Adapting Deep Neural Networks to a Low-Power Environment

Oscar Mañas Sanchez

Director: Antonio González
Co-Director: Jose Maria Arnau
Specialization: Computer Science

Universitat Politècnica de Catalunya
2017

Contents

List of Figures

List of Tables

1 Context and scope of the project

1.1 Context and problem formulation

The holy grail of computing is artificial intelligence: building a machine so intelligent, it can learn on its own without explicit instruction. Machine-Learning algorithms have become widespread in a very broad range of applications and cloud services. These applications are displacing scientific computing as the major driver for high-performance computing. While this profound shift in applications is occurring, the machine-learning domain has greatly evolved since 2006, where a category of algorithms, called deep learning (convolutional and deep neural networks), has emerged as state-of-the-art across a broad range of applications [1], [2], [3], [4], [5].

Deep learning is a critical ingredient for achieving modern AI, and it is revolutionizing the world. In the last years, deep neural networks (DNN) have been conquering, one by one, various algorithm domains related to computer vision in particular and machine perception in general. There is an endless number of potential use cases: from self-driving cars [6] to pharmaceutical research [7] and faster drug development [8], from automatic image captioning [9] in online image databases to smart real-time language translation [10] in video chat applications. Even in the field of art, there have been developed applications for transferring the pictorial style of a painting onto an image [11].

Deep learning allows the AI *brain* to perceive the world around it; the machine learns and ultimately makes decisions by itself. It takes massive amounts of data to train the machine to do this. In addition, highly sophisticated deep neural networks are needed to process it all. In 2012, Google's Deep Learning project, Google Brain, learned to recognize cats by watching movies on YouTube [4]. But it required 2,000 CPUs (16,000 CPU cores) in servers powered and cooled in one of Google's data centers to do this. Few organizations have machines of this scale. Around the same time, NVIDIA Research teamed with Stanford University to use GPUs for deep learning. It turned out that 12 GPUs could deliver the deep learning performance of 2,000 CPUs.

That is the reason why these days, working with deep neural networks goes hand in hand with the use of GPUs. On a high level, working with deep neural networks is a two-stage process: First, a neural network is *trained*, i.e. its parameters are determined using labeled examples of inputs and desired output. Once a deep neural network has been trained for hours, days, or weeks, the next step is to put it to use by deploying it in the field, where it will repeatedly run *inference* computations. Examples of inference (giving an input to a DNN which then extracts information based on that input) include classifying images, localizing faces, or translating human speech in real-time.

Training and inference have different performance goals. Training strives for throughput (because it is common to batch hundreds of training inputs), whereas inference strives for latency (because users do not want to wait several seconds for their input to be processed). Furthermore, the computational requirements for running inference are a lot less than for training a deep neural network. This, in conjunction with the mobile nature of some of the before mentioned applications, make the next logical step pretty clear: be able to run inference live, in real time, on people's devices (i.e. run inference in a low-power environment using a mobile GPU), while the training step is done beforehand using dedicated hardware (i.e. high end GPUs).

There already exist mobile applications that are capable of perform this kind of tasks [12], but they rely on the cloud to run the most computationally expensive parts. This is not ideal for several reasons. First, it does not guarantee the user's privacy because the data is being sent to remote servers. Second, the algorithm performance may be damaged due to latency issues on a poor network connection. In addition, sending continuous data to a server may be too expensive. In this context, this project aims to make mobile devices perceive, process and comprehend in real time, without having to connect to remote servers.

Nevertheless, running large neural networks requires a lot of memory bandwidth to fetch the weights and a high computational power to perform a lot of floating point operations, and mobile devices have limitations. While they have improved significantly in computation power in the last years and are capable of carrying out billions of arithmetic computations per second, they also have various resource constraints like power, memory, and compute capability. This makes power and memory hungry applications such as deep neural networks hard to deploy, requiring smart software design. As a result, mobile presents both an opportunity and challenge for machine learning systems.

The objective of this project is to be able to run in real time a specific application like FASTER R-CNN [13] that does inference using a particular deep learning framework like CAFFE [14]. The application takes an image as input and outputs a list of regions with predictions of objects inside those regions. It uses a DNN that combines a Region Proposal Network (RPN) and a Fast Region-based Convolutional Network (Fast R-CNN); the RPN component tells the unified network where to look, and then the Fast R-CNN component is used to perform object detection inside the regions of interest. The application will be run in a concrete low-power environment like the NVIDIA JETSON TX1 [15], which is a high-end mobile platform with a low-power GPU (NVIDIA Tegra X1) used in commercial products. To accomplish this, several software optimizations will be applied.

