UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

FINAL DEGREE THESIS

**Bachelor's degree in Industrial Electronics and Automatic Control Engineering**

# DESIGN AND IMPLEMENTATION OF A MOBILE PHONE APPLICATION TO HELP PEOPLE WITH VISUAL DYSFUNCTION VISUALLY INSPECT THEIR SURROUNDING SPACES

## Report and Annexes

| | |
|---|---|
| **Author:** | Pau Besses Casas |
| **Director:** | Edmundo Guerra Paradas |
| **Co-Director:** | Antoni Grau Saldes |
| **Call:** | June 2021 |

# Abstract

This project consists in the development of software that helps people with visual impairment move and get along in indoor spaces, which might probably be their personal and domestic surroundings.

This software is meant to allow its user to take a photo of the environment that surrounds him and give him an oral response that explains some of the characteristics of the taken picture, therefore defining the space that the person wants to analyse. Furthermore, the user must be capable of letting the application know what in particular he wants to graphically examine.

The user runs the mobile phone application each time he wants to use it, operating it through voice commands. In order to detect, recognize and inspect the surrounding objects and environments, Deep Learning and cloud technologies are used to provide the computational efforts and communications.

An evaluation of the accuracy and robustness of the neural networks has been performed at the same time than they have been developed in order to design and implement solutions that make them more reliable. Programming languages for the creation of software applications and communication protocols have been successfully implemented to develop the fully functional software.

# Resum

Aquest projecte consisteix en el desenvolupament de software amb l'objectiu d'ajudar a persones amb discapacitat visual a moure's i ubicar-se en espais interiors, que probablement siguin el seu entorn personal i domèstic.

Aquest software està dissenyat per permetre al seu usuari fer una foto de l'entorn que l'envolta i donar-li una resposta oral que expliqui algunes de les característiques de la fotografia, definint per tant l'espai que la persona vol analitzar. A més, l'usuari ha de ser capaç de fer saber a l'aplicació què vol examinar gràficament en particular.

L'usuari executa l'aplicació mòbil cada vegada que la vol utilitzar, operant-la mitjançant ordres de veu. Per tal de detectar, reconèixer i inspeccionar els objectes i entorns circumdants, s'utilitzen tecnologies d'aprenentatge profund i xarxes d'interacció entre dispositius per proporcionar els esforços computacionals i les comunicacions.

S'ha realitzat una avaluació de la precisió i robustesa de les xarxes neurals al mateix temps que s'han anat desenvolupant per tal de dissenyar i implementar solucions que les facin més fiables. S'han implementat llenguatges de programació per a la creació d'aplicacions software i protocols de comunicació amb èxit per tal desenvolupar el programari funcional en la seva totalitat.

# Resumen

Este proyecto consiste en el desarrollo de software con el objetivo de ayudar a personas con discapacidad visual a moverse y ubicarse en espacios interiores, que probablemente sean su entorno personal y doméstico.

Este software está diseñado para permitir a su usuario hacer una foto del entorno que le rodea y darle una respuesta oral que explique algunas de las características de la fotografía, definiendo por tanto el espacio que la persona quiere analizar. Además, el usuario debe ser capaz de hacer saber a la aplicación qué quiere examinar gráficamente en particular.

El usuario ejecuta la aplicación móvil cada vez que la quiere utilizar, operándola mediante comandos de voz. Con el fin de detectar, reconocer e inspeccionar los objetos y entornos circundantes, se utilizan tecnologías de aprendizaje profundo y redes de interacción entre dispositivos para proporcionar los esfuerzos computacionales y las comunicaciones.

Se ha realizado una evaluación de la precisión y robustez de las redes neurales a medida que se han ido desarrollando con el fin de diseñar e implementar soluciones que las hagan más fiables. Se han implementado lenguajes de programación para la creación de aplicaciones software y protocolos de comunicación con éxito para desarrollar el software funcional en su totalidad.

# Appreciations

I would like to thank the director and co-director of this Final Degree Thesis, Edmundo Guerra Paradas and Antoni Grau Saldes, for guiding me through each step of this project.

I also want to thank my parents, Anna Maria and Àlex, my grandparents, Caterina and Vicens, and Júlia for supporting me during the entire development of this project.

# Glossary

AI - Artificial Intelligence – Set of mathematic and computational techniques and algorithms that let machines learn and solve problems by mimicking the ways humans do.

GDP - Gross Domestic Product.

NN - Neural Network – Algorithm that is based on the functioning of human brains in order to find patterns and relationships between data in order to solve problems.

CNN - Convolutional Network – Complex neural network commonly used to analyse visual data, inspired on animal's visual cortex.

RCNN – Region Based Convolutional Network – CNN that selects the most relevant image regions and analyses them in order to inspect the overall image.

SDK - Software Development Kit – Set of software development tools grouped in a package.

HTTP – Hypertext Transfer Protocol – Connectivity protocol that permits information transfer between sites in the World Wide Web.

CPU – Central Processing Unit – Electronic component that computes the core instructions of a computer program.

GPU – Graphic Processing Unit – Electronic component that has the specific objective of manipulating and computing graphical data in a computer (such as images).

# Index

# 1. Preface

## 1.1. Origin of the work

Deep Learing is one of the youngest technological fields, but still one of the most powerful advances in industry, science and society.

This is because it is an area of knowledge that presents an exponential growth, as does for example the area of micro and nanoelectronics. This great acceleration in the research, creation and implementation of artificial intelligence software is due to the fact that the benefit generated by these tool for both companies and society represents a turning point in productivity, effectiveness and well-being that no tool before had been able to provide. Therefore, over the last few decades there has been a major interest in innovation in these technologies that has led us to an era where not only is there a lot of software created and constantly being implemented, but an era where this technology is reaching everyone's hands, whatever the their knowledge on the subject is.

The following figures briefly show the growth and economic impact of artificial intelligence:

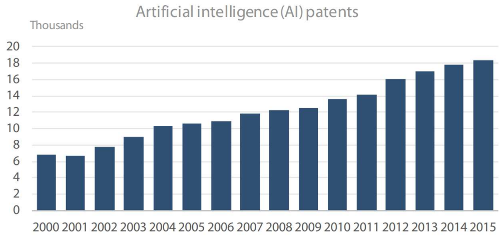


**Figure 1.1.1.** Number of AI patents worldwide from year 2000 to 2015, in thousands (Source: EPRS | European Parliamentary Research Service, **"**Economic impacts of artificial intelligence" __ Bibliography .2)

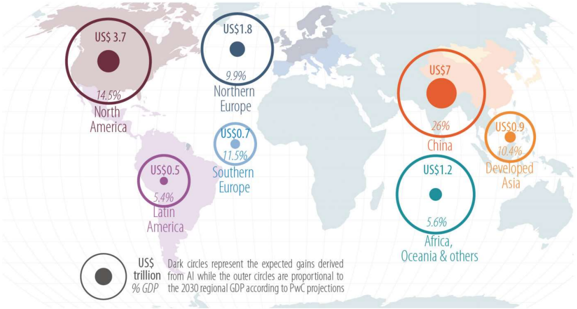


**Figure 1.2.2.** Expected gains from AI in different regions of the world by 2030 (Source: EPRS | European Parliamentary Research Service, **"**Economic impacts of artificial intelligence" __ Bibliography .2)

One of the most exploited fields of study and with the most pronounced growth in the last years is computer vision; the economic value of which is expected to grow incessantly. Growth due to the immense range of applications that computer vision has: From medicine, surgery and neuroscience to the world of entertainment, in video games and multimedia content, through countless fields where this technology has resulted a major breakthrough.

The potential of artificial intelligence to solve complex problems that previously could not be solved and the great saving and optimization of time and resources involved in studying and interpreting large and complex amounts of data, make machine learning and deep learning (one of the most characteristic tools of artificial intelligence) capable of analysing the pixels that make up images and frames of videos at a very high speed and with a high efficiency. In this way neural networks are able to analyse large clusters of pixels by treating them as data sets interconnected with each other, following patterns and forming structures. Thus, studying the relationships between each and every one of the pixels that make up an image, neural networks can draw very complex conclusions from the analysed images and videos.

The rapid growth and powerful development of artificial intelligence technologies is also thanks to the fact that they are based on software, which is supported and run on hardware, which not only is highly powerful and efficient but is also growing at a really rapid rate. In fact, artificial intelligence has the advantage of being developed over tools that have been studied and improved for a much longer time than the AI itself.

As an industrial electronics and automatic control engineering student, I have had the opportunity to experience the ease of studying, developing, and applying artificial intelligence techniques and tools. Although research in this field of engineering is always constant, there is a lot of software created and resources developed that have already been used to solve problems and that can be redesigned, restructured and / or reimplemented in order to solve new problems. This way, it is relatively easy and effective to create completely new AI software able to provide new benefits to society, made from tools that have been already designed and implemented.

It is for this reason that this project is based, for the most part, on studying existing AI tools that already work in order to redesign them and establish certain interconnections and operating structures between them, so that a new tool capable of solving a set out problem and covering a specific need of the society is created.

# 2. Introduction

## 2.1. Project objectives

The main objective of this project is the creation of a software infrastructure that increases the standard of living of people with visual impairment. In the present work, this improvement of life standards will be achieved by providing people an alternative to seeing the world that is not their eyes. Concretely, this project's objective is helping people visualize their surrounding spaces.

The benefits that this software provides are direct and personal, in other words they just have an impact on the user that utilizes the software. So the created tools in this project do not benefit the entirety of a collective but are meant to facilitate the quotidian life of each and every person that uses them.

The collective the software has the objective of being helpful for is all the people that have any type of visual difficulties. These difficulties may go from total blindness to any degree of partial blindness or vision loss. In fact, the software created in this project has the objective of helping all these people that believe that by using it, their comfort and security when being in or moving through some places can increase.

The secondary objective of this project is studying, exploring and creating software tools, methods and techniques that might be useful for the resolution of future problems, which may have from similar to completely different purposes from those present in this project.

The achievement of this secondary objective, which can be translated as specific software objectives, must result in the achievement of the main objective of the project.

The specific software objectives of this project are:

- The creation of a software application completely adapted for its use by people with visual dysfunction. Thus, a fully voice-operated software.

- The design, training and testing of deep learning networks that analyse graphic data. This objective will be achieved by:
  - Implementing and structuring already trained networks in order to carry out general analysis of images, classifying the inspected images by the place that they represent

  (bedroom, square, kitchen, etc.) and recognizing the object that occupies the largest area on the picture.

  - Designing, training and validating convolutional networks so that they are capable of detecting, classifying and locating different classes of objects in the analysed images.

- The collection, selection, documentation and treatment of data in order to constitute appropriate datasets for the training and testing of the networks.
- The programming of software that imports the images to be analysed, passes them through the AI networks and converts the conclusions drawn from the image inspection into results interpretable by the user.
- The design and implementation of an efficient infrastructure that connects the different elements and platforms that may form the system. The connection has to be fast, must let each element send and receive different types of information and must not be interrupted or annulated when no information is being sent.
- Treatment of cloud storage services, in order to create the infrastructure previously mentioned, if needed.

# 3. Design of the System Architecture

The first step in developing the software is structuring the elements that will constitute the software, by assigning specific functions and tasks to each element and designing the interactions and connections between them.

To do it, it is needed to formulate all the relevant tasks and functions the software should perform in order to achieve the objectives of the project. These tasks and functions should not be neither too specific (which would result in a large number of variables that could be grouped in order to simplify the organization and structuring of the software) nor too generic (which would result in indivisible sets of tasks and functions that could not be assigned to any specific software element).

Then these tasks will be assigned to specific elements that will form the totality of the structure, by choosing which element within our reach best fits each function the system has to carry out. Finally the design of the infrastructure that will connect each of the system elements and the connections and interaction protocols between them have to be done.

## 3.1. General architecture

We firstly design the general architecture that will constitute the connections between the main elements of the software.

The tasks and functions the developed software in this project has to be able to perform are:

- Let the user take a picture of the environment or space he chooses.
- Let the user communicate whether he wants to get a general description of the taken picture or know if there is a specific object on the image and, in case it was, where it is located.
- Process the taken picture in order to make it suitable to be analysed by the neural networks.
- Analyse the picture with the appropriate convolutional network depending on the orders given by the user.
- Elaborate an answer according to what the user wants to know, based on the conclusions made by the neural network about the inspected image.
- Communicate the final answer to the user via audio.
- Set everything ready for the next image inspection.

Now some considerations have to be done for the appropriate assignment of the mentioned tasks to software tools and platforms and hardware devices.

The user will be in contact with the program through his or her mobile phone, as much of the population is familiar with these devices and uses them in everyday life due to their ease of use and accessibility. Furthermore, nowadays mobile phones have software that enables the use of the vast majority of their applications and tools to all people who cannot receive visual information from the device. This visual aid software for the visually impaired is also available on other electronic devices such as computers, tablets, smart TVs and home automation systems. As Felipe Yagüe comments, "".

In addition, smartphones have a great versatility and ease of use that are perfect for the tasks of taking images, communication of commands by the user and information of the results to the user.

On the other hand, the power, efficiency and speed of the latest phones and computers are at very similar levels, with obvious differences between devices but with lines of research, development and integration of systems virtually identical. However, as a student, I have always designed and implemented artificial intelligence tools and data processing software from computers.

I also have experience solving computer vision problems across platforms and programming languages that the vast majority of programmers and developers use on computers. And while there are alternatives and versions of these platforms and languages to be used in smartphones and other devices, the most familiar programming and coding environment to me is the computer.

Therefore, the designing, modelling, training and testing of neural networks, all the image processing and analysing and the complex mathematical and logistic operations will be developed and executed on a computer.

Having done these considerations we proceed to the assignment of the tasks and functions the system has to carry out to the different elements that will form it:

- Smartphone:
    - Let the user take a picture of the environment or space he chooses.
    - Let the user communicate whether he wants to get a general description of the taken picture or know if there is a specific object on the image and, in case it was, where it is located.
    - Communicate the final answer to the user via audio.
    - Set everything ready for the next image inspection.
- Computer:
    - Process the taken picture in order to make it suitable to be analysed by the neural networks.

- Analyse the picture with the appropriate convolutional network depending on the orders given by the user.
- Elaborate an answer according to what the user wants to know, based on the conclusions made by the neural network about the inspected image.
- Set everything ready for the next image inspection.

In order to establish an organized, structured and fluid communication between the computer and the smartphone a cloud storage platform could be used. This way, with the only requirement of having an internet connection, both devices could upload and download information from the same repository establishing a dialog between them.

As I personally have experience with the Google Firebase platform for the development of mobile phone applications, and due to its high-level programming and its ability to be of use for many platforms and programming languages, we will build the bridge between the user interface and the image treatment and analysis program with this cloud database service.

Google Firebase is a free platform that is programmed using high level programming language and that provides different products, explained below, that allow the development of communication protocols that make possible the connection between the devices that are used in this project.

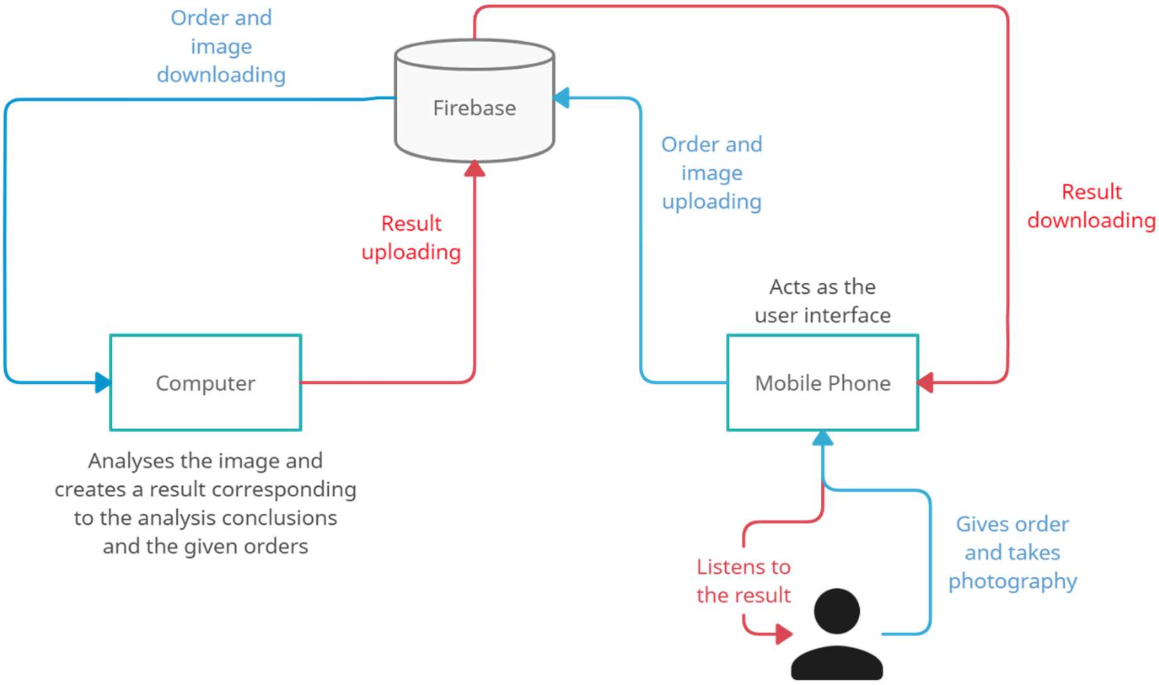

Order and image downloading
Firebase
Order and image uploading
Result downloading
Result uploading
Acts as the user interface
Computer
Mobile Phone
Analyses the image and creates a result corresponding to the analysis conclusions and the given orders
Gives order and takes photography
Listens to the result

