphs be compared. The software development method Scrum proved to be apropriate and adequately adapted to the situation of one single developer.

7.3.1 Feedback from Slagkryssaren

Slagkryssaren was satisfied with the work done, and also surprised with the obtained results. They consider that this is a topic worth to keep investigating in. They agree that it might be a matter of interest to extend the testing to other devices with different versions of Android in order to get a more accurate idea of the influence that the characteristics of each device has on the results of the test.

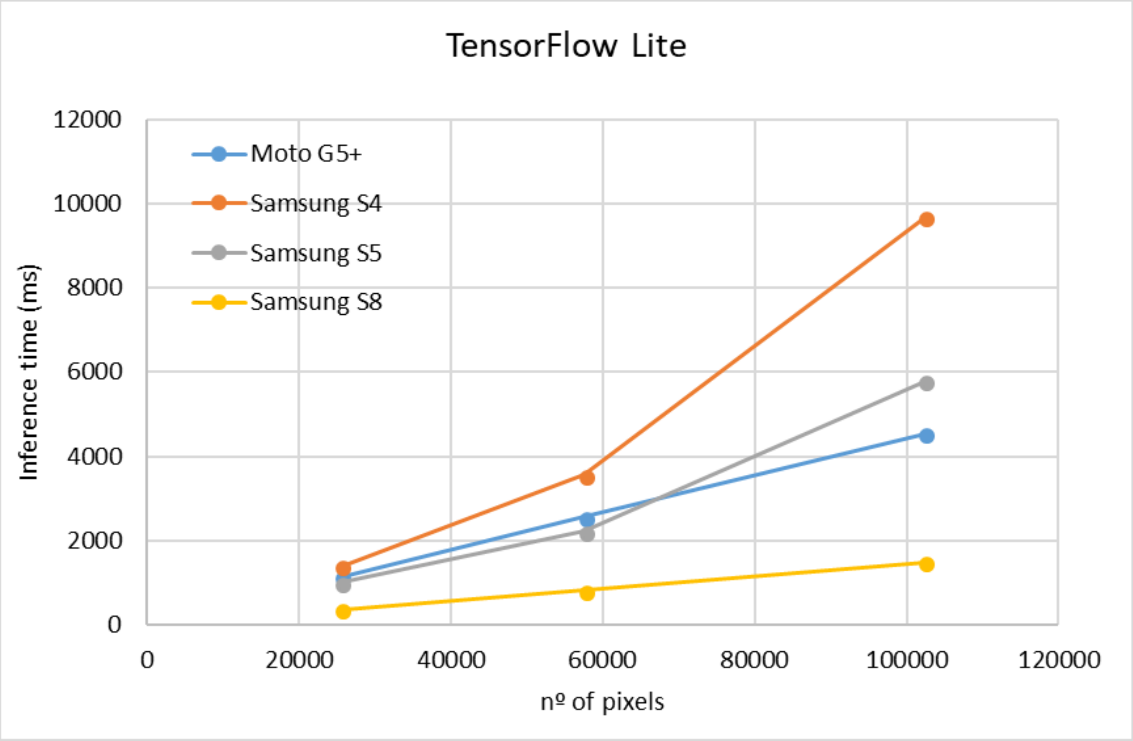


Figure 13: Performance comparison of TF Lite across devices. The X axis represents the workload (number of pixels). The inference time varies from 371 to 9679 ms.