1.2 Stakeholders

1.2.1 Developer

Is the person in charge of research, document and implement all the required software. In addition, he/she is responsible for the project management and the writing of the report and all the required documentation. This actor works as agreed with the director and co-director and he is, ultimately, the person in charge of accomplish the deadlines.

1.2.2 Director and co-director

They are the main responsibles for guiding, giving advice and, in general, helping the developer. Their action is key to determine possible errors in the project, both in its proposal and execution. In particular, Antonio González from the Computer Architecture Department (DAC) and leader of the ARCO research group, has acted as director. Jose Maria Arnau, also from the DAC, has acted as co-director.

1.2.3 Beneficiaries

Since this project is not about creating a product, it may seem that it does not have any direct beneficiary. Nevertheless, optimize and adapt an application that uses deep neural networks to a low-power environment may bring knowledge to researches interested in similar topics. In addition, smartphone users can benefit from these optimizations and have a better user experience when using applications that implement them. Furthermore, this project can also serve as a starting point for future projects that also use the Jetson TX1 platform.

1.3 State-of-the-art

This is a very hot topic, so there is a lot of research on this subject going on these days. Some researchers are trying to minimize the memory footprint using lower precision and approximate computing, and reducing the cost of convolutions with FTT [16]. Others are trying to reduce the size of deep neural networks by factoring the learned parameters to keep them on a constrained rank [17].

Another approach is CNN compression. In [18], a three stage pipeline is aimed to reduce the storage requirement of neural networks. The network is first pruned by learning only the important connections. Next, weights are quantized to to enforce weight sharing. Finally, Huffman coding is applied to the weights. In [19], the process also consists in three steps. The first step is rank selection with variational Bayesian matrix factorization. Second, Tucker decomposition is applied on kernel tensors. Finally, the network is fine-tuned to recover the accumulated loss of accuracy.

Other applications allow to further optimize the model. One example is the spotting of audio keywords like “Ok, Google”, which uses a finished trained model on the device [20].

Another improvement is the final fine-tuning of the model and adaption of some parameters on the target device [21]. For example, the model is pre-trained to recognize faces. Then, on the mobile device, the network is trained to recognize one specific face [22].

One example of a real application running deep learning algorithms inside a phone is Google Translate [23]. To achieve real time, the Google team heavily optimized and hand-tuned the math operations, using the mobile processor's SIMD instructions and tuning things like matrix multiplies to fit processing into all levels of cache memory.

Maybe the most similar approach to what is being proposed in this project is the work of folks at Facebook with Caffe2Go [24]. They designed a lightweight and modular framework by building on top of the open-source Caffe2 project [25]. Because of the modular design, the framework can speak the same language but be optimized for each platform. When the graph is actually run, it instantiates itself with the various hardware features to achieve maximum speed.

1.4 Scope

The first task will be getting up and running the application Faster R-CNN in the Jetson TX1 platform. Next, a profiling of the application will be performed in order to identify the main bottlenecks and determine the parts where the optimizations are most needed.

Since the application is built with Python on top of the Caffe deep learning framework, which is written in C++, the improvements will be focused on Caffe. The optimizations will be centered in reducing the memory bandwidth and the energy required to run inference in the Jetson TX1 platform, which in turn will contribute to reduce the inference time. In that sense, the long term goal is to to achieve real-time processing speed. That means about 20-30 frames per second, i.e. about 30-50 ms to process a single image. But this is a pretty ambitious goal and is unlikely to be reached, since the current time per frame in the Jetson TX1 platform is over 2 seconds. The actual goal will be to get as close as possible to a real-time value.

The list of optimizations will include exclusively software optimizations, like using half-precision floating point data types, reusing the same memory buffer between layers, using data compression algorithms to minimize the size of the weights, pruning connections with weights close to zero and applying a retraining step afterwards, and using cuSPARSE [26] for handling the sparse matrices resulting from the pruning. The use of half-precision floats is especially attractive, not only because it contributes to reduce the memory footprint, but also because mobile platforms offer FP16 units not available in desktop, which double the throughput with respect to FP32 units. As a disclaimer, since the project is still at a pretty early stage, these are only some ideas of the optimizations that may be performed. As the project develops, some of these may be discarded and some others may be added.

After each optimization, the code will be tested in terms of accuracy and efficiency. Regarding the accuracy, a series of unit tests will be run to ensure that the application is still giving a correct result. To test efficiency, the code will be instrumented with several timers in order to measure how much time is spent in each of the stages of the algorithm.