**Figure 3.1.1.** Global architecture of the system

## 3.2. Local architectures and connectivity

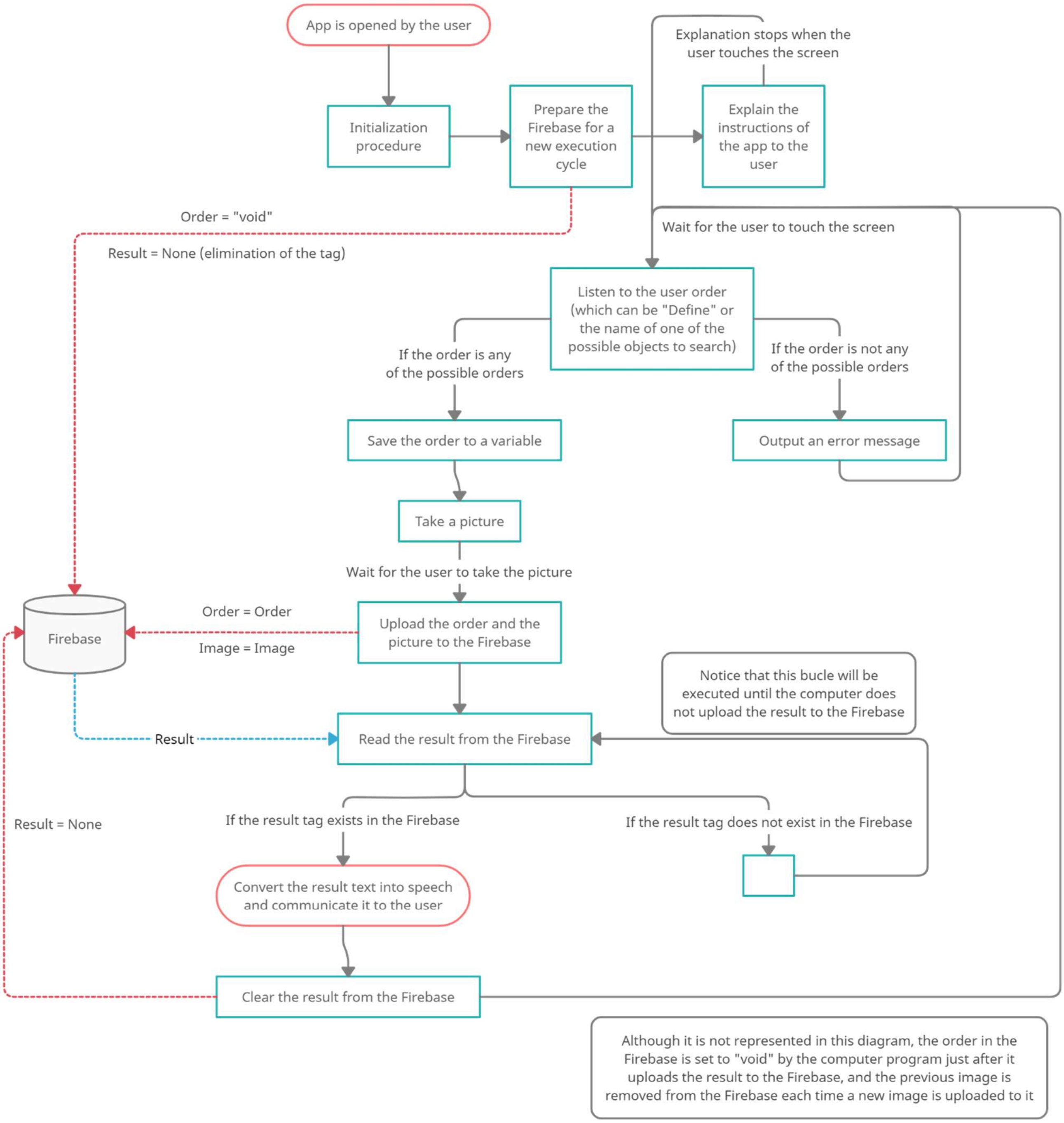

App is opened by the user
Initialization procedure
Prepare the Firebase for a new execution cycle
Explanation stops when the user touches the screen
Explain the instructions of the app to the user
Order = "void"
Result = None (elimination of the tag)
Wait for the user to touch the screen
Listen to the user order (which can be "Define" or the name of one of the possible objects to search)
If the order is any of the possible orders
If the order is not any of the possible orders
Save the order to a variable
Output an error message
Take a picture
Wait for the user to take the picture
Firebase
Order = Order
Image = Image
Upload the order and the picture to the Firebase
Notice that this bucle will be executed until the computer does not upload the result to the Firebase
Result
Read the result from the Firebase
Result = None
If the result tag exists in the Firebase
If the result tag does not exist in the Firebase
Convert the result text into speech and communicate it to the user
Clear the result from the Firebase
Although it is not represented in this diagram, the order in the Firebase is set to "void" by the computer program just after it uploads the result to the Firebase, and the previous image is removed from the Firebase each time a new image is uploaded to it

**Figure 3.2.1.** Architecture of the smartphone program and its connections and interaction with the Firebase

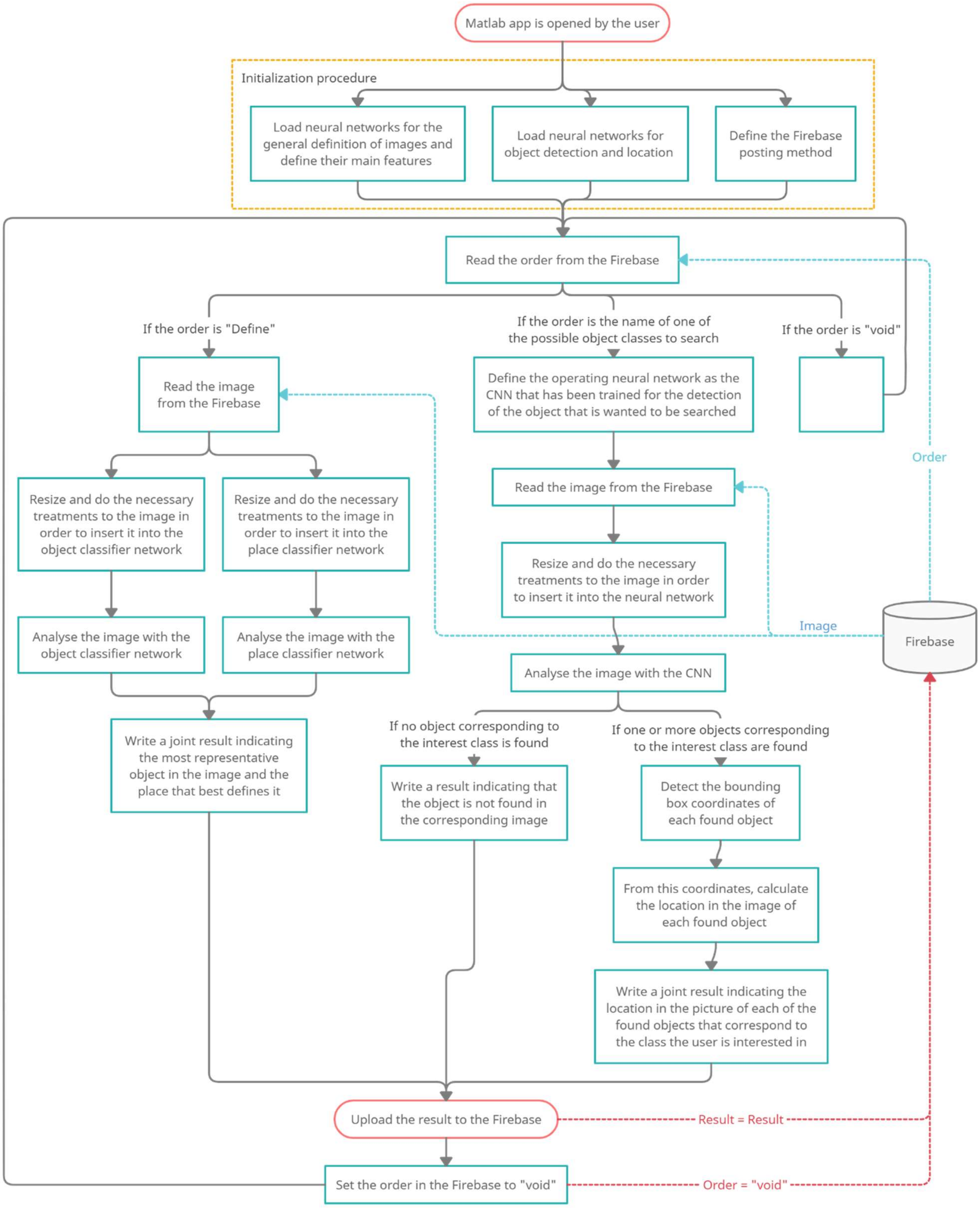

Matlab app is opened by the user
Initialization procedure
Load neural networks for the general definition of images and define their main features
Load neural networks for object detection and location
Define the Firebase posting method
Read the order from the Firebase
If the order is "Define"
If the order is the name of one of the possible object classes to search
If the order is "void"
Read the image from the Firebase
Define the operating neural network as the CNN that has been trained for the detection of the object that is wanted to be searched
Order
Resize and do the necessary treatments to the image in order to insert it into the object classifier network
Resize and do the necessary treatments to the image in order to insert it into the place classifier network
Read the image from the Firebase
Resize and do the necessary treatments to the image in order to insert it into the neural network
Image
Firebase
Analyse the image with the object classifier network
Analyse the image with the place classifier network
Analyse the image with the CNN
Write a joint result indicating the most representative object in the image and the place that best defines it
If no object corresponding to the interest class is found
If one or more objects corresponding to the interest class are found
Write a result indicating that the object is not found in the corresponding image
Detect the bounding box coordinates of each found object
From this coordinates, calculate the location in the image of each found object
Write a joint result indicating the location in the picture of each of the found objects that correspond to the class the user is interested in
Upload the result to the Firebase
Result = Result
Set the order in the Firebase to "void"
Order = "void"

**Figure 3.2.2.** Architecture of the computer program and its connections and interaction with the Firebase

## 3.3. Matlab

The computer software will be developed on Matlab. The reasons why are explained below:

Matlab is a programming language that is based on matrices, as the core elements that it uses to perform mathematic computations are vectors and arrays, which allows it to perform complex multi-variable and multi-dimensional operations from a very low language level. This makes Matlab be an incredibly efficient tool to develop prototypes with state of the art techniques. Therefore, in its highest level, which is the programming language provided to its users, this programming platform is able to find efficient solutions to really complex problems.

The problems that Matlab is able to deal with go far beyond maths. This language is used by engineers, scientists and companies to develop a really varied range of applications that go from data analysis, to internet connections and HTTP protocols, passing through automatic control and many other objects of study.

What makes Matlab a rich software development platform is that it is, in a great part, developed by its community. This is due to the fact that there exist community forums developed by Matlab and the users themselves which have the objective of allowing collaboration between them. Software is shared between the users and doubts about the programming language are usually solved by the community. Moreover, the users themselves and their experience and discoveries make Matlab be in a state of continuous improvement.

This programming language provides the opportunity of rapidly creating code and testing it in at the same time it gives great stability and security when large programs are being developed. This is due to the workspace that it offers, which allows the evaluation of code at the same time it is created.

Matlab allows its users to create applications, which can be compiled and executed as computer programs. Adding robustness and performance to the developed systems.

The software created in this project will be developed as a computer application, programmed with the Matlab App Designer. The design, evaluation and correction of each part of code that forms the computer application will be done in the Matlab Live Editor.

This programming language is a library-based language. Libraries are known as Toolboxes and provide complex functionalities that widely expand the capabilities of the platform.

Multiple Artificial Intelligence and Deep Learning Toolboxes will be used in this project in order to carry out the analysis of the images that the user of the application wants to inspect and with the objective of arriving to the conclusions that answer the user's questions about the environment that surrounds him.

I have personal experience with this programming language as it is commonly used in research and educational environments such as universities.

However, some disadvantages are related to the use of this programming language:

It costs money. Users can only use Matlab for free with a free trial of 30 days or if they have an academic authorization, which is usually given by universities to their students. This represents an impediment as we firmly have to suppose that the user of the software developed in this project wants to use it for a period of time longer than 30 days and that he does not have an academic authorization.

Therefore, the user of the developed software in this project is forced to buy Matlab. A home license, which is enough for the user in order to execute the application software, costs 119€ (price on June of 2021).

The following are Matlab web sites that correspond to some of its previously explained features and products:

- Matlab: https://es.mathworks.com/products/matlab.html (Bibliography .6)
- Matlab App Designer: https://es.mathworks.com/products/matlab/app-designer.html (Bibliography .7)
- Matlab Live Editor: https://es.mathworks.com/products/matlab/live-editor.html (Bibliography .8)

## 3.4. Mit App Inventor

The smartphone software will be developed on Mit App Inventor.

Mit App Inventor is a development platform for software applications. The applications created on this platform can be executed on Android and iOS operating systems. This environment is located on the web and allows the software development directly from it, without the need of having to install any software to the computer.

This development environment uses a visual programming language based on blocks, which are used in the programming workspace formed by a GUI (Graphical User Interface). Blocks are dragged to the GUI and interconnected between them in order to construct processes that will be executed when the created application runs. Each block has properties that can be edited in order to model their behaviour and function. When observing this programming method based on visual blocks, it may recall other programming languages such as Scratch.

Mit App Inventor is ideal for creating user interfaces as it allows the design of the elements that appear in the screen, their structuring and their graphical interactions. The developer is able to be aware of how the screen will look like, in every moment, on the device where the application is executed, and he or she is capable of modelling it.

An important characteristic of Mit App Inventor is that it is a free platform.

However there are limitations in the use of this programming language:

Its ease of use and programming rapidity is achieved at a cost of its simplicity and its severe limitations and impediments when trying to develop complex applications that require a deeper understanding in software designing and developing. The block-based visual language that Mit App Inventor uses makes its blocks be the possibilities and limitations of the platform at the same time, as programming cannot go beyond the capabilities of the blocks themselves.

Mit App Inventor was first developed for the programming of applications, and the possibility of the created apps to be executed on iOS systems is relatively new. Therefore, issues and disadvantages can be found when using the developed apps on iOS systems.

Nevertheless, this limitations do not affect this project as the application that acts as the user – software interface can be perfectly designed, programmed and implemented using the development services that Mit App Inventor provides. We will make use of some of the most complex blocks in the programming environment.

The following is the web site link of Mit App Inventor: https://appinventor.mit.edu/ (Bibliography .9)

## 3.5. Networks and comunications

As previously argued, the bridge between the computer program and the smartphone application will be made with Firebase, which is a platform developed by Google for software development support. Firebase has many products that are used in order to make software applications capable of managing data in a really simple way. The Firebase products used in this project are based on cloud storing, we will use them as a database for our nominal data (orders and results) and graphic data (images).

For the management of the nominal data the “Realtime Database” platform will be used, for the management of the images the “Storage” platform will be used. With “Realtime Database” data is stored as JSON and is actualized automatically each time a client uploads or retrieves data. The “Storage” platform is a Firebase SDK that enables the storing of multimedia files such as images and videos, it is also actualized automatically each time a client does an operation to the database.

On both platforms, all the operations between the database and the clients are synchronous, fast and are done under security protocols that are malleable by the programmer.

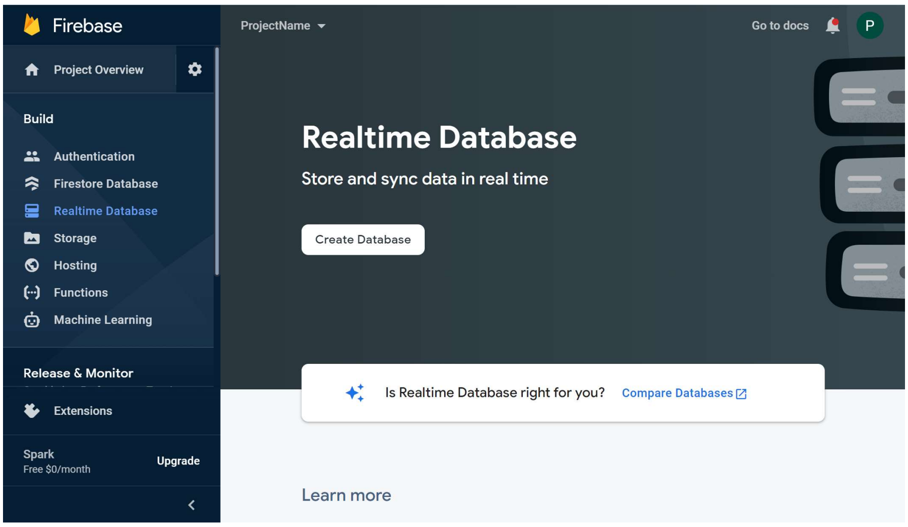
It is important to emphasize that the Firebase services that are going to be used in this project, as well as much others from this platform, are completely free.

Firebase website: https://firebase.google.com/ (Bibliography .1)

In order to create a Firebase project, enter the previously cited web site, create a user (if you don’t have one), log in and click “Go to console” in order to visualize all the created projects and to add new ones. To add a new one click the “Add project” button, define the project name and select whether you want to enable or disable the “Google Analytics” service, which provides application development tools with extended and advanced utilities that are not of use for this project. We will not enable the “Google Analytics” service. Having done these steps the project will be ready to use.

### 3.5.1. Realtime Database

To create a cloud database for nominal data stored as JSON, we enter the created project and go to Realtime Database. The next image corresponds to the Realtime Database creation window of a project that we have named “Project Name”. At the centre-left top of the window the project name is displayed, at the left appears a list with some of the Firebase products. To create the database click on the “Create Database” button.



Realtime Database

Store and sync data in real time

Is Realtime Database right for you?

**Figure 3.5.1.** Realtime Database creation window

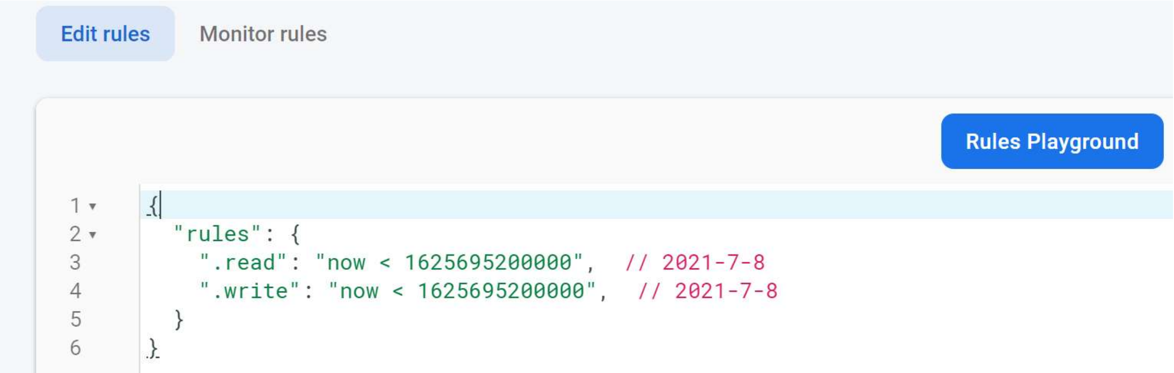
Then, we choose the location of the server where the database will be stored, which can be the United States (us-central 1), Belgium (europe-west 1) or Singapore (asia-southeast 1). Belgium has been chosen as the location of this project's database.

Now, the security rules can be set up. There are two main security modes:

- The locked mode only lets authorized clients upload and download data from the database, which implies a deeper security barrier to threats and anomalies. However, for this project such security protocols are not needed since we are not creating commercial software with loads of users. Moreover, in case we were creating a commercial app or the project is expanded in the future, each user will have their own database so that there do not exist interferences between clients and the information of each database will be completely independent from the others.
- The test mode allows every client that sends a reading or writing request to the database to connect to it and carry out the corresponding operation. Nevertheless, the client needs the database URL and the database connection code in order to be able to read or write to it, which is understood as a secure enough protocol for this project as no delicate or personal data of any type is handled and the codes and URLs of the databases will be known by nobody but the software developer. However this mode is disabled in 30 days and if the security rules are not updated the database ends up being unusable.

We will work with the test mode security protocol.

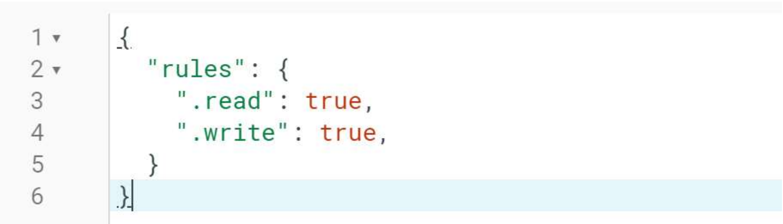
In order to eliminate the 30-day trial period, enter the created Realtime Database, we click on the "Rules" button and then go to "Edit rules". The next figure corresponds to the default rules set by the test mode.



```
{
  "rules": {
    ".read": "now < 1625695200000",  // 2021-7-8
    ".write": "now < 1625695200000",  // 2021-7-8
  }
}
```

**Figure 3.5.2.** Test mode preset rules of the Realtime Database

As it can be seen in the code above, the reading and writing operations by any client that correctly requests the connection to the database are enabled, but will be disabled on the date 2021 – 7 – 8, which is 30 days after the creation of the database (created on the 2021 – 6 – 8). In order to make the reading and writing period indefinite, we edit the rules as shown in the next image.



```
{
  "rules": {
    ".read": true,
    ".write": true,
  }
}
```

**Figure 3.5.3.** Test mode rules of the Realtime Database, modified in order to indefinitely enable the read and write operations

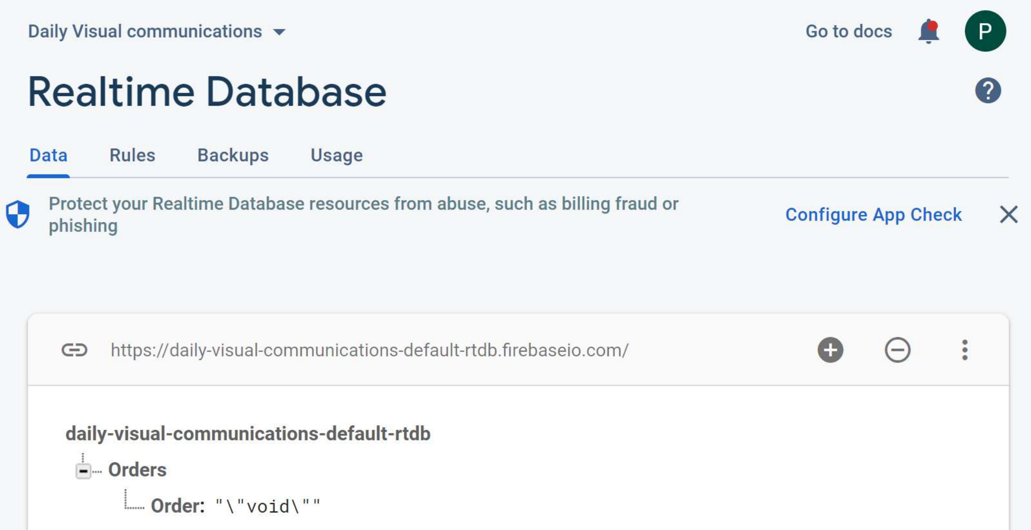
Once the database is set, notice there is a main repository with the name of the project. In this repository, data and data buckets can be created by clicking the "+" sign next to the repository name. Data is stored as tag – value pairs in JSON format, buckets are sub-repositories where local data is stored so that different sets of data can be managed independently.