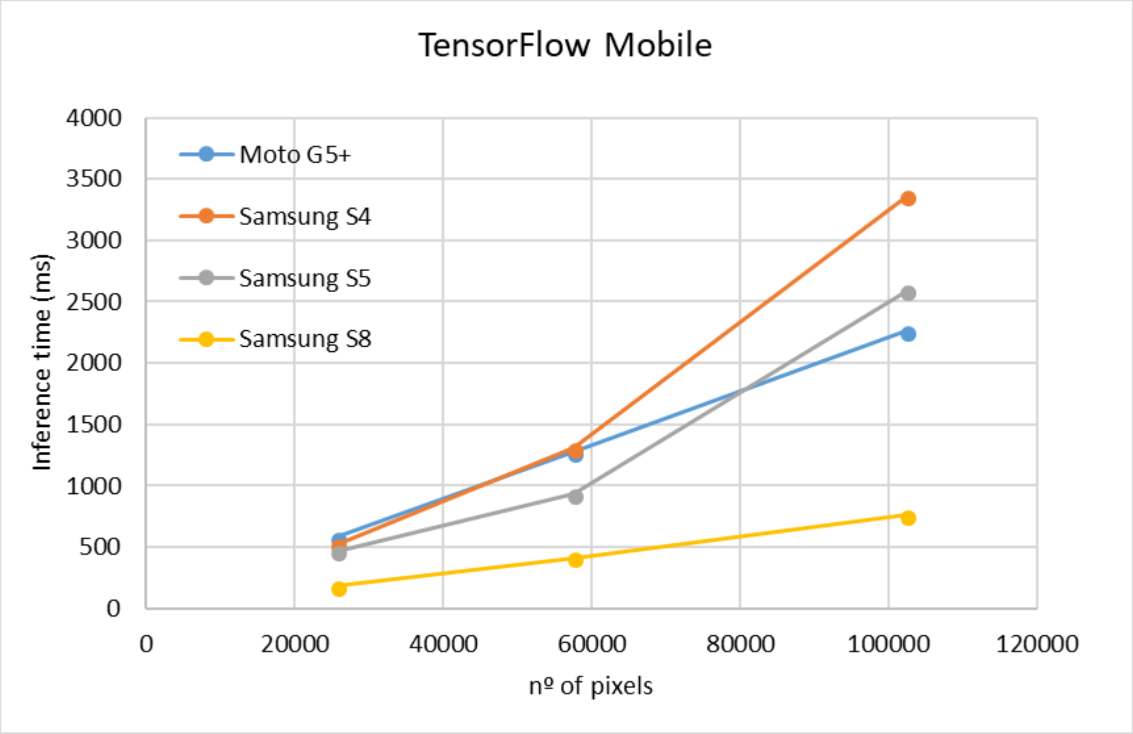


Figure 14: Performance comparison of TF Mobile across devices. The X axis represents the workload (number of pixels). The inference time varies from 182 to 3360 ms.

8 Conclusions and future work

This thesis presents an analysis of the variation of the execution speed of a convolutional neural network depending on if it is run as a TensorFlow Lite model or as a TensorFlow Mobile model. This was done in collaboration with Slagkryssaren, a software development company based in Stockholm. This company stated a series of requirements for the project and provided with guidance and assistance in its completion. The main focus of the project was to export an specific convolutional network implemented in TensorFlow to the Mobile and Lite versions of the library, for later embedding the resulting models in an Android application and testing it in different devices. The purpose of this thesis, to present the comparison between this two frameworks and the related research done, has been accomplished succesfully in this report.

The goal of the project, providing with a solution in the form of an Android application which allows the comparison of the execution speed of a CNN implemented in both TensorFlow Lite and TensorFlow Mobile has been achieved, as well as its subgoals, namely getting acquainted with the TensorFlow framework, exporting the CNN graph in the Lite and Mobile formats, developing the test application in Android and running the tests to compare the two models. The problem statement, which questions how much does TensorFlow Lite improve over its predecessor TensorFlow Mobile, is now answered: there is not yet an improvement in the performance of CNNs but a decay. Through a literature study related work has been conveniently analyzed and the knowledge adquired from it utilized in the thesis. From all the study, the three related works with highest impact in the project are described: a comparative study between TensorFlow and Microsoft CNTK which gave insights about other existing machine learning frameworks; a descriptive report of TensorFlow's performance in large scale systems, which made an impact in the system's architecture design; and the Alexnet paper which helped understanding the design of the CNN.

8.1 Discussion

The project requirements from the stakeholder together with the system architecture designed were described in detail. These helped to design the system architecture and to put it into realization. Methodological theory was presented at its different levels, with several research methods described. From these, some were chosen giving a motivation and subsequently applied to the research. A deductive approach was selected due to the nature of the problem statement, the experiment was used as data collection method, and statistics as the data analysis method. Also, the interview is punctually used to collect information about the project's requirements from the stakeholders. Regarding software development methods, several approaches were studied to perform this project. To plan the steps of its execution, the agile development method Scrum was utilized. The Android platform was chosen to develop the test application, and Git was the version control system used, together with the Gitflow workflow.

This study has several delimitations. The most influential is possibly restricting the performance comparison exclusively between TensorFlow Lite and TensorFlow Mobile, without taking into account other machine learning libraries. Another big delimitation analysing the performance between the two frameworks only in the context of convolutional neural networks, not taking into consideration other kinds of networks. Of course, all conclusions drawn apply only to CNNs. Other delimitations are the small number of devices in which the tests are performed as well as their respective operative systems. With a bigger supply of devices measurements which represent better the general performance of the TensorFlow frameworks could have been made. As well due to time constraints there were few (only 3) image sizes to test the models on each device. A set of images with more continuously distributed sizes would have shown how the ammount of work affects the performance of the models in a more reliabe fashion.