1.5 Methodology

As a result of the tight timetable the project will be developed in, an agile approach seems the best fit. Agile methodologies provide flexibility, fast development and results in less time than any other ordinary methodologies. However, currently accepted agile methodologies are very team-oriented, so strictly following any of them would make little sense. Nonetheless, some of these methodologies concepts are still valid for solo projects.

1.5.1 Short development cycle

By using an iterative approach with short cycles (goals will be specified on a weekly basis), it is much easier to keep the project on schedule and be conscious about the current state of the project. On a higher level, iterations will be done on a feature basis. That is, each iteration will focus on an optimization and will consist of the following stages: (1) profile of the application to detect bottlenecks, (2) think about the best way to optimize the code, (3) implement the optimization and (4) evaluate if it has been enough improvement. If the answer is no, the next iteration will focus again on the same optimization.

1.5.2 Intensive client feedback

Although the project does not have a real client, this concept from agile methodologies still applies: by letting the person that is ultimately responsible for evaluating the project check the project state as frequent as possible, chances for misunderstandings are greatly reduced and, if they occur, are corrected much faster.

1.6 Monitoring tools

Git [27] and GitHub [28] are going to be the tools used to monitor the evolution of the project. Git enforces a short cycle approach (by means of commits) and forces the developer to document all changes. GitHub provides a remote Git repository and an integrated issue tracking system, perfect for setting milestones and tracking the project's progress.

1.7 Validation methods

The combination of unit tests, GitHub issues and weekly meetings with the project tutor ensure the validation of the objectives on their own without need for further measures.

1.8 Possible obstacles and solutions

1.8.1 Optimizations selection

A bad selection of the optimizations to be performed could result in a waste of time without a significant gain. That is why the selection of the optimizations will be guided by Amdahl's law: the first optimizations to be implemented will be those focusing on parts that have a greater margin of improvement. This way, the time spent in each optimization will be proportional to the theoretical gain that could be obtained if it is correctly implemented.

1.8.2 Scheduling

Four months can seem like a long time, but a bad scheduling can make it insufficient. To avoid this problem, a very rigid but realistic timetable with weekly meetings with the project tutor will be set to ensure the project is on schedule, or to apply the appropriate corrections otherwise.

1.8.3 Bugs

Considering that a deep learning framework is a relatively complex software, it is easy to introduce bugs while modifying the source code. To ensure no bugs are present, a series of unit tests will be run after each optimization to verify that the application is still giving a correct result.

1.8.4 Computational power

Some of the optimizations may require a retraining step. While running inference is pretty cheap in terms of computational power, training needs a lot more power since hundreds of millions of inputs are passed forward and backward through the network. Shall this be the case, using the ARCO research group hardware would probably be enough.

2 Project planning

2.1 Planning and scheduling

The estimated project duration is of about 4 months. The project starts on Wednesday 1st February, 2017 and the deadline is on Sunday 25th June, 2017, the day before presentations start.

As a disclaimer, it must be pointed out that the initial planning could be revised and updated as a result of the evolution of the project. Furthermore, the use of agile methodologies implies that new requirements that alter the original planning may appear.

2.2 Task description

2.2.1 Acquire background in deep neural networks

In the last months I have been learning about deep learning in general and deep neural networks in particular. I started by reading an online book called *Neural Networks and Deep Learning* [29] that gave me a general view on how artificial neural networks work, from simple perceptrons to more complex convolutional neural networks, and how they are trained. From there, I took a course on Coursera titled *Neural Networks for Machine Learning* [30] imparted by Geoffrey Hinton (University of Toronto). Once I got familiar with all the concepts related to neural networks, I read three articles involved in the particular application that is going to be examined in this project [31], [32], [33]. This process took several months because I was also studying my seventh full time semester at FIB. This task did not require any material resources (except for the printed documents), but it did require human resources to read and understand all the information.

2.2.2 Get familiar with Caffe

A deep learning framework is a pretty complex piece of software, and the kind of optimizations we will be doing require a deep understanding of the intricacies of the source code of Caffe: how data is organized, how memory is managed, etc. That is why, before starting to optimize this framework, a deep dive into the source code of Caffe is required. This task will take several weeks, but it can be done in parallel with the GEP course. As a material resource, the Caffe deep learning software is required. This task also requires human resources to read and understand the source code.