In our database there are two buckets:

- The "Orders" bucket that contains the tag – value pair where the tag is "Order" and the value is the word that describes the order given by the user. This bucket always exists and the value of the tag – value pair is equal to "void" when there is no order given, is equal to "Define" when the user wants a general description of the image, or is equal to the name of the object the user wants to search (if it is one of the valid objects to search).
- The "Results" bucket that contains the tag – value pair where the tag is "Result" and the value is the phrase that describes the result given by Matlab. This bucket just exists since Matlab uploads the result to the database until Mit App Inventor removes it from the database (which

results in a time lapse of seconds or tenths of second), and is created again when Matlab uploads the new result. The largest part of the time, this bucket does not exist.

The next image shows the workspace of the Realtime Database used in this project. As it can be observed, the name of the Firebase project is “Daily Visual communications”, the connection URL of the database is https://daily-visual-communications-default-rtdb.firebaseio.com/ , the name of the main database repository is “daily-visual-communications-default-rtdb”, the “Orders” bucket contains the data “Order: void” where “Order” is the tag and “void” is the value, and the “Results” bucket does not exist as in this specific moment there is not any result to handle.



Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://daily-visual-communications-default-rtdb.firebaseio.com/

daily-visual-communications-default-rtdb
Orders
Order: "\"void\""

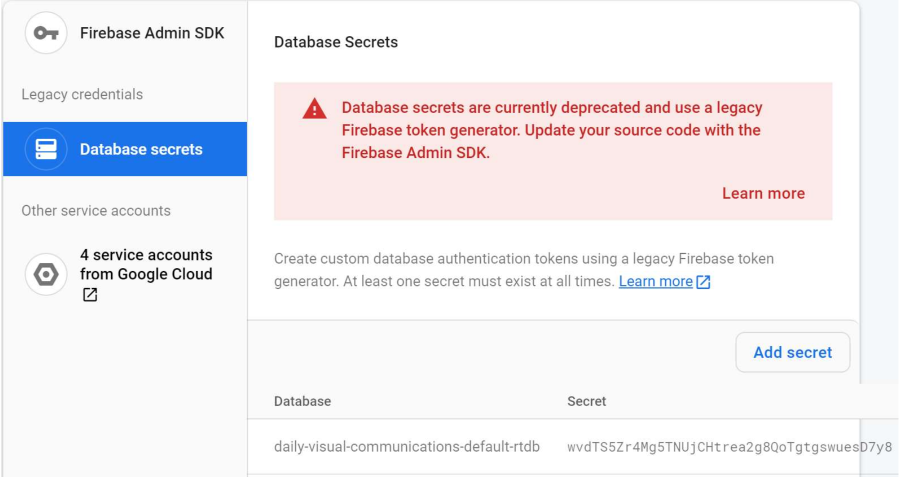
**Figure 3.5.4.** Realtime Database of the Firebase project that constitutes the bridge between the smartphone and the computer

### 3.5.2. Connection between the Realtime Database and Mit App Inventor

The Mit App Inventor development environment has a specific component, in the “Experimental” section, which enables the connection of the app with the Firebase and permits operations such as writing to and reading from the database, and removing data from it. The component is named “FirebaseDB”. A component can just work on a specific bucket, reason why we will use two FirebaseDB components (one for the “Orders” bucket and another for the “Results” bucket).

The component has some properties that have to be configured:

- Database secret: It’s a nominal and numerical character row that consists on a credential that authorizes the Mit App Inventor program to connect to the database. To know the database secret of the Firebase project, we go to the configuration icon on the top - left of the project screen and click on "Users and permissions". Finally, we click on "Service accounts" and on "Database secrets". The next image corresponds to the database secret screen of this Firebase project.



Firebase Admin SDK

Legacy credentials

Database secrets

Other service accounts

4 service accounts from Google Cloud

Database Secrets

Database secrets are currently deprecated and use a legacy Firebase token generator. Update your source code with the Firebase Admin SDK.

Learn more

Create custom database authentication tokens using a legacy Firebase token generator. At least one secret must exist at all times. Learn more

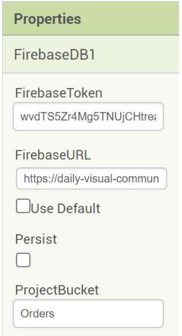
Add secret

| Database | Secret |
|---|---|
| daily-visual-communications-default-rtdb | wvdTS5Zr4Mg5TNUjCHtrea2g8QoTgtgswuesD7y8 |

**Figure 3.5.4.** Database secret of the Firebase project that constitutes the bridge between the smartphone and the computer

- Firebase URL: Consists on the connection URL of the Realtime Database, shown on the *Figure 3.5.3*.
- The name of the project bucket that will contain the data managed by the FirebaseDB component.

The next image corresponds to the FirebaseDB component created for the managing of the orders data on a bucket named "Orders". We have named this component FirebaseDB1. Another component named FirebaseDB2 has been created in order to manage the results data on a bucket named "Results".



Properties

FirebaseDB1

FirebaseToken: wvdTS5Zr4Mg5TNUjCHtre‹

FirebaseURL: https://daily-visual-commun

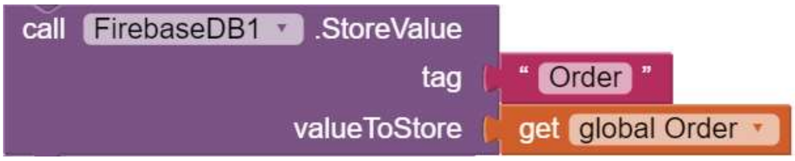
☐ Use Default

Persist ☐

ProjectBucket: Orders

**Figure 3.5.5.** Properties of the FirebaseDB1 component

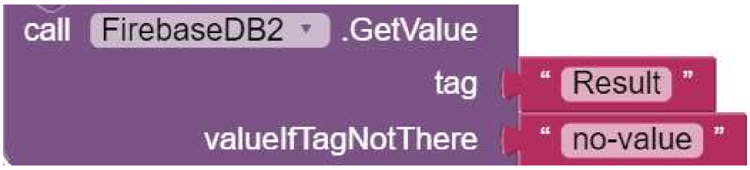
The next figure shows the method that writes a tag – value pair to the Realtime Database, where the tag is set as "Order" and the value corresponds to the value stored in a global variable named "Order". The data is stored in the "Orders" bucket as the FirebaseDB1 component is used.



```
call FirebaseDB1 .StoreValue
              tag  " Order "
     valueToStore  get global Order
```

**Figure 3.5.6.** FirebaseDB writing method example

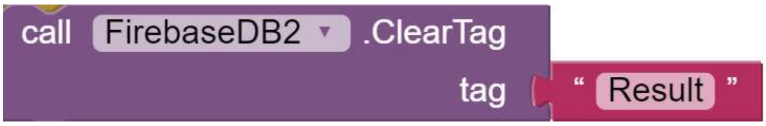
The following image shows the method used to read the value corresponding to a specific tag from the Realtime Database, where the tag is "Result". In this case, if the tag is not found in the database, the variable that was meant to store the value linked to it is set to "no-value". The tag – value pair is searched in the "Results" bucket as the FirebaseDB2 component is used.



```
call FirebaseDB2 .GetValue
                   tag  " Result "
     valueIfTagNotThere  " no-value "
```

**Figure 3.5.7.** FirebaseDB reading method example

The following image represents the method used to eliminate a tag – value pair from the database, where the tag to remove is set as "Result". The data is removed from the "Results" bucket since the component used is FirebaseDB2. As this bucket just contains one tag – value pair, when this data is eliminated the entire bucket is removed from de database.



```
call FirebaseDB2 .ClearTag
                 tag  " Result "
```

**Figure 3.5.8.** FirebaseDB clearing method example

### 3.5.3. Connection between the Realtime Database and Matlab

Matlab has its own methods in order to establish connection with databases, by using HTTP request methods. This methods are: Post, Get, Put, Patch and Delete. In this project we will use Get for reading and Put for writing.

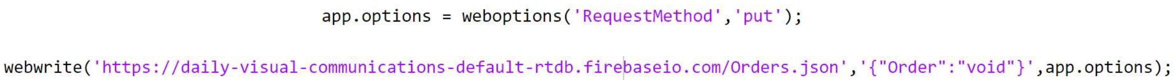
When performing a writing operation, the request method has to be defined, which can be a Post or a Put method. The Put method uploads the written data to a location in the database, overwriting the previous data stored in that location. The Post method expands a location in the database by adding the written data to it.

As there will just be a single order and a single (or none) result at a time, it is efficient to use the Put method and update the database with the new information each time an order or a result is given.

The writing operation of a tag – value pair in a specific bucket (that we will name “bucket”) of a Realtime Database (whose URL we will name https://Firebase.com/) with the Put request method would be done as follows:

webwrite(‘https://Firebase.com/bucket.json’,’{“tag”:”value”}’,weboptions(‘RequestMethod’,’put’))

In the next figure, the first line of code corresponds to the definition of the request method for the write operations. The method is set to Put. The second line corresponds to the writing operation of the tag “Order” and the value “void” to the “Orders” bucket of the Realtime Database of this project.



```
app.options = weboptions('RequestMethod','put');
webwrite('https://daily-visual-communications-default-rtdb.firebaseio.com/Orders.json','{"Order":"void"}',app.options);
```

**Figure 3.5.9.** Matlab writing operation to Firebase example

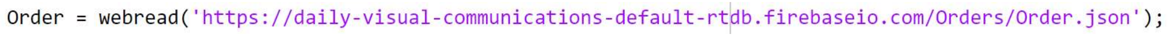
Notice that it is important to have a bucket for the orders and a bucket for the results because if both data were stored in the same bucket, every time a writing operation with the Put method was done, the previous stored order or result would be removed from the database resulting in the erroneous and unsynchronized elimination of data.

When performing a reading operation, the request method is always Get, which enables to ask the database for the value that corresponds to a certain tag.

The reading operation of the value corresponding to a tag (that we will name “tag”) located in a specific bucket (that we will name “bucket”) of a Realtime Database (whose URL we will name https://Firebase.com/) would be done as follows:

webread(‘https://Firebase.com/bucket/tag.json’)

The following figure shows the method used to read the value corresponding to the “Order” tag located in the “Orders” bucket of this project’s Realtime Database. The value is stored in a variable named “Order”.

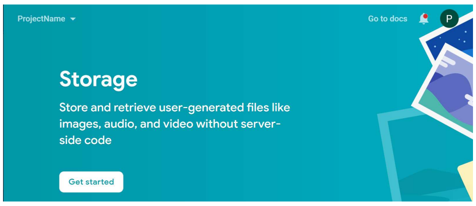


```
Order = webread('https://daily-visual-communications-default-rtdb.firebaseio.com/Orders/Order.json');
```

**Figure 3.5.10.** Matlab reading operation from Firebase example

### 3.5.4. Storage

To create a cloud database for multimedia data, we enter the created project and go to Storage. The next image corresponds to a section of the Storage creation window for the previously created project named “Project Name”. To create the database click on the “Get Started” button.



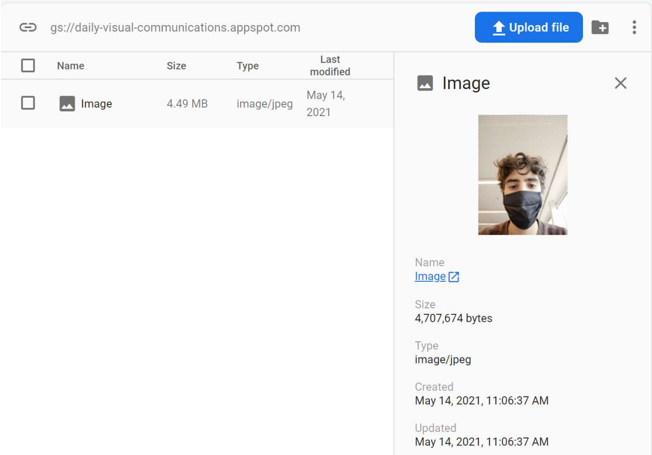
Storage

Store and retrieve user-generated files like images, audio, and video without server-side code

**Figure 3.5.11.** Storage creation windows

Then we accept the security rules set by default as the security protocol of the database, as they allow the connection to any client that has the database URL, its ApiKey and the authorization of the Storage server, without any time limitation period. Finaly, as done with the Realtime Database, we choose the location of the database server. There are many locations for the Storage server. For this project we have set the location as eur3 (europe-west).

The next image shows the workspace of the Storage used in this project. As it can be observed, the connection URL of the database is gs://daily-visual-communications.appspot.com and there is stored an image that was uploaded on the 2021 – 05 – 14, that has a size of 4.48 MB and that is stored in JPEG.



| ☐ | Name | Size | Type | Last modified |
|---|---|---|---|---|
| ☐ | Image | 4.49 MB | image/jpeg | May 14, 2021 |

Name
Image

Size
4,707,674 bytes

Type
image/jpeg

Created
May 14, 2021, 11:06:37 AM

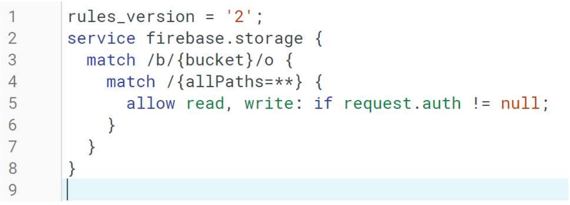
Updated
May 14, 2021, 11:06:37 AM

**Figure 3.5.12.** Storage of the Firebase project that constitutes the bridge between the smartphone and the computer

Once done, we click on “Rules” on the Storage workspace and configure the rules as following:

- We replace the characters “{bucket}” in the third code line by the URL of the database, without the “gs://” characters. This change is done in order to facilitate the configuration of the connection protocols between the Storage platform and Mit App Inventor.
- W change the last line of code, which is “allow read, write: if request.auth != null;”, by “allow read, write;” in order to let any client that has the database URL and its ApiKey stablish connection with the database.

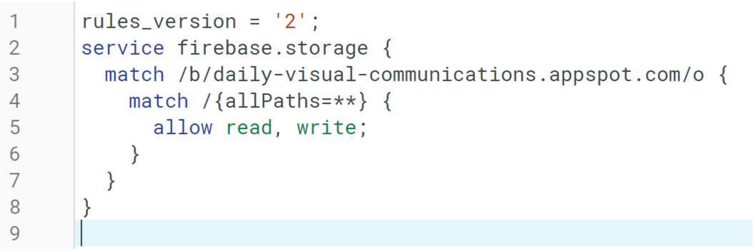
The following image shows the Storage database default rules corresponding to the Firebase session used in this project:



```
1   rules_version = '2';
2   service firebase.storage {
3     match /b/{bucket}/o {
4       match /{allPaths=**} {
5         allow read, write: if request.auth != null;
6       }
7     }
8   }
9   
```

**Figure 3.5.13.** Firebase Storage default rules

The following image shows the Storage database modified rules corresponding to the Firebase session used in this project:



```
rules_version = '2';
service firebase.storage {
  match /b/daily-visual-communications.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

**Figure 3.5.14.** Firebase Storage modified rules

### 3.5.5. Connection between the Storage and Mit App Inventor

Mit App Inventor does not have any component that allows the connection to the Firebase Storage platform. However, this development environment permits the uploading of extensions that consist on components with specific functionalities that are not available on the traditional components.

There is an extension component named FirebaseStorage that allows the communication between the developed app and the Storage database. It can be found on the following web site: https://community.thunkable.com/t/free-firebasestorage-extension-v2-0-new-version/8544 (Bibliography .5).

In order to upload the component to the development environment, we download the .aix file that can be found on the previous web site and import it to the “Extension” repository. The following image shows the “Extension” repository of the Mit App Inventor project used to develop the current application, with the FirebaseStorage component in it.

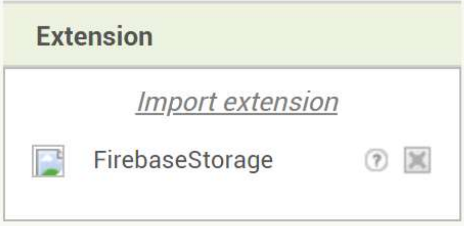
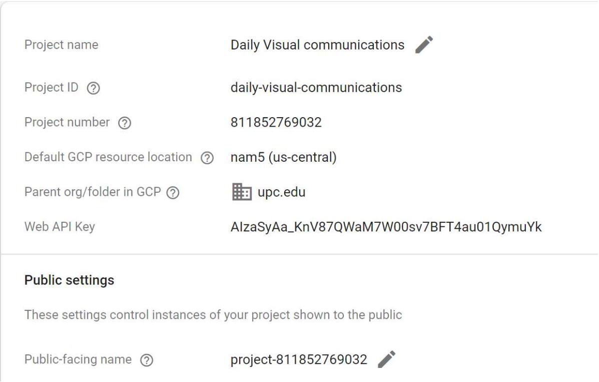


**Figure 3.5.15.** FirebaseStorage component extension

The component has the following properties, which need to be configured:

- ApiKey: It consists on an identification code that is used to enable the client connect to the server, which is the database, by using an API. An API (Application Programming Interface) is an interface that permits and organizes the connections and interactions between software platforms and applications.
  The ApiKey of the Firebase project can be found clicking the configuration button that appears at the top-left corner of the Firebase workspace, going to “Users and permissions” and clicking on “General”.

The image below shows the screen with the general properties of the Firebase project, including the ApiKey.



| | |
|---|---|
| Project name | Daily Visual communications |
| Project ID | daily-visual-communications |
| Project number | 811852769032 |
| Default GCP resource location | nam5 (us-central) |
| Parent org/folder in GCP | upc.edu |
| Web API Key | AIzaSyAa_KnV87QWaM7W00sv7BFT4au01QymuYk |

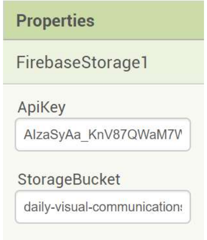
Public settings

These settings control instances of your project shown to the public

| | |
|---|---|
| Public-facing name | project-811852769032 |

**Figure 3.5.16.** General properties of the Firebase project

- StorageBucket: Refers to the name of the Storage database understood as its URL without the "gs://" characters, which in this case would be "daily-visual-communications.appspot.com". The next figure shows the created FirebaseStorage component and its properties.



Properties

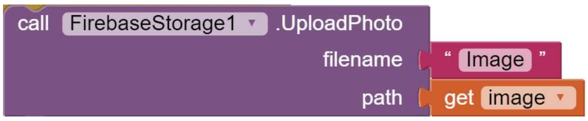
FirebaseStorage1

ApiKey: AIzaSyAa_KnV87QWaM7W

StorageBucket: daily-visual-communication:

**Figure 3.5.17.** Properties of the FirebaseStorage1 component

The next figure shows the method used to upload the image saved in the variable "get – image" to the Storage database that has been connected to the FirebaseStorage1 component. The image is uploaded with the name "Image".