8.2 Future work

Subsequent future work could be done on this matter, by waiting until a stable and efficient version of TensorFlow Lite is released and making the tests again to confirm that it has performance improvements. As well, other machine learning libraries could be considered in future work, and analyse their performance in relation to the TensorFlow frameworks. During this research, an attempt was made of performing the tests in a Pixel 2 phone, but the tests failed due to technical problems. Tests on this device were of special interest for its capability of using TensorFlow Lite with NNAPI (Android neural networks API) which is designed to run machine learning models. This chip would enormously boost the performance of the CNN running in TensorFlow Lite, and it would be fascinating to test how far can the NNAPI go. Some other possible improvements are enlarging the set of smartphones used for the tests. The selection of devices attempted to cover the whole quality range of Android devices in the market. Nevertheless, it is obvious that this number falls too short to give an accurate insight into the whole spectrum of phones. Another improvement that could have been interesting to do but was not feasible due to time constraints is to check the performance of the TensorFlow models in each device but running on different versions of the operating system. Such a study would have given a clearer idea of how the OS affects the results. This would have meant to manually download and install several versions of the OS for each device, and run the tests on them but it would have taken too much time and it was out of the scope of this project. This is something that could be studied in the future.

References

[1] S. Harnad, "The annotation game: On turing (1950) on computing, machinery, and intelligence (published version bowdlerized)," 2008.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, 2016, pp. 265–283.

[3] T. team. (2018). Tensorflow lite versus tensorflow mobile, [Online]. Available: `https://www.tensorflow.org/mobile/` (visited on 05/25/2018).

[4] L. Matney, "Google shares developer preview of tensorflow lite," *Google Open Source Blog, Available at least as early as Jul*, [Online]. Available: `https://techcrunch.com/2017/11/14/google-releases-developer-preview-of-tensorflow-lite/?guccounter=1` (visited on 11/15/2017).

[5] A. Håkansson, "Portal of research methods and methodologies for research projects and degree projects," in *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2013, p. 1.

[6] S Rajasekar, P Philominathan, and V Chinnathambi, "Research methodology," *arXiv preprint physics/0601009*, 2006.

[7] T. D. Jick, "Mixing qualitative and quantitative methods: Triangulation in action," *Administrative science quarterly*, vol. 24, no. 4, pp. 602–611, 1979.

[8] (2018). Slagkryssaren, [Online]. Available: `https://slagkryssaren.com/` (visited on 05/30/2018).

[9] O. Omidvar and D. L. Elliott, *Neural systems for control*. Elsevier, 1997.

[10] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[11] A. Karpathy. (2018). CS231n Convolutional Neural Networks for Visual Recognition course notes, [Online]. Available: http://cs231n.github.io/convolutional-networks/ (visited on 04/05/2018).

[12] R. Airola and K. Hager, *Image classification, deep learning and convolutional neural networks: A comparative study of machine learning frameworks*, 2017.

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 609–616.

[14] T. team. (2018). About tensorflow, [Online]. Available: https://www.tensorflow.org (visited on 05/25/2018).

[15] ——, (2018). Introduction to tensorflow mobile, [Online]. Available: https://www.tensorflow.org/mobile/mobile_intro/ (visited on 05/25/2018).

[16] K. Varda, "Protocol buffers: Google's data interchange format," *Google Open Source Blog, Available at least as early as Jul*, vol. 72, 2008.

[17] T. team. (2018). Introduction to tensorflow lite, [Online]. Available: https://www.tensorflow.org/mobile/tflite/ (visited on 05/25/2018).

[18] X. Zhang, Y. Wang, and W. Shi, "Pcamp: Performance comparison of machine learning packages on the edges," 2018.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[20] R. B. Burns and R. B. Bursn, "Introduction to research methods," 2000.

[21] J. Dewey, "Logical method and law," *Cornell LQ*, vol. 10, p. 17, 1914.

[22] N. J. Salkind and T. Rainwater, *Exploring research*. Prentice Hall Upper Saddle River, NJ, 2003.

[23] M. N. Saunders, *Research methods for business students, 5/e*. Pearson Education India, 2011.

[24] M. D. Myers, *Qualitative research in business and management*. Sage, 2013.

[25] M. McCormick, "Waterfall vs. agile methodology," *MPCS, N/A*, 2012.

[26] K. Beck and E. Gamma, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[27] K. Schwaber and M. Beedle, *Agile software development with Scrum*. Prentice Hall Upper Saddle River, 2002, vol. 1.

[28] S. Alliance, "Learn about scrum," *Scrum Alliance*, 2016.

[29] L. Rising and N. S. Janoff, "The scrum software development process for small teams," *IEEE software*, vol. 17, no. 4, pp. 26–32, 2000.

[30] A. Developers, *What is android*, 2011.

[31] A. Studio, "The official ide for android," *Android Studio. URL: https://developer. android. com/studio/index. html*, 2016.

[32] L. Torvalds and J. Hamano, "Git: Fast version control system," *URL http://git-scm. com*, 2010.

[33] A. Dwaraki, S. Seetharaman, S. Natarajan, and T. Wolf, "Gitflow: Flow revision management for software-defined networks," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ACM, 2015, p. 6.

[34] T. team. (2018). Preparing models for mobile deployment, [Online]. Available: `https://www.tensorflow.org/mobile/prepare_models` (visited on 05/25/2018).

TRITA TRITA-EECS-EX-2018:443