2.2.3 Jetson TX1 set up

The whole project will be developed in a specific high-end mobile platform: the NVIDIA Jetson TX1, which packs a Tegra X1 GPU and an ARM Cortex-A57 processor. Despite being a mobile platform, it can run a Desktop OS like Ubuntu. The first step is installing Ubuntu 14.04 in the system and, after that, installing the Caffe dependencies. Luckily enough, NVIDIA also ships an all-in-one package called JetPack which bundles and installs all software tools required to develop for the platform, such as cuDNN and CUDA. The last step is forking the Caffe repository from GitHub, build it and run some tests in order to make sure that everything works properly. Once the Jetson TX1 developer kit arrives, this task should not take longer than a day. As a material resource, this task will require the Jetson TX1 developer kit and all the software necessary to run Caffe. This task also needs human resources to set up the Jetson TX1 module and install the required software.

2.2.4 Initial neural net characterization

Before applying any optimization, first we have to find out where the bottlenecks are, i.e. which parts of the code up the bulk and execution time. This is called profiling or performance characterization: analyze the program to obtain detailed information about execution time, memory usage, energy consumption, etc. A Python script will be written in order to obtain information about execution time and memory. As for energy consumption, the Jetson TX1 developer kit has hardware counters that allow to extract how much energy is consuming a particular running application. The idea is to extensively profile the application at first, and then a less detailed profiling will be run after each optimization. This task should take several days, and it requires the Jetson TX1 developer kit and all the involved software as a material resource. This task also needs human resources to write the scripts that will be used to collect the profiling data and then analyze that data to detect bottlenecks.

2.2.5 Use half-precision floating points

This optimization consists in substituting all the single-precision floats used in the application for half-precision floats. The motivation behind this optimization is that regular, single-precision floats take 32 bits of memory, but half-precision floats take only 16 bits. This not only will cut the memory footprint by half, but will allow us to benefit from FP16 units only present in mobile GPUs like the Tegra X1: NVIDIA claims that FP16 units can deliver two times more throughput than regular FP32 units, so the execution time will also be reduced. Furthermore, this optimization is extensively supported by the literature: some studies have shown that the use of half-precision floats incurs little to no degradation in the classification accuracy [34]. This task should take several weeks, depending on the easiness of adapting the whole application to half-precision floats. As material resources, it requires the Jetson TX1 developer kit and all the involved software. This task also needs human resources to adapt the application to FP16.

2.2.6 Pruning of the neural net and retraining

The next optimization consists in pruning the DNN to reduce memory footprint and computation time. Once the neural net has been trained, each connection has a weight assigned. These weights are then used in the forward pass to perform the operations that ultimately allow to classify an image. But if a weight is close to 0, it has little to no impact in the final decision of the net. That is why connections with weights close to zero can be pruned away with little impact on accuracy. More aggressive pruning can be employed to further reduce the size of the DNN. However, very aggressive pruning (around 90% of the net) has a significant impact on accuracy. In that case, weights which have not been pruned need to be adjusted. Hence, a retraining step is needed after pruning. Since the retraining of the net alone could take up to a week, this task may take several weeks. As a material resource, this task requires the Jetson TX1 developer kit and all involved software. This task also needs human resources to write the scripts needed to do the pruning and to perform the retraining of the net.

2.2.7 Use cuSPARSE

The last optimization is a logical step after performing the pruning. In Caffe, weights are represented as multi-dimensional matrices, so it is not surprising that a great portion of the computation consists in operations with matrices. The previous optimization has the side effect of converting most of these matrices in sparse matrices since pruned weights are set to zero, and operations with sparse matrices can be exploited to improve performance over dense lineal algebra. By default, Caffe uses cuBLAS to perform matrix operations. But if these matrices are sparse, there is a better solution: cuSPARSE. Therefore, the optimization consists in substituting every call to a cuBLAS routine for a call to a cuSPARSE routine. This task should not take more than a few days, and it requires the Jetson TX1 developer kit and all the involved software (particularly cuSPARSE) as a material resource. This task also needs human resources to adapt the application to cuSPARSE.

2.2.8 Final neural net characterization

The final task consists in reporting the gains obtained with the aforementioned optimizations. To this extent, the application will be characterized again measuring execution time, memory usage, energy consumption, etc. This task should take several days, and it requires the Jetson TX1 developer kit and all the involved software as a material resource. This task also needs human resources to run the scripts that will be used to collect the profiling data and then analyze that data to detect improvements with respect the initial version of the application.

2.3 Estimated time

In table 1 an estimation of the number of hours dedicated to each task is shown.