```
call FirebaseStorage1 .UploadPhoto
                 filename  " Image "
                     path  get image
```

**Figure 3.5.18.** FirebaseStorage photo uploading method example

When an image is uploaded to the Storage database, the previous image that may be stored in the same repository is removed. Therefore, as it happens with the Put method, by using this component only one image at a time can be stored in the database. This is useful as we want the software to analyse one single image each time the user wants to carry out an inspection.

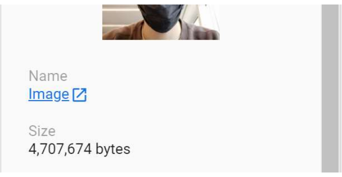
Notice that from the smartphone application, images are only uploaded to the Storage database. The image downloading operation is not carried out by the Mit App Inventor developed app as it just has to storage the images to the database so that they can be downloaded and analysed by the computer.

### 3.5.6. Connection between the Storage and Matlab

The computer program only performs the image downloading operation from the Firebase Storage database, as it just has to read the images uploaded by the smartphone in order to analyse them. In Matlab, the reading operation from a Storage database is done with the same method that is used to read from a Realtime Database. As it happens with the read operation of nominal data, the request method used by Matlab in order to read images from the Firebase is the Get method.

In order to carry out the reading operation it is necessary to know the URL of the repository where the images are stored. As previously mentioned, since there is only one single image stored in the database, knowing the URL of the general repository is enough, without having to know the addresses of the sub-repositories where multiple images would be stored.

In order to get the repository URL, we upload an image to the Storage database and open it by clicking on the "Image" link shown in the next figure:



Name
Image

Size
4,707,674 bytes

**Figure 3.5.19.** Link button that connects to the repository where the image can be downloaded from

Once done, the corresponding image will be opened in the repository where it is located. The repository URL is found on the address bar, and it goes from "firebasestorage" to "alt=media" both included, starting with the characters "https://".

The following image shows the repository of the Storage database used in this project, with a specific image stored in it at the moment of the screen capture. The URL of this repository is "https://firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media".

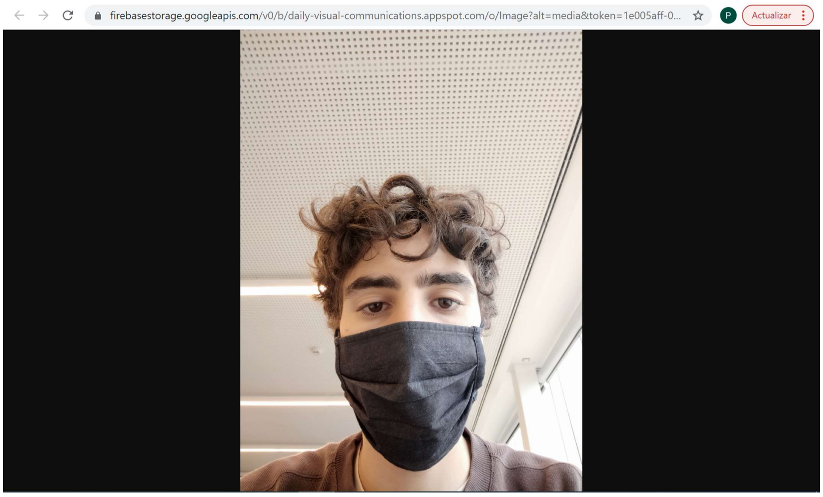
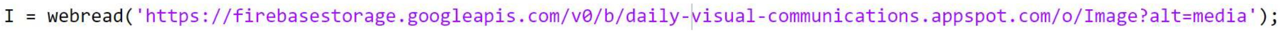

firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media&token=1e005aff-0...
P
Actualizar

**Figure 3.5.20.** Firebase Storage image repository

In Matlab, to perform the reading of an image located in a specific repository use the following method:

webread('URL')

The following figure corresponds to the line of code that reads the Firebase Storage database used in this project, downloads the image stored in it an saves it to a variable named "I".



```
I = webread('https://firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media');
```

**Figure 3.5.21.** Matlab reading operation from Firebase Storage example

# 4. Neural networks design, training and testing

Each time the software does an Iteration, the core work is done by the neural network that analyses the corresponding image. The greatest workload and use of computational resources is carried out from the input of the image into the CNN until it obtains the output information.

Thus, a correct design of the networks' architecture, and a deep and accurate training and testing of them have to be done in order to ensure that the most complex operations the system will have to carry out are done with the greatest reliability, precision, accuracy, efficiency and rapidity.

## 4.1. Introduction to neural networks

### 4.1.1. Definition

Neural Networks are the core tool of Machine and Deep Learning. They try to find solutions to specific problems by using methods that are inspired on the human brain and mind. NNs (Neural Networks) analyse the data that conforms a particular subject of study in order to find patrons that interconnect them, to the objective of reconstructing the complex mathematical structures that explain the relations between the components that define and form the features of the studied problem. Once the Neural Network understands the structures and relations between the data of a problem, it can find solutions to the problem using the same data that has been studied or using new data that follows similar behaviours and patrons than the original.

In order to classify images with a Neural Network, where each image corresponds to an object, into object classes, two processes have to be carried out:

- Training process: Let's say we want to create a Neural Network that classifies three object classes. Then a dataset has to be developed, formed by images that correspond to the studied object classes. The dataset has to be organized so that it is formed by input and output data. The input data are all the images that constitute the dataset, the output data is formed by all the labels that define the class corresponding to the object type of each image.
  When training the NN, it will carry out computational and algorithmic operations in order to find the relations between the graphic features and shape patterns of the studied images and their corresponding object class.
  The larger the dataset used to train the network is, the more accurate and precise the found patrons and relations will be.

- Testing process: Once the training is done, when a new image is inserted into the Neural Network, it studies the shapes and features of it and relates them with the previous found patters that relate graphic features of images which its corresponding class, being then able to determine the object class that corresponds to the image.
  A testing set of images is analysed by the network to ensure that it solves the classification problem correctly and accurately. If it doesn't, changes on the training or the design of the network have to be done in order to make it capable of recognizing the features and characteristics that correspond to each class so it is able to carry out the classification with precision.

### 4.1.2. Architecture of a neural network

In order to find the patterns that relate the data with their solution (which might be a class, a number corresponding to a regression, etc.) during the training process the network moulds itself in order to act like a function where, when the input data is introduced to it, it is mathematically transformed into the solution. NNs are formed by internal layers and each layer is made up by cells that transform the given inputs into outputs, where the initial inputs of the NN is the data and the final output is the solution.

These layers can be defined as neurons, as they are inspired by the human nerve cells, and form structures inspired by the human brain structures. Each neuron transforms single or multiple inputs into a single or multiple output. Therefore, each neuron can be understood as a mathematical function. In Neural Networks, neurons are named perceptrons. The following image represents the architecture of a single perceptron.

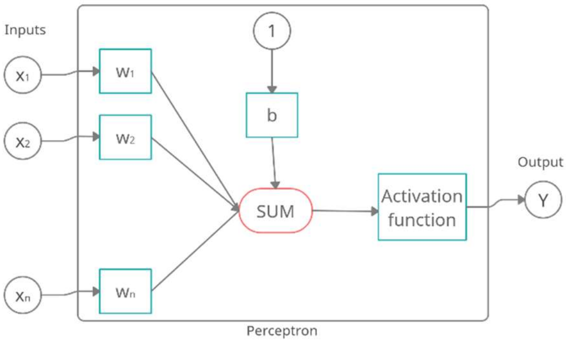


**Figure 4.1.1.** Architecture of a perceptron

- Elements from $X_1$ to $X_n$ correspond to the input numbers that are introduced into the perceptron.
- Each of these inputs is multiplied by a certain weight, which goes from W1 to Wn.

- Then they are added together and summed with a bias, which is represented as “b” in the figure above.
- The result of the summation is introduced to an activation function that transforms it into the final output.
- Y corresponds to the output given by the perceptron.

If c corresponds to the number introduced into the activation function:

$$c = b + \sum_{i=1}^{n} w_i + x_i \quad \text{(Eq. 4.1)}$$

There are lots of activation function models. One of the most popular ones are:

Being c the input of the activation function and Y its output.

- Linear activation function:

$$Y = c \quad \text{(Eq. 4.2)}$$

- Rectified linear activation function or ReLU (most common activation function in deep learning):

$$Y = \begin{cases} 0 \; if \; c \; < 0 \\ c \; if \; c \; \geq 0 \end{cases} \qquad \text{(Eq. 4.3)}$$

- Sigmoid activation function (useful in binary classification problems):

$$Y = \frac{1}{(1+e^{-c})} \qquad \text{(Eq. 4.4)}$$

- Hyperbolic tangent activation function (common in language processing and speech recognition):

$$Y = \tanh(c) \qquad \text{(Eq. 4.5)}$$

- Softplus activation function (common in speech recognition):

$$Y = \log(1 + \; e^{-c}) \qquad \text{(Eq. 4.6)}$$

Neural Networks model the weights and bias of each neuron during the training, so that they together end up transforming the given data into the expected result. During the training process, multiple iterations are done. In each iteration, the network modifies the mathematical function of its perceptrons and verifies whether the created combination of functions correctly solves the problem, comparing the obtained result with the expected result given by the output data of the training dataset. If a specific combination of functions correctly solves the problem, this combination is reinforced and taken into account in the next iterations in order to end up building a model that minimizes the error of the solution given by the network. As more iterations are done, the network tends to find a better model.

If a neural network is composed by a single layer of perceptrons, it is called a Feed Forward Neural Network (FF). If a NN is formed by multiple layers of perceptrons, where the outputs of a layer are the inputs of the next layer, it is called a Deep Feed Forward Neural Network (DFF). The next figure presents a simplified diagram for each architecture.

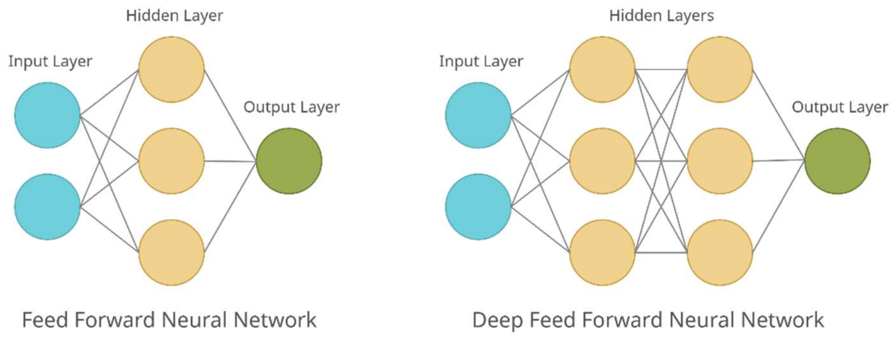


**Figure 4.1.2.** FF and DFF architecture

When neural networks with multiple hidden layers are used in order to solve problems, we talk about Deep Learning.

The structures that form the layers of a NN and the organization of the interconnections between their perceptrons form the specific architecture of the Neural Network. The structure of a network directly affects the way it learns from data and its usefulness and effectiveness in order to solve some specific types of problems. Moreover, the architecture of a neural network defines the algorithm and the mathematical model that the NN will use in order to train itself and get conclusions from the data that describes a given problem.

In a NN not all layers may be formed by the same types of perceptrons, as there exist multiple classes of perceptrons that although having the same objective as the common perceptron, which is computing mathematical and algorithmic operations in order to transform an input into an output, are formed by completely different mathematical models and structures. Therefore, the processing each type of perceptron does to input data is completely different and drives to specific outputs with particular utilities. The next image presents simplified diagrams of some of the most common network architectures in deep learning, showing the structures created by the layers and the types of perceptrons that form them. Later on, we will study the NN architectures used in this project.

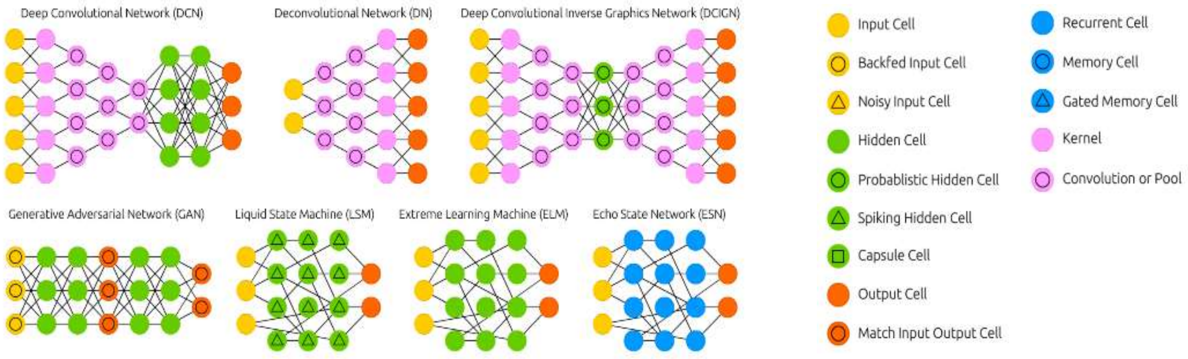


**Figure 4.1.3.** NN architectures (Source: The Asimov Institute, “The neural network zoo” _ Bibliography .12)

During the training phase, a percentage of the perceptrons of each layer are disabled, which means that their input and output connections are cut off, so that they do not operate. In each iteration, the perceptrons are disconnected randomly based on the probability (in percentage) that each of them is disabled. This desactivation of perceptrons is carried out by layers that are placed just after the layers whose perceptrons are wanted to be randomly disabled. This layers are called dropout layers.

## 4.2. Pretrained Deep Neural Networks

### 4.2.1. Object classification

ImageNet is a free-use database that contains more than 14 million images divided in more than 20000 classes.

ImageNet website: https://www.image-net.org/ (Bibliography .3)

Matlab has a large set of neural networks that have been trained with a subset of ImageNet that contains more than a million images corresponding to 1000 different classes (belonging to various objects, animals, food, etc.). These networks are then, able to classify images in these 1000 different classes.

The following image shows some of the various architectures that define the neural networks which Matlab offers for its use, that have been trained with the ImageNet dataset.

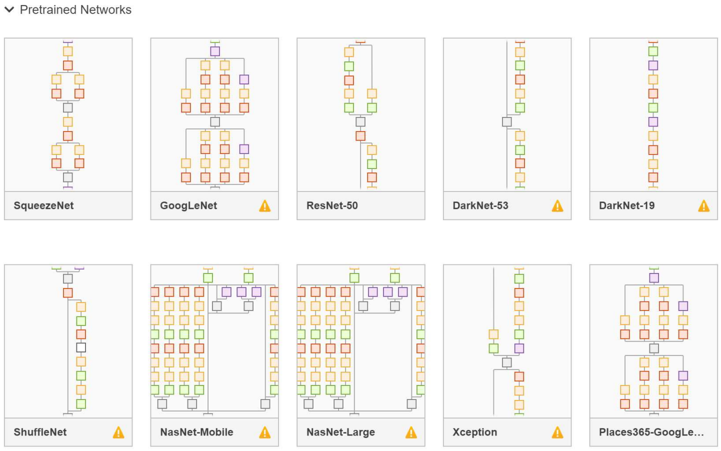
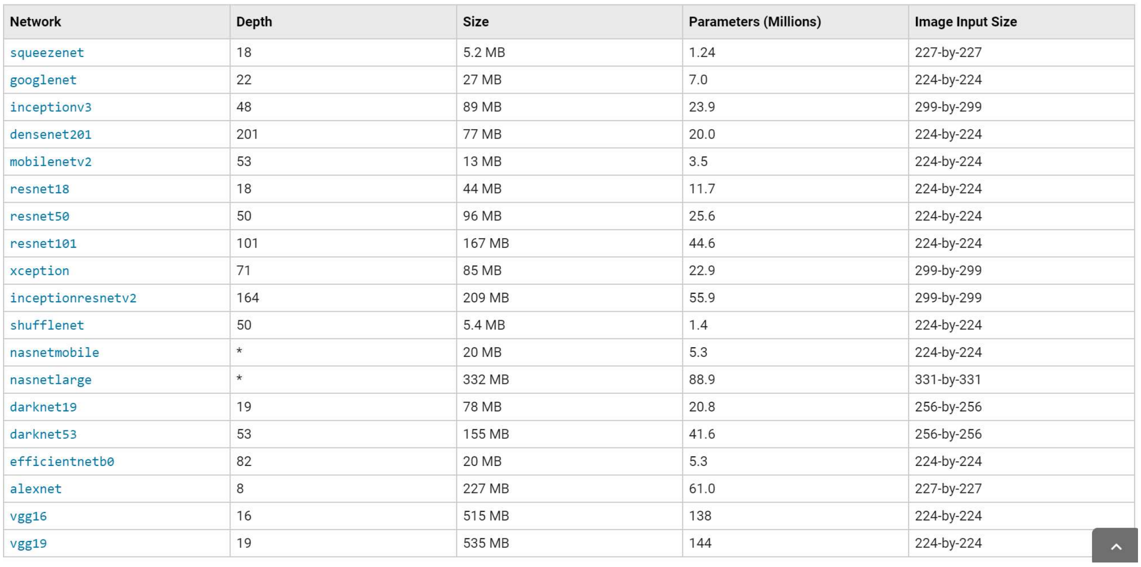


**Figure 4.2.1.** Architecture diagrams of some of the CNNs trained with the ImageNet dataset (Source: Pau Besses Casas, "Matlab, Deep Network Designer")

The next figure represents some of the characteristics of the neural networks that have been trained with the specific subset of the ImageNet dataset. The depth of the network represents the number of fully connected layers that form the path from the input layer to the output layer. The image Input Size is measured in pixels.



| Network | Depth | Size | Parameters (Millions) | Image Input Size |
|---|---|---|---|---|
| squeezenet | 18 | 5.2 MB | 1.24 | 227-by-227 |
| googlenet | 22 | 27 MB | 7.0 | 224-by-224 |
| inceptionv3 | 48 | 89 MB | 23.9 | 299-by-299 |
| densenet201 | 201 | 77 MB | 20.0 | 224-by-224 |
| mobilenetv2 | 53 | 13 MB | 3.5 | 224-by-224 |
| resnet18 | 18 | 44 MB | 11.7 | 224-by-224 |
| resnet50 | 50 | 96 MB | 25.6 | 224-by-224 |
| resnet101 | 101 | 167 MB | 44.6 | 224-by-224 |
| xception | 71 | 85 MB | 22.9 | 299-by-299 |
| inceptionresnetv2 | 164 | 209 MB | 55.9 | 299-by-299 |
| shufflenet | 50 | 5.4 MB | 1.4 | 224-by-224 |
| nasnetmobile | * | 20 MB | 5.3 | 224-by-224 |
| nasnetlarge | * | 332 MB | 88.9 | 331-by-331 |
| darknet19 | 19 | 78 MB | 20.8 | 256-by-256 |
| darknet53 | 53 | 155 MB | 41.6 | 256-by-256 |
| efficientnetb0 | 82 | 20 MB | 5.3 | 224-by-224 |
| alexnet | 8 | 227 MB | 61.0 | 227-by-227 |
| vgg16 | 16 | 515 MB | 138 | 224-by-224 |
| vgg19 | 19 | 535 MB | 144 | 224-by-224 |

**Figure 4.2.2.** Features of the CNNs trained with the ImageNet dataset (Source: Matlab, "Pretrained Deep Neural Networks" __ Bibliography .10)

When in Matlab an image is inserted into one of these pretrained networks, the CNN gives the labels that define the most probable classes that could correspond to the actual classes of the objects that appear in the image and the probabilities for each label of actually corresponding to the real class. If more than one object is contained on the image, the network output label with the highest probability defines the class of the object that represents a biggest area on the picture and has a greatest presence and clarity.

The following images have been analysed by the ResNet-50 pretrained network, capable of recognizing objects belonging to 1000 different classes (as the other pretrained NNs do). Both images contain a coffee mug, a computer and a desk. Nevertheless, the network does two completely classifications based on which are the objects that occupy a larger area and have a higher presence and sharpness.

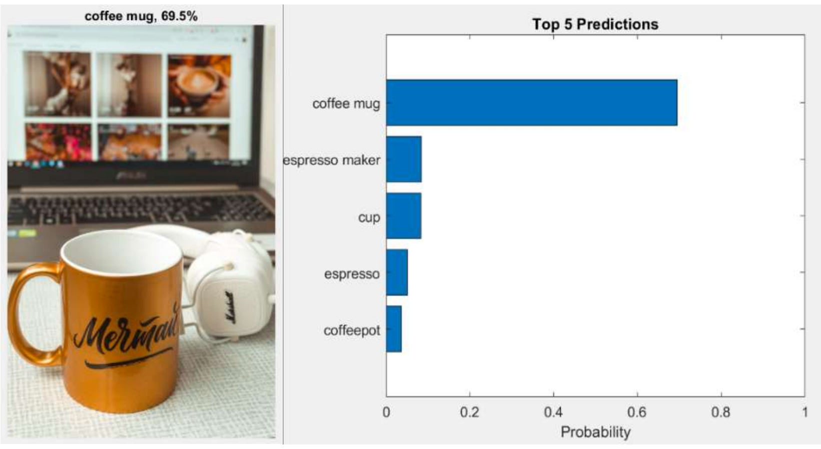


**Figure 4.2.3.** Image analysis by the ResNet-50

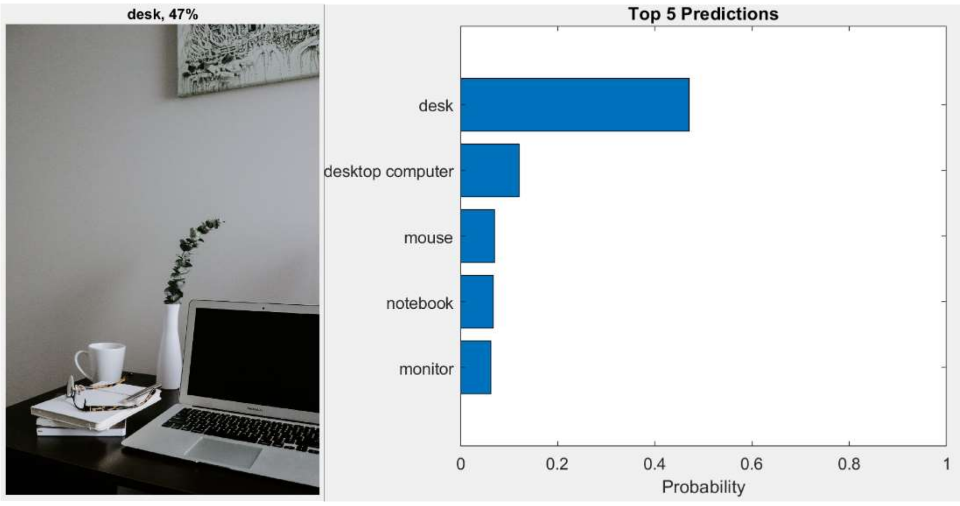


**Figure 4.2.4.** Image analysis by the ResNet-50

As these neural networks are capable of classifying into 1000 different object classes and recognize multiple objects in an image, they are optimal for giving a general description of the environment the user wants to examine. Moreover, as they are able to recognize multiple objects in an image and calculate the probability for each detected object of having been correctly classified, they constitute ideal tools for the general analysis of the objects that are placed in a certain space.

### 4.2.2. Convolutional Neural Networks

All the networks that Matlab provides in order to classify images between the 1000 classes from the ImageNet dataset that have been used for the training, are convolutional neural networks.

CNNs (or Convolutional Neural Networks) are NNs designed for graphical analysis such as object detection and recognition. Their architecture is modelled for this specific purpose: CNNs are made up of convolutional layers, which act as filters that extract features from objects.

On the first convolutional layers, filters extract specific features from the image and reconstruct it with these extracted features (these figures can be, for example, lines, arches and corners). As deeper the layers are, they extract more complex figures from the image, which has already passed through previous layers of convolution (which can go from object contours to complex shapes based on profundity, colour and shading features). These layers transform the original image in an image formed by the basic shapes that have a greater importance in its definition.

Moreover, between the convolutional layers there are Max Pooling layers, which reduce the dimensions of the image while preserving the maximum amount of information. This way, the original image ends up being a single pixel or a reduced set of pixels that form a particular shape.

Generally each pixel has a value that corresponds to its brightness in grey-scale images or has three values that correspond to the brightness of each of its RGB (red green blue) parameters in colour images. ReLu activation functions are placed between convolutional layers in order to replace all negative pixel values by zeros, which facilitates the use of the convolution filters.

The network analyses the final set of pixels that the image has been transformed to when passing through the convolutional and max pooling layers. Particular structures that this reduced set of pixels may form are directly related to the different classes that can correspond to the original image.

For example, if a network is trained to classify between dogs and cars and its final layer is formed by four pixels, a high value of pixels 1 and 3 may correspond to an original dog image whereas a high value of pixels 2 and 4 might correspond to an original car image (for example).

The next image shows the output that each perceptron of each layer computes from a given image, and its final classification.

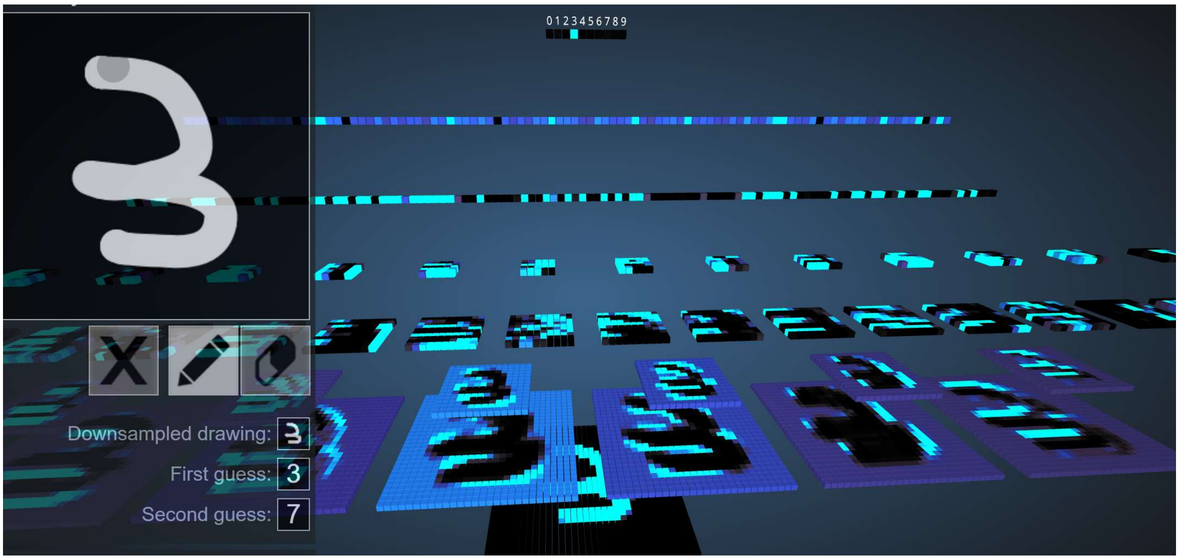


**Figure 4.2.5.** Visualization of the graphic classification process of a CNN

Web site used for this simulation: https://www.cs.ryerson.ca/~aharley/vis/conv/

The layers that can be observed in the figure shown above have the following functions:

- Input layer: The original image is inserted into the network.
- First hidden layer: Convolutional layer that remarks basic shapes in the original image.
- Second hidden layer: Max Pooling layer that reduces the dimensions of the images generated in the first hidden layer.
- Third layer: Convolutional layer that defines complex shapes corresponding to specific combinations of the images generated in the second hidden layer.
- Fourth layer: Max Pooling layer that reduces the dimensions of the images generated in the third hidden layer.
- Fifth layer: Fully connected layer that transforms all the images generated in the fourth layer into a single array of pixels. Each pixel of the generated array is created by multiplying the pixels of the previous layer by a specific combination of weights.
- Sixth layer: Fully connected layer that computes a more specific array of pixels whose value directly corresponds to the class of the original image.
- Output layer: The combination of values that correspond to the pixel array created in the sixth layer is directly related to a class. Thus, classifying the image.

This network has been trained to classify numbers from 0 to 9.

### 4.2.3. ResNet - 50

We will use the ResNet-50 architecture for the detection of the objects that appear in an image as it is a precise, fast and scalable network. ResNet-50, which is the abbreviation for Residual Network 50 is a type of convolutional neural network.

- Architecture:

ResNet – 50 does not use the output of a layer as the input of the next layer (as basic CNNs do), but the input of a layer is the result of the summation of the output of the previous layer and the input of the previous layer.

The next diagram illustrates a residual block, which is the group of layers that perform the complete previously described mathematical process in a Residual Network.

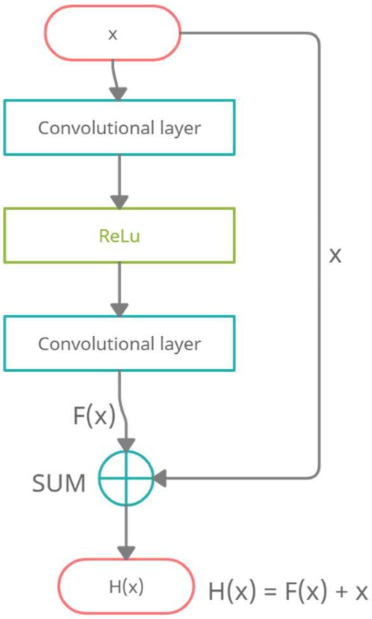


**Figure 4.2.6.** Residual block of a Residual Neural Network

Conventional CNNs consist in a simple succession of convolutional, ReLU and Max Pooling layers, while Residual CNNs compute this addition process for every residual block. This addition process gives stability to the network, which is important in order to achieve scalability.

About scalability:

Conventional CNNs improve their accuracy and efficiency as more layers are used to form their internal structure. However, this principle works until a limit of layers is reached. When this limit is reached, as more layers are introduced to the network, less is the accuracy and efficiency of its performance. Moreover, if a sufficiently large number of layers are introduced into a conventional CNN, its accuracy and reliability drops devastatingly.

On the other hand, Residual CNNs always improve their performance as more layers are used to form them. Therefore, by forming larger successions of residual blocks the network reliability increases. This is due to the mathematical stability and robustness that the residual block presents in comparison with the conventional simple convolutional layers. This makes Residual CNNs scalable, which is a crucial characteristic in deep learning models.

The next figure corresponds to the simplified diagram of a basic CNN (at the left) and a Residual CNN (at the right).

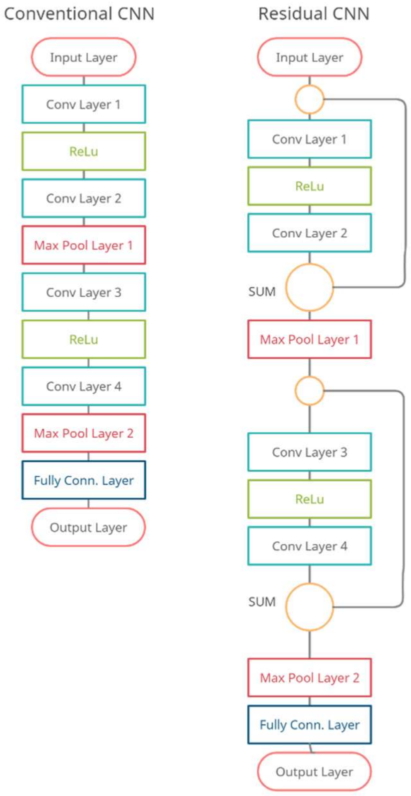


**Figure 4.2.7.** Basic CNN and Residual CNN architectures

ResNet – 50 is a Residual Neural Network formed by 50 hidden layers.

### 4.2.4. Place classification

In addition to recognizing the main objects that appear in an image in order to make a general description of the environment the user wants to inspect, the developed software will also classify the inspected image into the place that it represents. For example, if the user takes a photo of a bedroom, the software will recognize the place corresponding to the image as a bedroom and will communicate this result to the user.

Matlab provides a neural network that has been trained to classify images between 365 different places. This NN model is called Place365GoogLeNet and has been trained with the Places365 dataset, which contains 1.8 million training images divided into 365 space categories.

Places365 website: http://places2.csail.mit.edu/download.html (Bibliography .4)

The following image has been classified using the Place365GoogLeNet network.

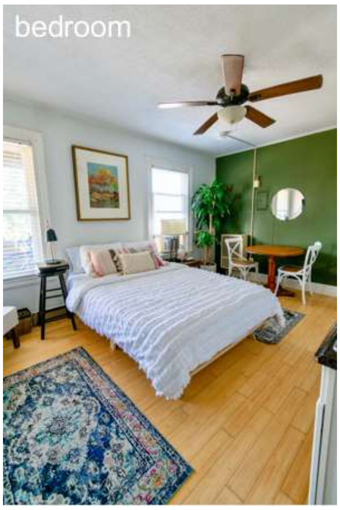


**Figure 4.2.8.** Image analysis by the Place365GoogLeNet

### 4.2.5. GoogleNet

Place365GoogLeNet is the trained version of the neural network GoogLeNet on the Places365 dataset.

GoogLeNet is a model formed by 27 hidden layers which are grouped, in turn, into 9 inception blocks. An inception layer is a convolutional layer that does not operate with a single convolution filter but with multiple (which are usually three) filters with different size, and that also contains a Max Pooling block. The outputs given by each convolution filter and the Max Pooling are added together, forming the input of the next layer.

The next image shows a simplified diagram of an inception block, whose architecture corresponds to the one used in the GoogLeNet neural nework.

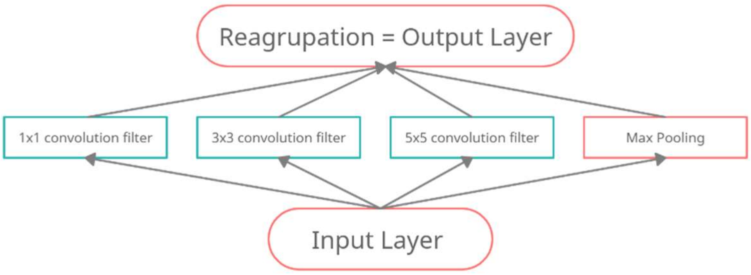


**Figure 4.2.9.** Architecture of an inception block

Basically, as it can be observed in the figure above, rather than stack convolutional layers sequentially, multiple convolution operations are executed on the same layer. This results in an optimization of computational resources.

The next diagram corresponds to the structure of the final layers of the GoogLeNet model.

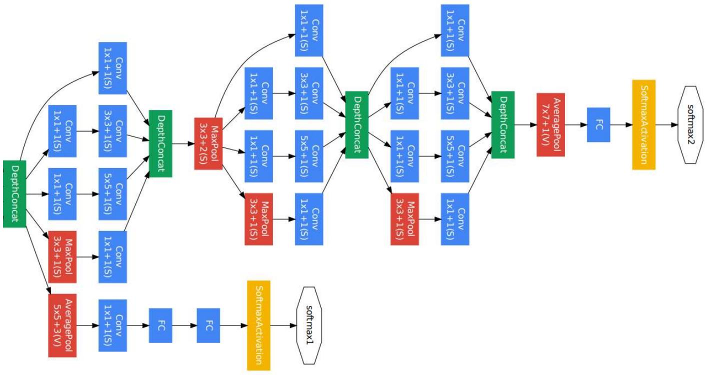
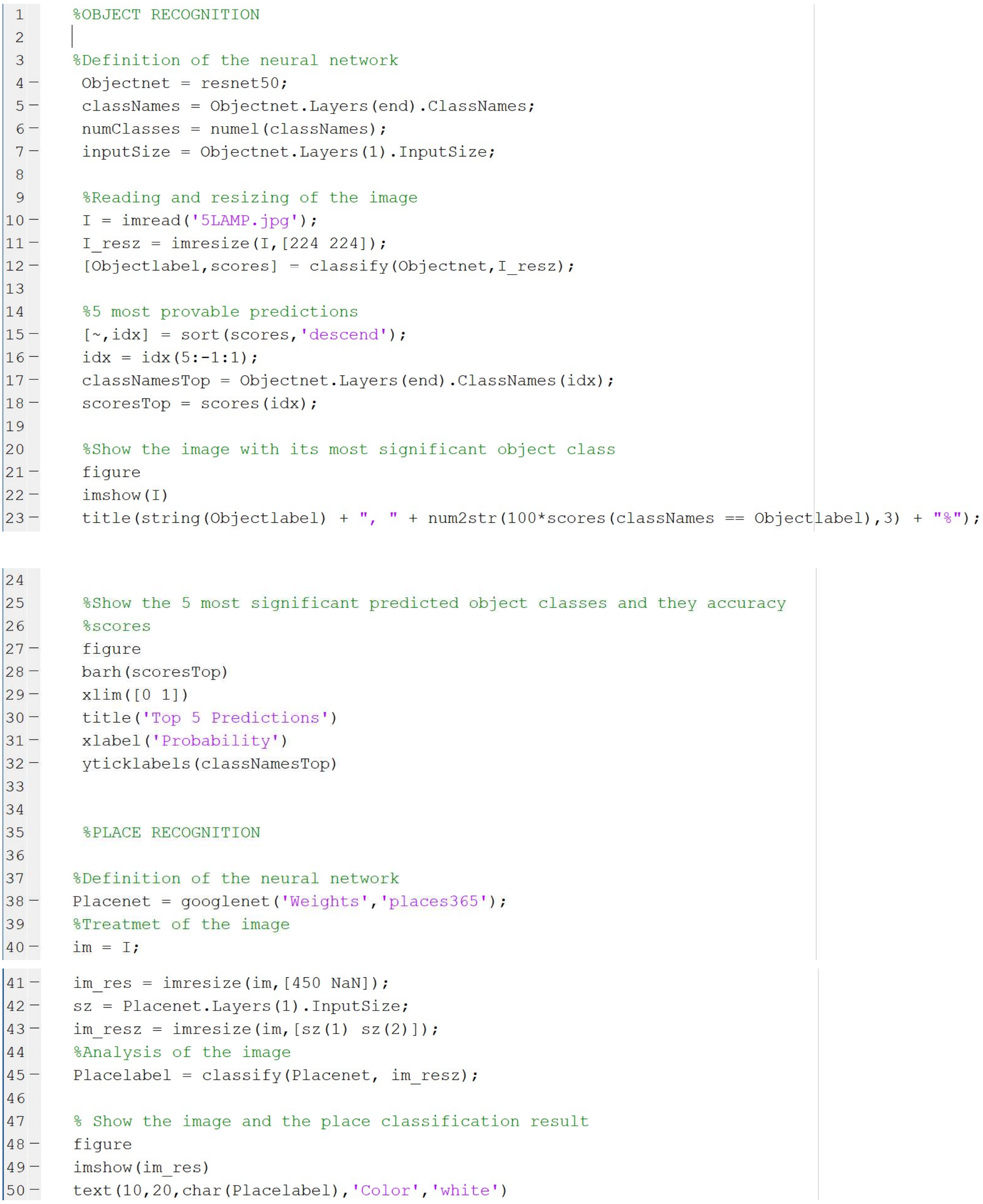


**Figure 4.2.10.** Last layers of the GoogLeNet network (Source: “Going deeper with convolutions” _ Bibliography .11)

## 4.2.6. Matlab code for general image analysis

The next figure corresponds to the code that implements the general analysis of an image, computing the detection of the main objects in it and the place class that best describes the environment of the picture. Each significant part of the code is commented in its beginning.