Task	Estimated duration (h)
Acquire background in DNN	70
Get familiar with Caffe	40
Jetson TX1 set up	10
Initial net characterization	30
Use of half-precision floats	90
Pruning and retraining	90
Use of cuSPARSE	60
Final net characterization	30
Final stage	30
Total	450

Table 1: Estimated time for each task

2.4 Gantt chart

Figure 1 shows the planning of the different tasks of the project in a Gantt chart.

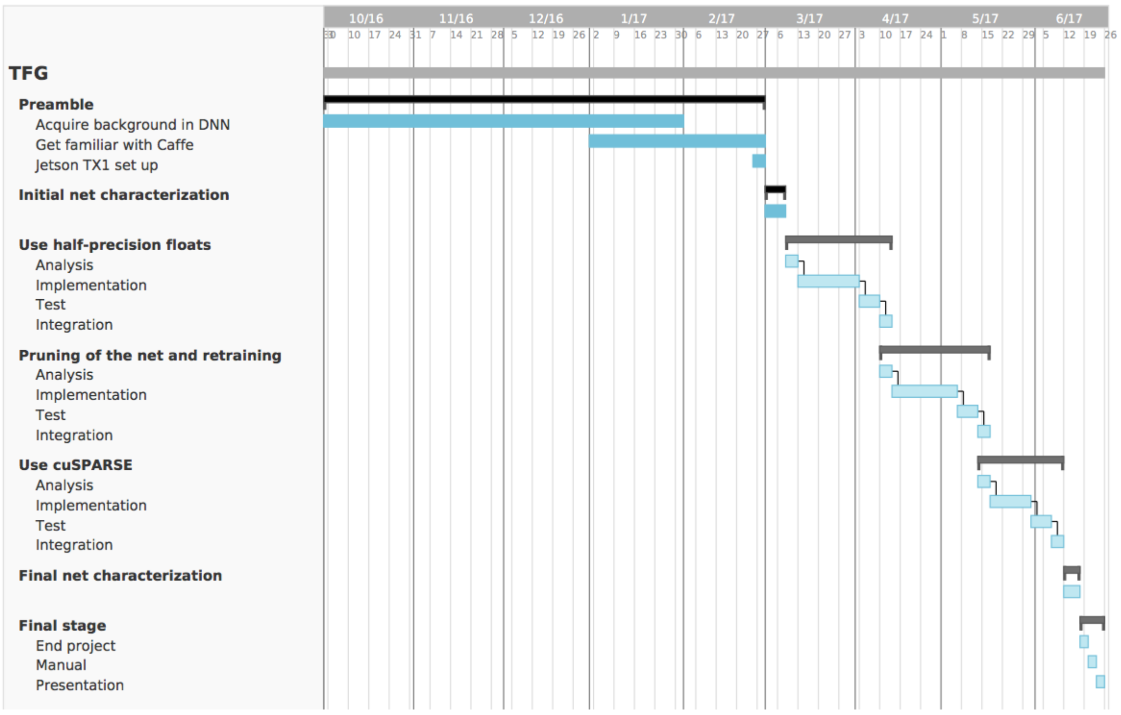

Figure 1: Gantt chart of the project. Generated with `https://www.teamgantt.com`.

2.5 Alternatives and action plan

As it was pointed out in the previous delivery, the agile methodology will allow to revise and adapt dynamically this initial planning. Therefore, if the real duration of the specified tasks differs from the expected duration, the planning will be modified accordingly. In the (rare) case that a task has less duration that expected, the next task will immediately start. On the other hand, if a task lasts more than expected, a later task will have to be shortened or, in the worst case, completely omitted. The mentioned optimizations are decreasingly ordered based on the estimated gain in terms of execution time, memory usage and energy consumption. So if the project was running short on time, the last optimization could be cut out.

Since weekly meetings have been arranged with the project tutor, there will be enough time to detect possible deviations from the original planning and correct them.

In conclusion, with an estimated dedication of 30 hours per week and a total budget of (16x30) 480 hours, the project planning is feasible.

Next, some examples of potential sources of delays are mentioned.

2.5.1 Complexity of the platform

The Caffe deep learning framework is not a final product, and it is constantly being developed by the community. The lack of documentation and information may generate some problems or difficulties, which in turn may cause delays in the development of the optimizations.

2.5.2 Bugs

Considering that a deep learning framework is a relatively complex software, it is easy to introduce bugs while modifying the source code. This may also cause delays if bugs are not located and fixed fast enough.

2.5.3 Optimizations selection

A bad selection of the optimizations to be performed could result in a waste of time without a significant gain. This could cause delays with respect to the planning and carries the risk of not being able to finish the implementation of all the proposed optimizations.

2.5.4 Unavailability of the Jetson TX1 developer kit

It may happen that due to reasons beyond our control, we are unable to access remotely the machine, like for example a disconnection from the network, problems with credentials, etc. If the unavailability of the machine persists long enough, it could cause deviations in the project planning.

3 Budget and sustainability

3.1 Project budget

In order to carry out this project, the resources mentioned in the previous delivery will be needed. In this document, an estimation of the cost of the project is presented, taking into account the aforementioned hardware and software resources, and the corresponding amortizations. In addition, the indirect costs of the project are taken into consideration.

3.1.1 Hardware budget

In order to implement the optimizations previously explained, a set of hardware will be needed for different purposes.

Product	**Price**	**Units**	**Useful life**	**Amortization**
Toshiba Portégé R30-A-17E	900 €	1	5 years	75 €
Jetson TX1 developer kit	450 €	1	3 years	75 €
Total	1.350 €			150 €

Table 2: Hardware budget

Table 2 shows an estimation of the cost of the hardware used in the project, taking into account its useful life as well as its amortizations.

3.1.2 Software budget

Additionally, some software will be needed to carry out the project. Since this project is completely built with free tools, the total software cost is zero. Furthermore, the amortizations are also zero.

Product	Price	Units	Useful life	Amortization
JetPack (Jetson SDK)	0 €	1	-	0 €
Caffe	0 €	1	-	0 €
LaTeX	0 €	1	-	0 €
Atom	0 €	1	-	0 €
Git	0 €	1	-	0 €
GitHub	0 €	1	-	0 €
Evince	0 €	1	-	0 €
Vim	0 €	1	-	0 €
Google Docs	0 €	1	-	0 €
TeamGantt	0 €	1	-	0 €
Total	0 €			0 €

Table 3: Software budget

Table 3 shows an estimation of the cost of the software used in the project, taking into account its useful life as well as its amortizations.

3.1.3 Human resouces budget

This project is going to be developed by one person. Hence, this person will need to be a project manager, a software developer and a tester. Thus, each role is differentiated in the total of 450 hours. In table 4 an estimation of the human resources cost is provided.

Role	Hours	€/hour	Salary
Project manager	50	50	2.500 €
Software developer	300	35	10.500 €
Tester	100	30	3.000 €
Total	450		16.000 €

Table 4: Human resources budget

Next, table 5 provides a distribution of the time that each role spends in the different tasks of the project.

Task	**Duration (h)**	**Dedication (h)**		
		Project Manager	**Software developer**	**Tester**
Acquire background in DNN	70	20	30	20
Get familiar with Caffe	40	0	20	20
Jetson TX1 set up	10	0	10	0
Initial net characterization	30	10	20	0
Use of half-precision floats	90	0	70	20
Pruning and retraining	90	0	70	20
Use of cuSPARSE	60	0	40	20
Final net characterization	30	10	20	0
Final stage	30	10	20	0
Total	450	50	300	100

Table 5: Time estimation by role

3.1.4 Unexpected costs

Table 6 shows the number of extra hours resulting from the worst case of planning deviation. This allows to have a certain budget margin that will help against unexpected events that may occur during the execution of the different tasks.

Role	**Hours**	**€/hour**	**Salary**
Project manager	15	50	700 €
Software developer	15	35	525 €
Tester	10	30	300 €
Total	40		1.525 €

Table 6: Unexpected costs

3.1.5 Indirect costs

Next, table 7 provides an estimation of the indirect costs that are not comprised in any of the previous categories.

Product	**Price**	**Units**	**Estimated cost**
Electricity	0,07 €/kWh	1500 kWh	102 €
ADSL	35 €/month	4 months	140 €
Office supplies	150 €	-	150 €
Total			392 €

Table 7: Indirect costs

3.1.6 Total budget

By adding all the budgets provided above, the total estimated budget for this project is computed, as shown in table 8. Notice that a 5% of contingency has been added over the cumulative total in order to cover unexpected expenses that may occur during the course of the project.

Concept	**Estimated cost**
Hardware resources	150 €
Software resources	0 €
Human resources	16.000 €
Unexpected costs	1.525 €
Indirect resources	392 €
Subtotal	18.067 €
Contingency (5%)	903,35 €
Total	18.970,35 €

Table 8: Total budget

Finally, an estimation of the approximate budget per task is also provided in table 9. The cost of each task is computed taking into account the number of hours spent in the task and the required resources.