```
%OBJECT RECOGNITION

%Definition of the neural network
 Objectnet = resnet50;
 classNames = Objectnet.Layers(end).ClassNames;
 numClasses = numel(classNames);
 inputSize = Objectnet.Layers(1).InputSize;

 %Reading and resizing of the image
 I = imread('5LAMP.jpg');
 I_resz = imresize(I,[224 224]);
 [Objectlabel,scores] = classify(Objectnet,I_resz);

 %5 most provable predictions
 [~,idx] = sort(scores,'descend');
 idx = idx(5:-1:1);
 classNamesTop = Objectnet.Layers(end).ClassNames(idx);
 scoresTop = scores(idx);

 %Show the image with its most significant object class
 figure
 imshow(I)
 title(string(Objectlabel) + ", " + num2str(100*scores(classNames == Objectlabel),3) + "%");


 %Show the 5 most significant predicted object classes and they accuracy
 %scores
 figure
 barh(scoresTop)
 xlim([0 1])
 title('Top 5 Predictions')
 xlabel('Probability')
 yticklabels(classNamesTop)


 %PLACE RECOGNITION

%Definition of the neural network
Placenet = googlenet('Weights','places365');
%Treatmet of the image
im = I;
im_res = imresize(im,[450 NaN]);
sz = Placenet.Layers(1).InputSize;
im_resz = imresize(im,[sz(1) sz(2)]);
%Analysis of the image
Placelabel = classify(Placenet, im_resz);

% Show the image and the place classification result
figure
imshow(im_res)
text(10,20,char(Placelabel),'Color','white')
```

**Figure 4.2.11.** Commented code for the general analysis and description of images

## 4.3. RCNN and transfer learning

The pretrained networks used for the general description of the images present three main problems for the recognition of specific objects:

- The classification cannot have a null result. In other words, when analysing images it does not exist the possibility of not having found any of the objects that correspond to the classes used for the network training. The NNs trained with the ImageNet dataset that Matlab provides will always classify an image into the object classes that may correspond to it, although the image does not correspond to any of the object classes.
- They are not able to find bounding boxes. They are capable of detecting objects in an image but cannot compute the coordinates on the image that correspond to the location of the object.
- They have been trained in order to classify between 1000 object types, but it is not possible to add new classes to the neural network or modifying any of them.

Therefore, in order to be able of searching specific objects in an image and detect their location, RCNN networks will have to be trained.

### 4.3.1. Region Based Convolutional Neural Networks

RCNN (Region Based Convolutional Neural Networks) are convolutional neural networks that analyse images where regions of interest have been previously computed. Therefore the CNN directly analyses the regions of interest instead of analysing the full image.

The regions of interest are computed using what is called a Feature Map, which is compound by multiple layers that make groups of pixels based on their similarities. The first layers of the Feature Map generate groups of pixels that are located next to each other and that have specific similar characteristics. Firstly these groups are made out of small amount of pixels. As the Feature Map goes deeper they are grouped into larger sets that end up segmenting the image into the objects that may form it.

Therefore, a RCNN segments the analysed image into the objects that form it that may correspond to the classes that have been used for the network training. Then, the segments that correspond to specific classes are inserted into two neural models:

- A conventional CNN that classifies each segment into an object class.
- A Bounding Box Regressor network that computes the coordinates that correspond to the location of each segment that corresponds to a specific class.

Thus, objects in images are not only recognized but are also located. Moreover, if no valid segmentations can be done in the Feature Map, it means that no objects corresponding to the classes which the network has been trained to detect are located in the image, and the RCNN outputs a null result.

Faster RCNN can be understood as an improved version of RCNN. Faster RCNN passes the input image through a first set of convolutional layers that extract the most basic and important features from it. Then the output that generates the convolutional layers is regrouped into a single image, introduced into the Feature Map and subsequently introduced into the CNN classifier and the Bounding Box Regressor. This makes Faster RCNNs more efficient and accurate as the segmentation of the image done in the Feature Map is done on an image where important shapes and features are remarked.

We will use RCNNs in order to solve the specific object recognition and location problem in the environments the user wants to analyse.

### 4.3.2. Transfer learning

Training a RCNN takes really large datasets in order to achieve high rates in accuracy and reliability of the trained network.

Transfer learning method is used to transform networks previously trained to detect specific objects into networks that detect new object types. In order to do this transformation, the original network has to be trained with a large dataset formed by a big number of samples. On the other hand, the dataset needed to compute the network transformation can be relatively small.

Transfer learning works by taking advantage of the information learned by the initial and middle layers of the original network. The initial and middle layers of the original layer extract and detect certain features from the images related to the original dataset. These features are basic features that tend to appear both in the images related to the original dataset and the images related to the new dataset.

Therefore, the initial and middle convolutional layers of the original network can be used for the extracting of accurate and relevant information from the new objects we want the network to detect, without having to retrain them. This process results in a great saving of time and resources. Making transfer learning more efficient and powerful (if it is carried out correctly) than the original scratch learning.

The only layers that are modified from the original neural network are the final convolutional layers that extract more complex features, which are more related to the real images and that are specific for each object class.

For example, both a horse detection RCNN and a plane detection RCNN extract almost the same features in the first convolutional layers and similar shapes in the middle ones (which may be edges, circles, arched forms, etc.). However, the final layers extract features that are more deeply related to the class itself (as may be legs and tails for the horse detection network and wings and windows for the plane detection network). Therefore, by doing some appropriate changes in the last convolutional layers of the horse detection network, using a small dataset of plane images, this network can be transformed into a plane detection network.

In order to generate networks capable of detecting and locating specific objects we will compute a transfer learning from an original pretrained network. For doing this, the CNN will have to be designed and trained with the original large dataset. Subsequently, the transfer learning will be computed using a small dataset that refers to the new object classes.

The following is the process that we will perform from the creation and training of the original CNN to the transfer learning process in order to create networks that detect specific object classes.

We will firstly download the CIFAR10 dataset, which is the image dataset that will be used to train the original neural network. CIFAR10 contains 50.000 images divided into 10 object classes.

Then the original CNN is designed and developed. It will be formed by:

- An input layer for the introduction of the images to the network.

- Three middle layers, each of them formed by:
  A convolutional layer with 32 filters of 5x5 pixels.
  A ReLu activation function layer
  A max pooling layer

- A final layer formed by:
  A fully connected layer
  A ReLu activation function layer
  A fully connected layer
  A SoftMax activation function layer
  A final classification layer that outputs the result of the object detection

Once developed, the CNN is trained with the CIFAR10 dataset. Which takes a large amount of time.

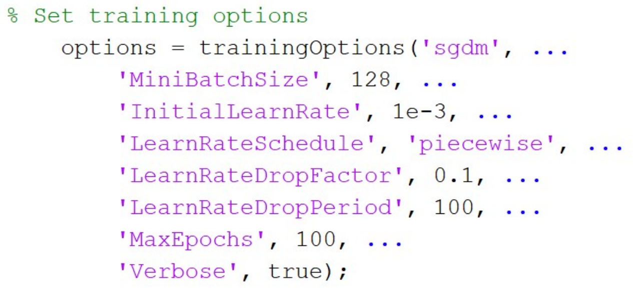
When the training process finalizes, the developed and trained CNN with the CIFAR10 dataset is tested and an accuracy value that defines the evaluation of its performance is computed. Finally, the CNN is saved.

The full commented code used for the development and training of the CNN can be observed in the *Annexes A1*.

Now each time we want to train a network to detect and located a specific object class or classes we load the saved original CNN as cifar10Net and use the Matlab function for the learning transfer of CNNs. This function is called trainRCNNObjectDetector and has the following parameters:

- New dataset that will be used to carry out the learning transfer.
- Original pretrained CNN
- Options and configuration parameters

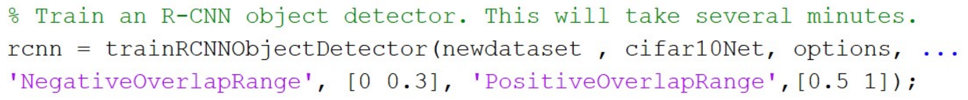
The next figure shows the lines of code corresponding to the definition of the option parameters that have to be described in the transfer learning process function.



```
% Set training options
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 100, ...
        'Verbose', true);
```

**4.3.1.** Training options of the learn transfer process

The next figure corresponds to the function that computes the transfer learn, setting the new dataset as the first parameter, the original network as the second, and the options and configuration features as the following parameters.



```
% Train an R-CNN object detector. This will take several minutes.
rcnn = trainRCNNObjectDetector(newdataset , cifar10Net, options, ...
'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1]);
```

**4.3.2.** Learn transfer function

This function generates a RCNN trained to detect the new object classes.

It is important to take into account that the new dataset can be really small compared to the original one. In fact, a transfer learning where the new dataset has similar dimensions or is bigger than the original one does not have any sense.

A new dataset between 100 and 150 samples is enough for an accurate transfer learning of the object classes we want to detect in this project.

The initial investment in training the original network with the CIFAR10 dataset is compensated by the possibility of developing RCNNs for the detection of specific objects in a really fast, efficient and simple way.

# 5. Implementation of the RCNN

Once the RCNN architecture is designed, we train it in order to make it capable of recognizing and locating specific object classes.

We firstly train the network for the detection of a specific object type. Then we test it and evaluate its performance and, if the model works, we generalize it and develop a more complex neural network in order to detect multiple objects.

The network will be initially trained to detect faces.

## 5.1. Training and testing the network for face detection and location

### 5.1.1. Creation of the training dataset

The neural network model is already designed. Therefore, a dataset has to be created and introduced to the RCNN training process as the training data. This dataset will be developed for the specific detection and location of faces.

The dataset introduced to the RCNN training process has to be a table that contains samples in its rows and attributes in its columns. The attributes are the following:

- A column with the image file. Actually, the information of this column corresponds to the directory where each image file is located in the computer.
- A single column with the coordinates of the bounding box that contains the face or with the coordinates multiple bounding boxes if various faces are located in the image.

Samples are each image – bounding box (or bounding boxes) pair.

The following table represents a simplified example of a face training dataset with three samples.

| Image Filename | Faces BBoxes |
|---|---|
| C:\Users\Image1.jpg | [200 312 40 37] |
| C:\Users\Image2.jpg | [] |
| C:\Users\Image3.png | [9 413 112 21] [108 100 76 14] |

**Figure 5.1.1.** Representation of a training dataset

In the dataset represented above, the first sample contains a single bounding box, the second does not contain any bonding box (i.e. there is not any face in the image), and the third contains two bounding boxes (i.e. two faces). The first bounding box is a rectangle that has its top left corner in the coordinates (200, 312) of the image and is 40 pixels high and 37 pixels wide. All bounding boxes used in this project are rectangular. Bounding boxes are also called ROIs (Regions of Interest).

Matlab has an integrated tool for the extraction of ROIs in images and the generation of training datasets as image – ROI tables. This image is called Image Labeller and can be opened from the Matlab workspace using the command “imageLabeler()”.

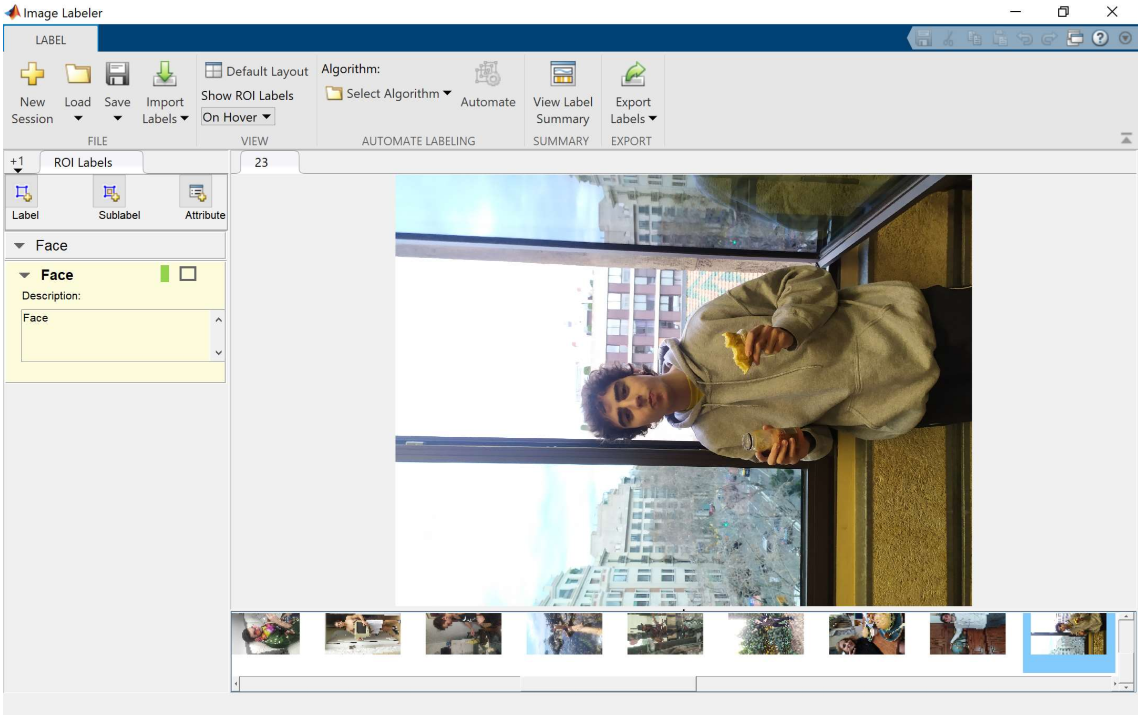


**Figure 5.1.2.** Image Labeler workspace

In the shown Image Labeler workspace a set of images has been loaded by clicking on “Load” and subsequently clicking on “Add images from folder” and selecting all the images of interest.

Once the workspace and the images are loaded, create a label. We have named the label “Face” as we are documenting faces on images. Now, using the mouse, all those regions of each image corresponding to faces are selected.

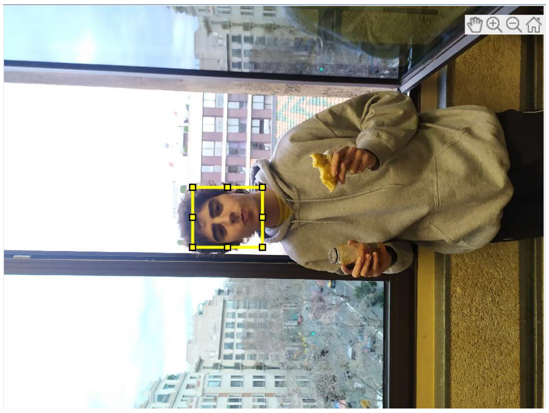
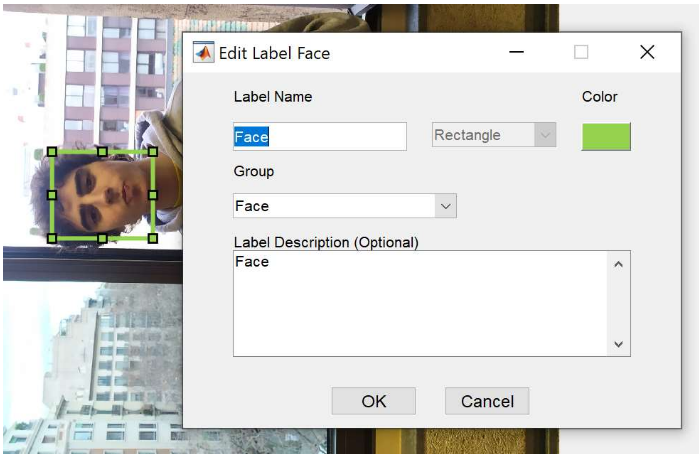


**Figure 5.1.3.** Generation of a bounding box for the labelling of the face located in the shown image

The generated ROI is edited to make it correspond to the "Face" label. Once a label is created and assigned to a ROI, the next generated ROIs are automatically assigned to this label.



Edit Label Face

Label Name: Face

Rectangle

Color: 🟩

Group: Face

Label Description (Optional): Face

**Figure 5.1.4.** Assignation of a label to a bounding box

The figure shown below illustrates the extraction of multiple ROIs in a single image.

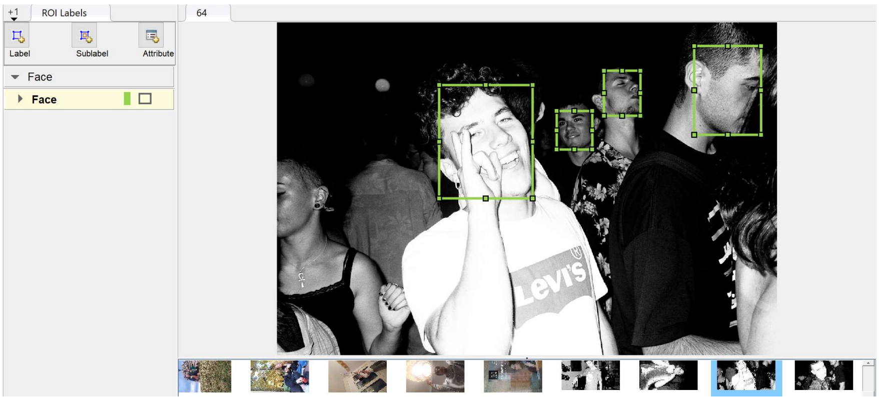
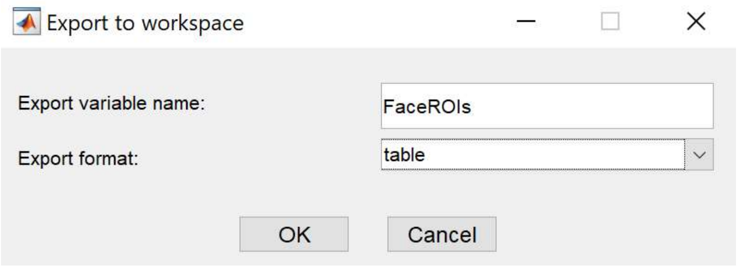


**Figure 5.1.5.** Extraction of multiple ROIs in an image

Once all images are documented with labelled ROIs, we export the dataset to the Matlab workspace as a table. We have named the table “FaceROIs”.

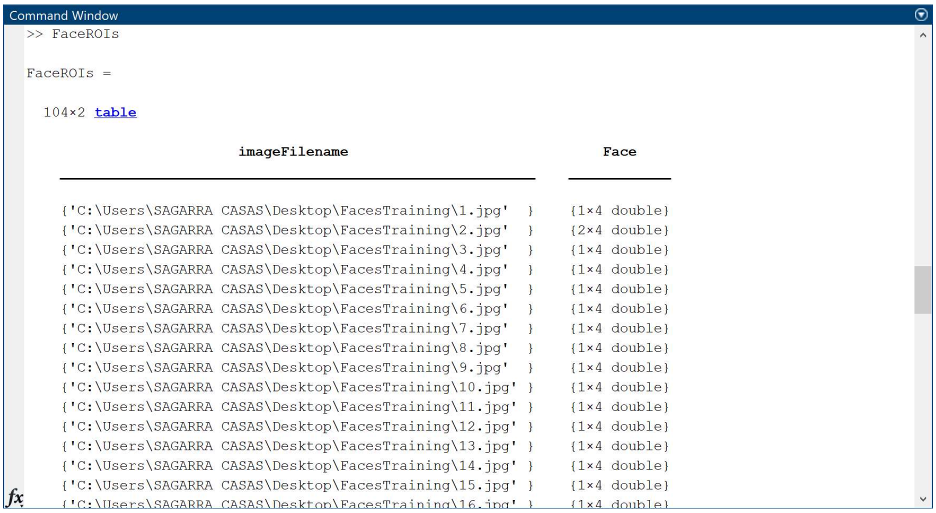


Export variable name: FaceROIs

Export format: table

**Figure 5.1.6.** Exportation of the dataset to the workspace

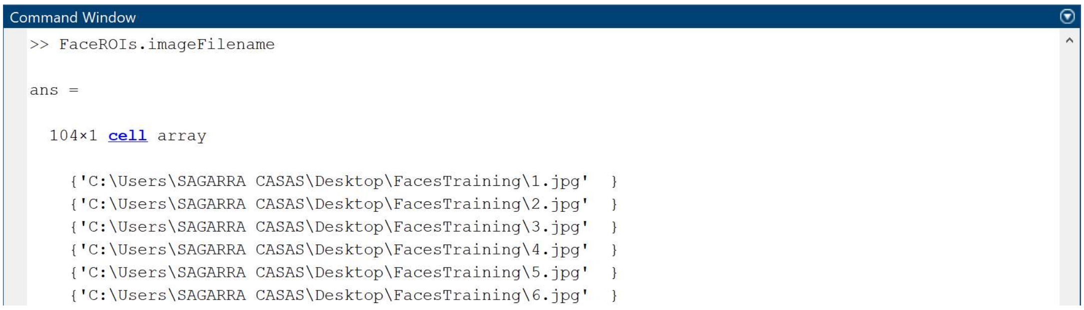
Then the complete dataset will be visualized in the workspace and saved as a table variable.