Task	**Estimated cost**
Acquire background in DNN	2.971,53 €
Get familiar with Caffe	1.483,73 €
Jetson TX1 set up	395,93 €
Initial net characterization	1.337,80 €
Use of half-precision floats	3.463,40 €
Pruning and retraining	3.463,40 €
Use of cuSPARSE	2.275,60 €
Final net characterization	1.337,80 €
Final stage	1.337,80 €
Total	18.067 €

Table 9: Estimated budget per task

3.2 Budget monitoring

In order to control the budget, at the end of each task the budget will be updated with the effective amount of hours, the cost of the resources used and the expenses of the unexpected events that may have occurred. These numbers will be compared with the previous estimations to obtain indicators that will show the amount of deviation from the initial budget planning. The following formulas will be applied:

$$\begin{aligned}
&\text{Cost deviation} = (EC - RC) \cdot RH \\
&\text{Consumption deviation} = (EH - RH) \cdot EC \\
\text{where:}& \\
&EH = \text{estimated hours} \\
&EC = \text{estimated cost} \\
&RH = \text{real hours} \\
&RC = \text{real cost}
\end{aligned}$$

Since the budget will be updated at the end of each task, this will help determine where the deviation has occurred, if it has occurred at all. In addition, since both the amount of hours spent in each task and the cost of the resources used in each task are tracked, this will help determine whether the deviation is due to a variance in cost or consumption.

Furthermore, a certain margin of deviation is allowed since the cost of unexpected events has been taken into account and a contingency percentage has been applied to the final budget estimation.

3.3 Sustainability and social commitment

With the object of identifying and evaluating the sustainability of the project, the impact in its environment will be analyzed regarding three dimensions: environmental, economical and social. This analysis will be based on the application of the sustainability matrix to the project, shown in table 10.

	PPP	**Useful life**	**Risks**
Environmental	Design consumption	Ecological footprint	Environmental risks
	9/10	18/20	-2/-20
Economical	Bill	Viability plan	Economical risks
	6/10	15/20	-2/-20
Social	Personal impact	Social impact	Social risks
	8/10	16/20	0/-20
Sustainability range	23/30	49/60	-4/-60
	68/90		

Table 10: Sustainability matrix of the project

3.3.1 Environmental dimension

The execution of this project uses the minimum amount of resources possible, limited only to the electricity required for the equipment to work. This fact limits the search of alternatives to reduce the consumption and the environmental impact. This also makes the reuse of resources difficult.

What is more, this project aims to reduce the energy consumption of inference in deep neural networks. Hence, not only it does not have almost any ecological impact, but also improves the ecological footprint of current solutions, which have the added cost of sending data to remote servers. The proposed solution does all the computation inside the device itself, thus all costs associated to the maintenance of networks, routers, switches and remote servers will be reduced.

3.3.2 Economical dimension

A detailed quantification of all the costs involved in the project has been done, both of material and human resources, as shown in previous sections of this document.

The proposed solution will be less expensive than current solutions from an economical point of view, simply because the ability to run inference directly on the device makes unnecessary a network connection. Hence, this solution will be cheaper for the end-user, since it will not require a data plan to work.

3.3.3 Social dimension

The execution of this project has not implied significant reflections in a personal, professional or ethic level since, in any case, it would only entail improvements regarding the execution time, memory usage or energy consumption of inference in deep neural networks.

As mentioned before, this is currently a hot topic and a lot of research is being carried out on this matter. Hence, collectives of researchers involved in this topic could benefit from this work. In addition, it could improve the quality of life of all users of applications powered by deep neural networks, since these systems could run faster and consume less.

The implementation of the technology and methods proposed in this project remains transparent to the users. The population, therefore, would experiment the aforementioned improvements passively, without noticing further change than a better user experience.

References

[1] H. Larochelle et al. "An empirical evaluation of deep architectures on problems with many factors of variation." In: *International Conference on Machine Learning* (2007), pp. 473–480. URL: http://www.dmi.usherb.ca/~larocheh/publications/deep-nets-icml-07.pdf.

[2] K. Jarrett et al. "What is the best multi-stage architecture for object recognition?" In: *Computer Vision, 2009 ...* (2009). Ed. by Ieee, pp. 2146–2153. URL: http://yann.lecun.com/exdb/publis/pdf/jarrett-iccv-09.pdf.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* (2012), pp. 1–9. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[4] Q. V. Le et al. "Building High-level Features Using Large Scale Unsupervised Learning". In: *International Conference on Machine Learning* (2012). URL: https://arxiv.org/pdf/1112.6209.pdf.