```
>> FaceROIs

FaceROIs =

  104×2 table

                      imageFilename                           Face
    ________________________________________________    ____________

    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\1.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'  }    {2×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\3.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\4.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\5.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\6.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\7.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\8.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\9.jpg'  }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\10.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\11.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\12.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\13.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\14.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\15.jpg' }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\16.jpg' }    {1×4 double}
```

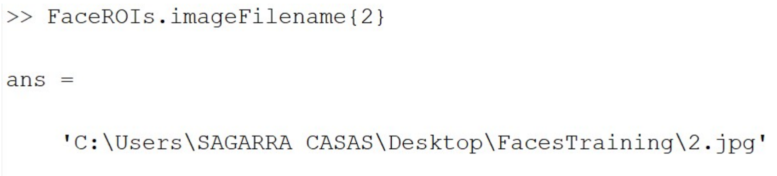
**Figure 5.1.7.** Visualization of the first samples of the training dataset, stored and visualized as a table

As it can be observed in the figure above, the first column corresponds to the image directories and the second corresponds to the bounding boxes stored as matrices of double type numbers.



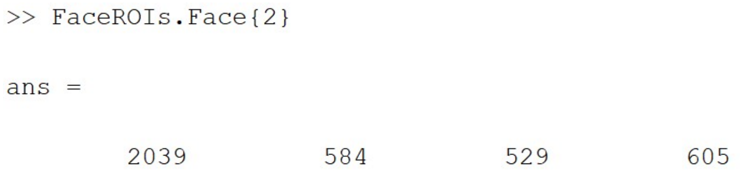
```
>> FaceROIs.imageFilename

ans =

  104×1 cell array

    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\1.jpg'  }
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'  }
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\3.jpg'  }
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\4.jpg'  }
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\5.jpg'  }
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\6.jpg'  }
```

**Figure 5.1.8.** Visualization of the imageFilename column



```
>> FaceROIs.imageFilename{2}

ans =

    'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'
```

**Figure 5.1.9.** Visualization of the directory of a specific image (which corresponds to second sample in the dataset)



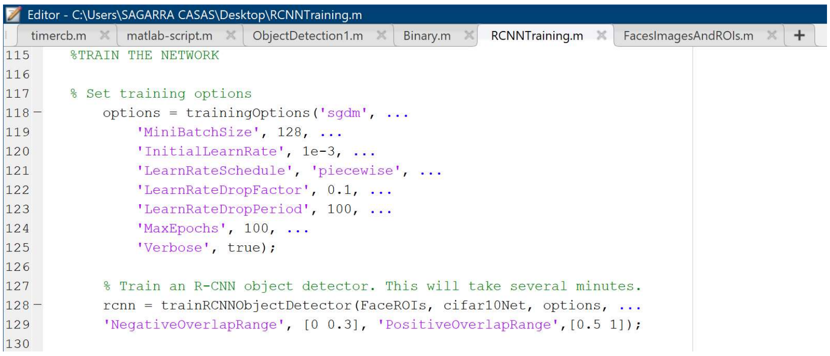
```
>> FaceROIs.Face{2}

ans =

        2039         584         529         605
```

**Figure 5.1.10.** Visualization of the bounding boxes coordinates of a specific image (which corresponds to second sample in the dataset)

## 5.1.2. Training the network

Although effectively, we will now train the network, it is important to remember that we are not training it from scratch but carrying out a learning transfer.

In order to design the training process, the variable that corresponds to the training dataset table is defined in the training code explained earlier (section *4.3.2. Transfer learning*) as the dataset to be used for the training process.

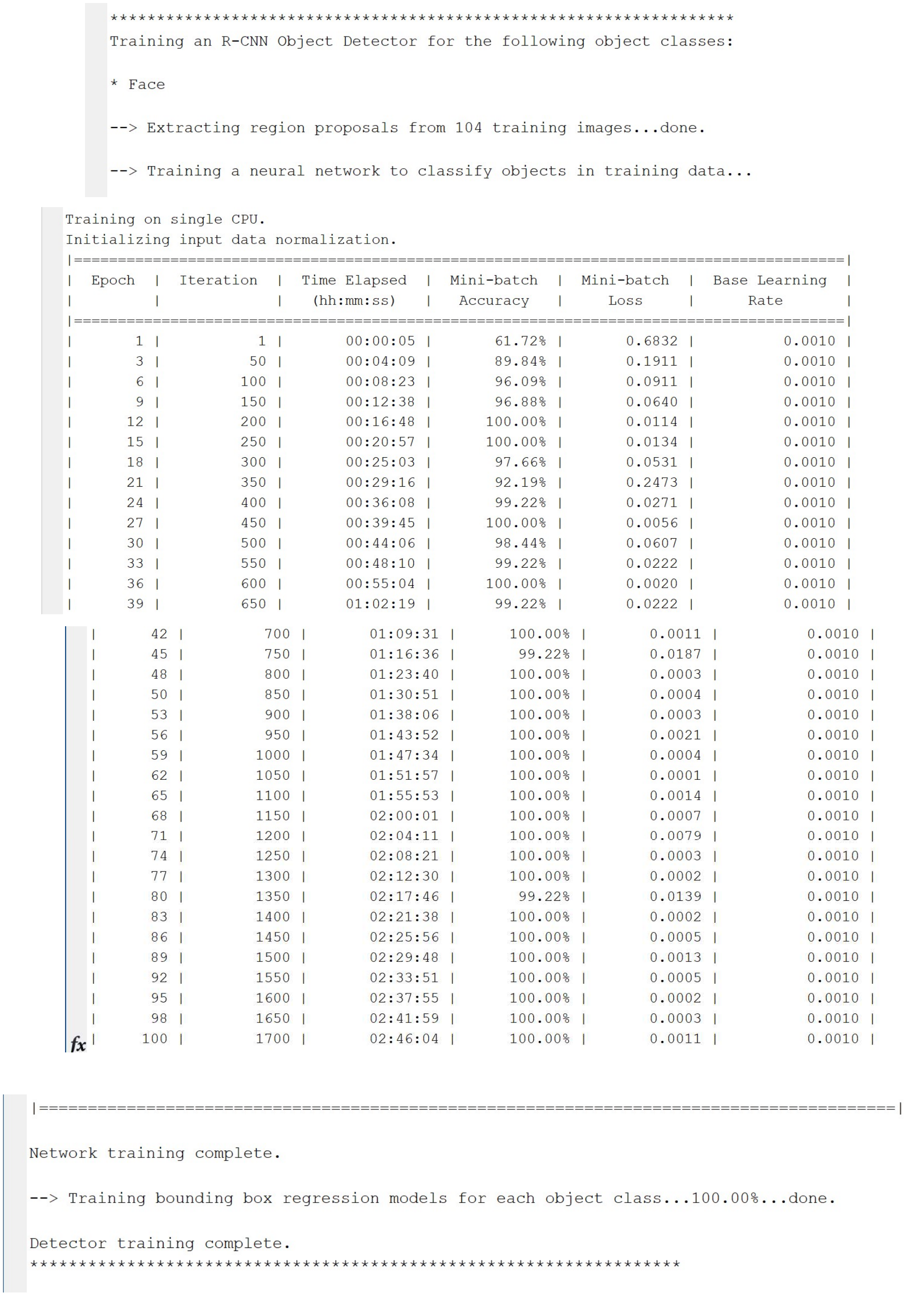


```
%TRAIN THE NETWORK

% Set training options
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 100, ...
        'Verbose', true);

    % Train an R-CNN object detector. This will take several minutes.
    rcnn = trainRCNNObjectDetector(FaceROIs, cifar10Net, options, ...
    'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1]);

```

**Figure 5.1.11.** Definition of the "FaceROIs" table as the dataset for the training process

Now, we run the transfer learning code. The next figure corresponds to the result code of the training process. This code shows the performance of the training during its execution, presenting the computed epochs and iterations, the time they have taken to compute and their accuracy and loss parameters. Accuracy and loss parameters are inversely related parameters that represent the precision and reliability of the network based on the correctness of the analysis it does in each epoch over the image – ROIs sample pairs. The greater the accuracy value, the better the performance of the network is. As greater the loss value is, larger have to be the changes on the weights and functions of the perceptrons and layers of the network in order to achieve a final accurate operating model.

The first accuracy value shown in the code is the accuracy of the performance of the pretrained network trained with the Cifar-10 dataset, not the accuracy of the result network trained to detect faces.

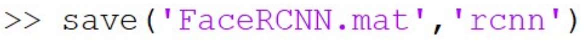


```
*******************************************************************
Training an R-CNN Object Detector for the following object classes:

* Face

--> Extracting region proposals from 104 training images...done.

--> Training a neural network to classify objects in training data...

Training on single CPU.
Initializing input data normalization.
|=====================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=====================================================================================|
|       1 |           1 |       00:00:05 |       61.72% |       0.6832 |          0.0010 |
|       3 |          50 |       00:04:09 |       89.84% |       0.1911 |          0.0010 |
|       6 |         100 |       00:08:23 |       96.09% |       0.0911 |          0.0010 |
|       9 |         150 |       00:12:38 |       96.88% |       0.0640 |          0.0010 |
|      12 |         200 |       00:16:48 |      100.00% |       0.0114 |          0.0010 |
|      15 |         250 |       00:20:57 |      100.00% |       0.0134 |          0.0010 |
|      18 |         300 |       00:25:03 |       97.66% |       0.0531 |          0.0010 |
|      21 |         350 |       00:29:16 |       92.19% |       0.2473 |          0.0010 |
|      24 |         400 |       00:36:08 |       99.22% |       0.0271 |          0.0010 |
|      27 |         450 |       00:39:45 |      100.00% |       0.0056 |          0.0010 |
|      30 |         500 |       00:44:06 |       98.44% |       0.0607 |          0.0010 |
|      33 |         550 |       00:48:10 |       99.22% |       0.0222 |          0.0010 |
|      36 |         600 |       00:55:04 |      100.00% |       0.0020 |          0.0010 |
|      39 |         650 |       01:02:19 |       99.22% |       0.0222 |          0.0010 |
|      42 |         700 |       01:09:31 |      100.00% |       0.0011 |          0.0010 |
|      45 |         750 |       01:16:36 |       99.22% |       0.0187 |          0.0010 |
|      48 |         800 |       01:23:40 |      100.00% |       0.0003 |          0.0010 |
|      50 |         850 |       01:30:51 |      100.00% |       0.0004 |          0.0010 |
|      53 |         900 |       01:38:06 |      100.00% |       0.0003 |          0.0010 |
|      56 |         950 |       01:43:52 |      100.00% |       0.0021 |          0.0010 |
|      59 |        1000 |       01:47:34 |      100.00% |       0.0004 |          0.0010 |
|      62 |        1050 |       01:51:57 |      100.00% |       0.0001 |          0.0010 |
|      65 |        1100 |       01:55:53 |      100.00% |       0.0014 |          0.0010 |
|      68 |        1150 |       02:00:01 |      100.00% |       0.0007 |          0.0010 |
|      71 |        1200 |       02:04:11 |      100.00% |       0.0079 |          0.0010 |
|      74 |        1250 |       02:08:21 |      100.00% |       0.0003 |          0.0010 |
|      77 |        1300 |       02:12:30 |      100.00% |       0.0002 |          0.0010 |
|      80 |        1350 |       02:17:46 |       99.22% |       0.0139 |          0.0010 |
|      83 |        1400 |       02:21:38 |      100.00% |       0.0002 |          0.0010 |
|      86 |        1450 |       02:25:56 |      100.00% |       0.0005 |          0.0010 |
|      89 |        1500 |       02:29:48 |      100.00% |       0.0013 |          0.0010 |
|      92 |        1550 |       02:33:51 |      100.00% |       0.0005 |          0.0010 |
|      95 |        1600 |       02:37:55 |      100.00% |       0.0002 |          0.0010 |
|      98 |        1650 |       02:41:59 |      100.00% |       0.0003 |          0.0010 |
|     100 |        1700 |       02:46:04 |      100.00% |       0.0011 |          0.0010 |
|=====================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.

Detector training complete.
*******************************************************************
```

**Figure 5.1.12.** Performance of the training process of the face detection neural network

It can be observed that the developed neural network is reliable and efficient as it presents final loss values lower than 0.001 and final accuracy values equal to 100%.

Now the trained RCNN network can be saved to our computer.



```
>> save('FaceRCNN.mat','rcnn')
```

**Figure 5.1.13.** Code executed to save the network as a rcnn object detector named “FaceRCNN.mat”

### 5.1.3. Testing the network

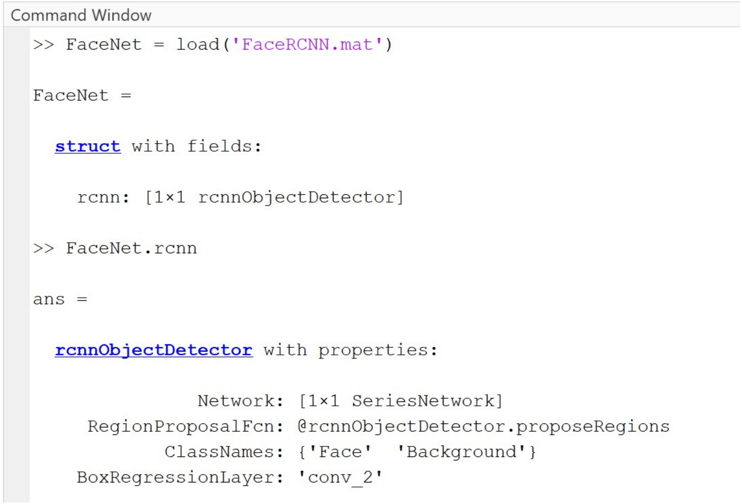
Now the created RCNN has to be tested in order to evaluate its performance. With the purpose of doing so, a set of testing images has to be analysed by the network in order to study the accuracy and robustness with which the networks detects and locates faces in images.

In order to analyse an image, we load the RCNN which has just been saved. Then we execute the RCNN “detect” function having previously set the image to analyse as the entry data of the network and the neural network as the face detection RCNN.

The “detect” function is a Matlab command that is used to detect, locate and classify objects in an image by using a trained neural network. The outputs of this function are: The coordinates of the bounding boxes of each detected object, the score probability for each object of having been detected and located correctly, the labels that correspond to the class of each detected object.

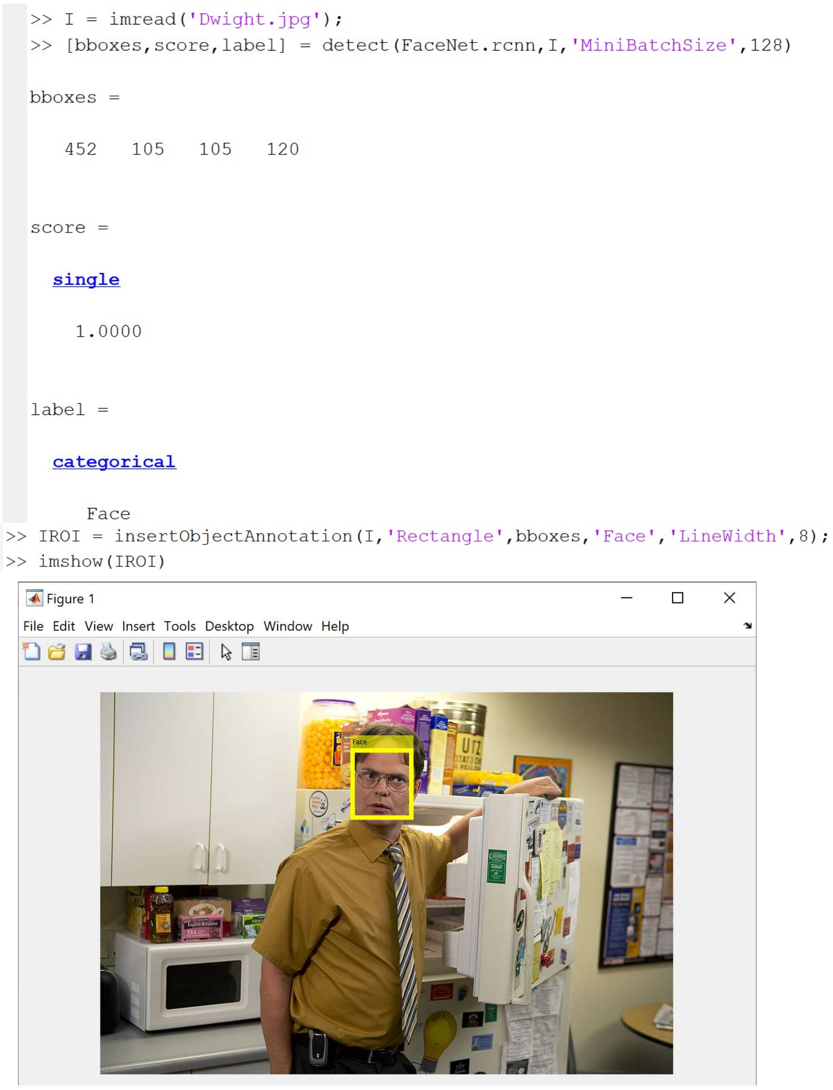
Finally, we visualize the generated bounding boxes on the image and study their corresponding label names and scores.

The following is the complete executed code used to analyse an image, whose file name is “Dwight.jpg”, and its results:



Command Window

```
>> FaceNet = load('FaceRCNN.mat')

FaceNet =

  struct with fields:

    rcnn: [1×1 rcnnObjectDetector]

>> FaceNet.rcnn

ans =

  rcnnObjectDetector with properties:

              Network: [1×1 SeriesNetwork]
    RegionProposalFcn: @rcnnObjectDetector.proposeRegions
           ClassNames: {'Face'  'Background'}
   BoxRegressionLayer: 'conv_2'
```