[5] Oriol Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: *CoRR* abs/1411.4555 (2014). URL: http://arxiv.org/abs/1411.4555.

[6] Chenyi Chen et al. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving". In: *CoRR* abs/1505.00256 (2015). URL: http://arxiv.org/abs/1505.00256.

[7] G. Dahl, T. Sainath, and G. Hinton. "Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout." In: *International Conference on Acoustics, Speech and Signal Processing* (2015). URL: http://www.cs.toronto.edu/~gdahl/papers/reluDropoutBN_icassp2013.pdf.

[8] T. Unterthiner et al. "Deep Learning as an Opportunity in Virtual Screening." In: *Proceedings of the Deep Learning Workshop at NIPS* (2014). URL: http://www.bioinf.jku.at/publications/2014/NIPS2014f.pdf.

[9] D. Bahdanau, K. Cho, and Y. Bengio. "Deep Visual-Semantic Alignments for Generating Image Descriptions." In: *Proceedings of the Annual Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). URL: http://cs.stanford.edu/people/karpathy/cvpr2015.pdf.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014). URL: http://arxiv.org/abs/1409.0473.

[11] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016). Ed. by Ieee. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf.

[12] *Prisma app.* https://en.wikipedia.org/wiki/Prisma_(app). Accessed: 2017-02-24.

[13] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). URL: http://arxiv.org/abs/1506.01497.

[14] *Caffe.* http://caffe.berkeleyvision.org/. Accessed: 2017-02-24.

[15] *Jetson TX1 Developer Kit.* http://www.nvidia.com/object/jetson-tx1-dev-kit.html. Accessed: 2017-02-24.

[16] Artem Vasilyev. "CNN optimizations for embedded systems and FFT". In: (2015). URL: http://cs231n.stanford.edu/reports/tema8_final.pdf.

[17] Preetum Nakkiran et al. "Compressing deep neural networks using a rank-constrained topology". In: *Sixteenth Annual Conference of the International Speech Communication Association* (2015). URL: http://preetum.nakkiran.org/pubs/rcnn_google.pdf.

[18] Song Han, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding". In: *CoRR* abs/1510.00149 (2014). Ed. by Ieee, pp. 4087–409. URL: http://arxiv.org/abs/1510.00149.

[19] Yong-Deok Kim et al. "Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications". In: *CoRR* abs/1511.06530 (2015). URL: http://arxiv.org/abs/1511.06530.

[20] Guoguo Chen, Carlos Parada, and Georg Heigold. "Small-footprint keyword spotting using deep neural networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference* (2015). URL: http://arxiv.org/abs/1510.00149.

[21] *How to add a brain to your smart phone.* https://petewarden.com/2014/04/08/how-to-add-a-brain-to-your-smart-phone/. Accessed: 2017-02-25.

[22] *Qualcomm Zeroth is advancing deep learning in devices.* https://www.qualcomm.com/news/onq/2015/03/02/qualcomm-zeroth-advancing-deep-learning-devices-video. Accessed: 2017-02-25.

[23] *How Google Translate squeezes deep learning onto a phone.* https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html. Accessed: 2017-02-25.

[24] *Delivering real-time AI in the palm of your hand.* https://code.facebook.com/posts/196146247499076/delivering-real-time-ai-in-the-palm-of-your-hand/. Accessed: 2017-02-25.

[25] *Caffe2.* https://github.com/caffe2/caffe2. Accessed: 2017-02-25.

[26] *cuSPARSE.* https://docs.nvidia.com/cuda/cusparse/. Accessed: 2017-02-27.

[27] *Git.* https://git-scm.com/. Accessed: 2017-02-26.

[28] *GitHub*. https://github.com/. Accessed: 2017-02-26.

[29] *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/. Accessed: 2017-03-3.

[30] *Neural Networks for Machine Learning*. https://www.coursera.org/learn/neural-networks. Accessed: 2017-03-1.

[31] Ross B. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). URL: http://arxiv.org/abs/1311.2524.

[32] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). URL: http://arxiv.org/abs/1504.08083.

[33] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). URL: http://arxiv.org/abs/1506.01497.

[34] Suyog Gupta et al. "Deep Learning with Limited Numerical Precision". In: *CoRR* abs/1502.02551 (2015). URL: http://arxiv.org/abs/1502.02551.