```
>> I = imread('Dwight.jpg');
>> [bboxes,score,label] = detect(FaceNet.rcnn,I,'MiniBatchSize',128)

bboxes =

   452   105   105   120


score =

  single

    1.0000


label =

  categorical

     Face
>> IROI = insertObjectAnnotation(I,'Rectangle',bboxes,'Face','LineWidth',8);
>> imshow(IROI)
```

**Figure 5.1.14.** Code executed to analyse an image with the face detection neural network, and its results

As it can be observed in the figure above, the network detected a face in the analysed image with an accuracy that the RCNN itself catalogued as a 100%. As expected, the network correctly detected and located all the faces in the analysed image.

The following figure corresponds to the result of the analysis of another image with the face detection network, where the scores of the detected faces have values of a 100% and the bounding boxes correctly locate the faces.

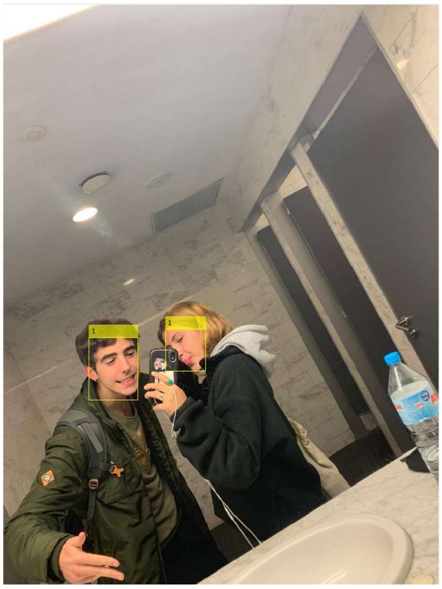


**Figure 5.1.15.** Face detection and location with the trained RCNN

### 5.1.4. Performance problems of the neural network

While testing the network, some adversities and limitations have been found in its performance.

Problems with false positives:

Although real faces are normally detected with scores equal to 1 (which correspond to an accuracy of a 100%), sometimes the network performs erroneous detections of objects, classifying them as faces when they are not. From all the image analysis that have been done in order to study the behaviour of the network, the following conclusions have been taken:

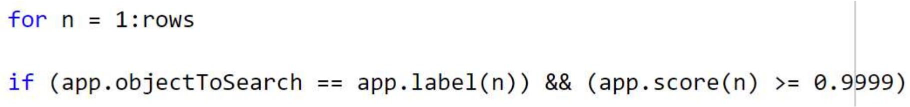
- The 80% (speaking in terms of probability) of the scores corresponding to wrongly detected faces are less than 1, having values between 0.6 and 0.99.
- The 20% of the scores corresponding to wrongly detected faces are equal to 1.
- The 70% of the scores corresponding to correctly detected faces are equal to 1.
- The 30% of the scores corresponding to correctly detected faces are less than 1, having values between 0.6 and 0.99.

In order to solve the false positive classification problem, a filter is applied to the Matlab final application just after the analysis of the corresponding image, so that only detected objects with a greater score value than 0.99 are communicated to the user as objects that correspond to the object type he or she is searching for.

However this solution has the problem of letting wrongly classified objects with a score greater than 0.99 be communicated to the user, and dismissing the correctly classified objects with a score smaller than 0.99. Nevertheless, we accept this bias as just a 20% of the wrongly detected objects have greater scores than 0.99 and a 30% of the correctly detected objects have smaller scores than 0.99.

The next figure shows the code lines of the final Matlab application that corresponds to this score filter. For each detected object, it is checked that the label of its classification corresponds to the object type the user has ordered to search for, and that its accuracy score is greater than 0.9999. If this conditions are fulfilled, the class and location of the object are uploaded to the Firebase, in order to communicate them to the user.

Through empirical testing, it has been proved that the results are better with a score threshold of 0.9999. "app.objectToSearch" represents the name of the object class the user wants to search, "app.slabel(n)" its classification label and "app.score(n)" its accuracy score. There are as many rows as detected objects in the image.



```
for n = 1:rows

if (app.objectToSearch == app.label(n)) && (app.score(n) >= 0.9999)
```

**Figure 5.1.16.** Score and label filter for image analysis

The next figure shows two image analysis where three objects have been wrongly detected as faces (with scores of 0.89494 and 0.63128 for the image at the left and a score of 0.9766 for the image at the right) and a face has correctly been detected (with a score of 1, in the image at the left).

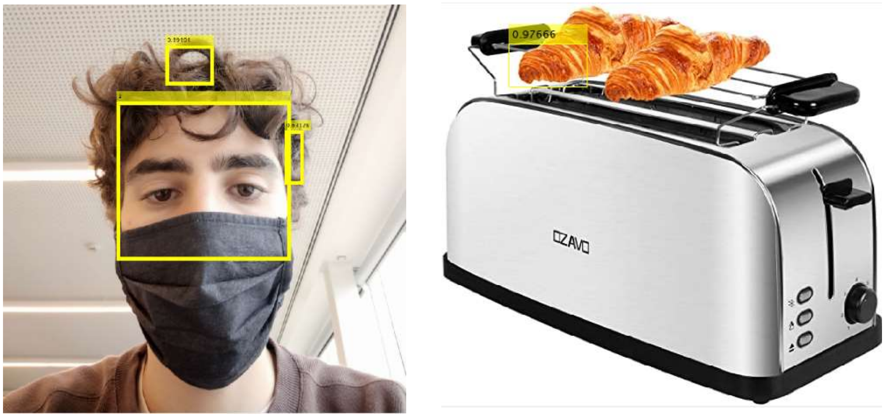


**Figure 5.1.17.** Wrongly and correctly detected objects in images, and their accuracy scores

With the network testing phase it has been observed that the problems with erroneous image analysis are directly related to the image resolution and orientation.

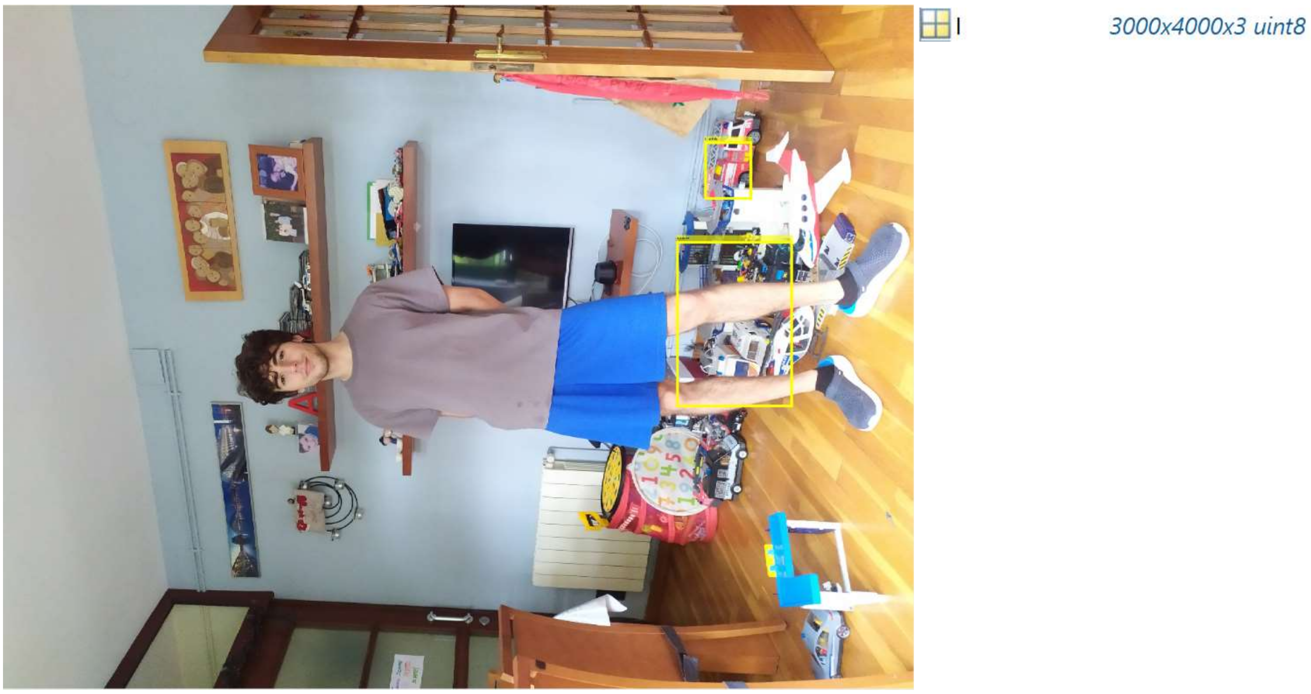
- Problems due to excess of resolution: If the analysed image has a large resolution, it is highly probable that not only the real faces are not correctly located or even not detected, but that multiple objects and sets of pixels are wrongly detected as faces.

With this face recognition RCNN that we have trained, images tend to cause problems with resolutions larger than 800 pixels high and as the resolution increases the analysis gets worse.

- Problems due to scarcity of resolution: If the analysed image has a low resolution, it is highly probable that few or no detections are made when it is analysed. This means that although no false positives will be detected, real faces that appear in the image will not be detected.
  With this face recognition RCNN that we have trained, images tend to cause problems with resolutions lower than 300 pixels high and as the resolution decreases from this threshold the analysis gets worse.

- Problems due to image orientation: If the faces that appear in the image are placed vertically, which is the natural orientation of faces when people are standing, they tend to be correctly detected with high accuracy scores. However, if a face in the analysed image is horizontally oriented, or tends to be horizontal, it may not be detected.
  This phenomenon has a really simple explanation, which is that all the faces used for the training of the neural network were vertically oriented or presented a very slight diagonal orientation.

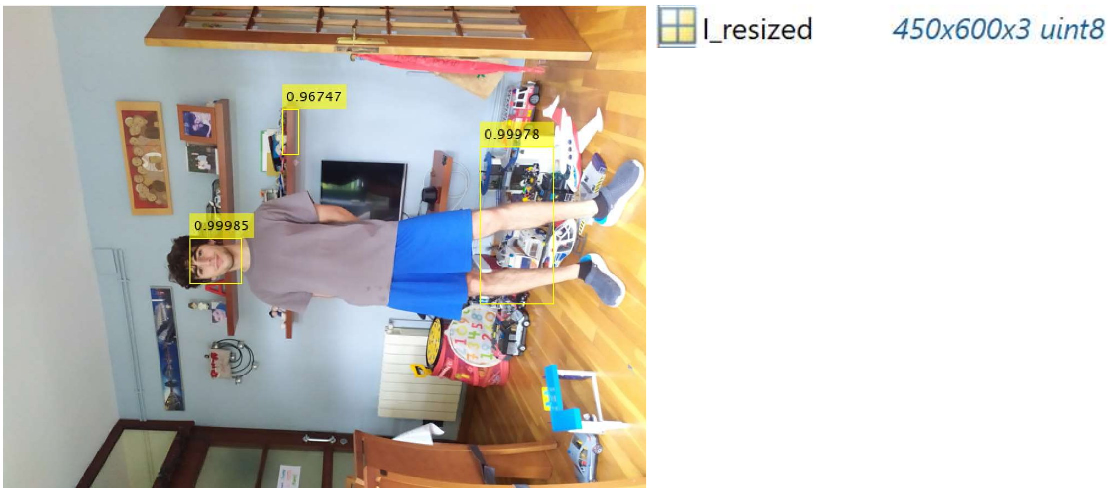
In the next figure, the analysis problems that generates the inspection of an image with excess of resolution and wrong orientation can be observed. On the left, the analysis result; on the right, the dimensions of the image.



I 3000x4000x3 uint8

**Figure 5.1.19.** Analysis result and dimensions of a high-resolution and rotated image

The following picture shows the performance of the analysis having reduced the dimensions of the image. The number of wrongly detected objects does not decrease but the face is correctly recognized. On the left, the analysis result; on the right, the dimensions of the image.



I_resized 450x600x3 uint8

**Figure 5.1.20.** Analysis result and dimensions of a moderate-resolution and rotated image

The following picture shows the performance of the analysis having reduced the dimensions of the image and having correctly reoriented it. The number of wrongly detected objects decreases to 0 and the face is correctly recognized.

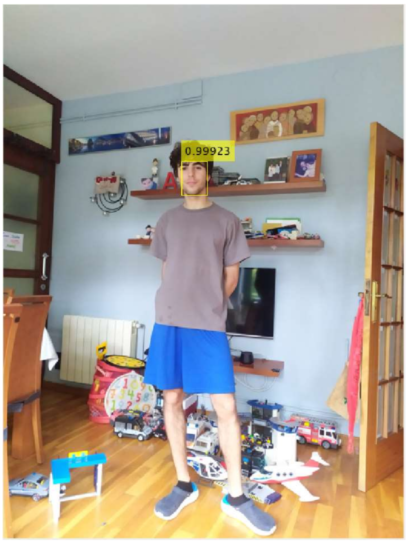
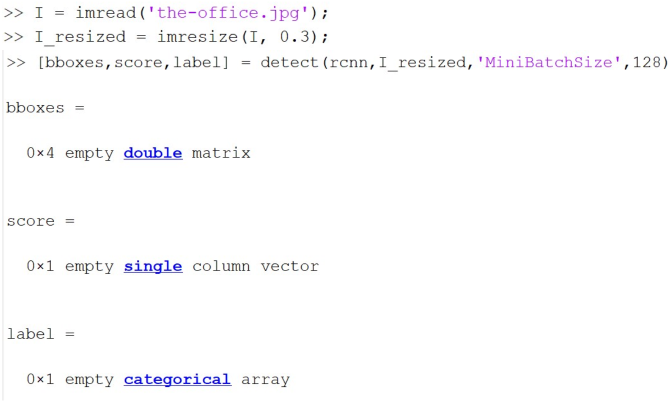


**Figure 5.1.21.** Analysis of a moderate-resolution and correctly oriented image

The next figure shows the analysis problems that an image with scarcity of resolution generates. The original image (which has a face in it) has been resized to a height dimension of 165 pixels in order to carry out this test. It can be observed that the network does not detect any face in the image.

The first line of code corresponds to the reading of the image, the second to its resizing to a 30% of its original dimensions, and the following lines of code correspond to the analysis of the image and its results.



```
>> I = imread('the-office.jpg');
>> I_resized = imresize(I, 0.3);
>> [bboxes,score,label] = detect(rcnn,I_resized,'MiniBatchSize',128)

bboxes =

  0×4 empty double matrix


score =

  0×1 empty single column vector


label =

  0×1 empty categorical array
```

**Figure 5.1.22.** Analysis of a low-resolution image, where no objects are detected

The next figure shows the analysis of the previous image with its original dimensions, which correspond to a height of 550 pixels. It can be observed that the face located in the image is correctly detected.

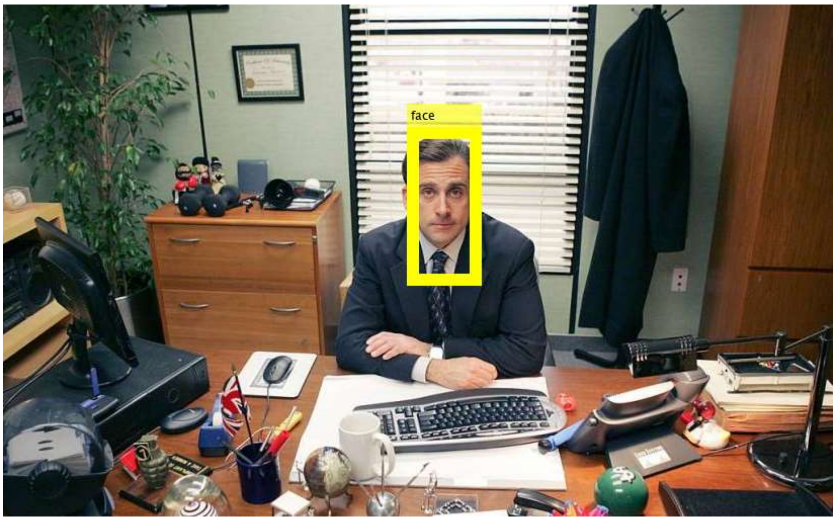
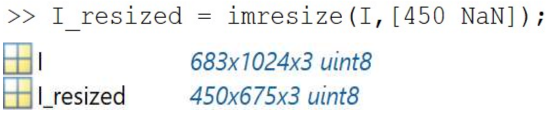


**Figure 5.1.23.** Analysis of a moderate-resolution image

The next code computes the resizing of the image stored in the "I" variable to the same image with a height dimension of 450 pixels, preserving the original dimensional relations between its height and width. At the bottom can be observed the dimensions of both the original and the resized image.



```
>> I_resized = imresize(I,[450 NaN]);
I              683x1024x3 uint8
I_resized      450x675x3 uint8
```

**Figure 5.1.24.** Matlab code used to resize images

It has been empirically proved while testing the performance of the RCNN that images with a height of 450 pixels are optimal and present better results when are analysed by the face detection network.

Therefore, in the code of the final Matlab application, images are resized to a height of 450 pixels for its analysis with the trained RCNNs. As images downloaded from the Firebase are rotated 90 degrees to the left by default, a function that reorients the images is executed before its analysis. Images are thus rotated 270 grades to he left in order to transform them to their original orientation. This is due to the fact that we assume that the majority of faces located in the pictures taken by the user will be oriented vertically.

## 5.2. Training and testing the network for detection and location of multiple objects

The next step in order to make a richer and more useful neural network is making it capable of detecting, recognizing and locating multiple objects, which may pertain to different classes, from given images.

The number of different classes and the specific types of objects that we want the network to be capable of detecting and recognizing can be as wide and varied as the programmer decides it to be. In our case, understanding that the procedure for adding a specific class to the network training is mechanic, repeatable and relatively low cost (as it just takes Matlab software and appropriate data in image format), we'll just train the network with the objective of detecting five types of objects. As the purpose of the software is helping people move and get along in indoor spaces, as previously explained, the object classes chosen to introduce to the network training are:

- Faces (therefore, people that may be looking to the user's direction)
- Lamps
- Keys
- Laptops
- Backpacks

The process for the network training will be the following:

From specific images, extract and document all those ROIs that contain each of the specific objects we want to work with, so that for each object class we have approximately from 100 to 150 labelled ROIs that refer to coordinates on the images.

Afterwards, run the code that carries out the training procedure of the neural network, introducing the images with their corresponding labelled ROIs as the training data for the supervised learning.

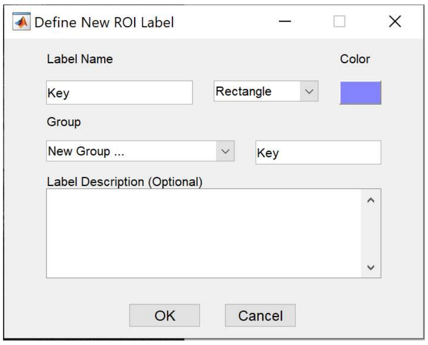
Finally, test and evaluate the neural network for multiple object detection.

### 5.2.1. Creation of the training dataset

As all necessary ROIs for the “Face” class have been previously labelled, it will be efficient to open the Matlab Image Labeller session used to document faces, import more specific images and label ROIs for the remaining object classes on the new images. Once done, it will be possible to export the labelled ROIs and images to the workspace in order to use them to train the network.

To Load the session used to label faces open the image labeller, go to “Load”, click “Session” and choose the corresponding Matlab file. In order to import images, go to “Load”, click “Add images from folder” and select all those images and folders to be documented.

Once done, add as many label groups as needed.



Define New ROI Label

Label Name: Key

Rectangle

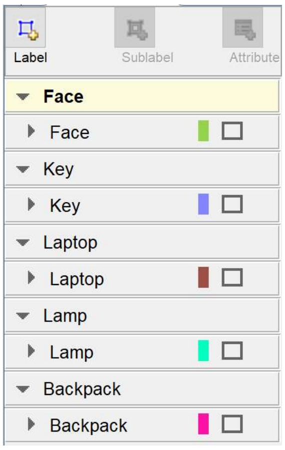
Color: 🟦

Group: New Group ...

Key

Label Description (Optional)

**Figure 5.2.1.** Creation of the “Key” label in the Image Labeler



- Face
  - Face 🟩 □
- Key
  - Key 🟦 □
- Laptop
  - Laptop 🟥 □
- Lamp
  - Lamp 🟩 □
- Backpack
  - Backpack 🟪 □

**Figure 5.2.2.** Set of created labels

Then, for every image select the ROIs that contain each of the objects, with the corresponding label for each object class.

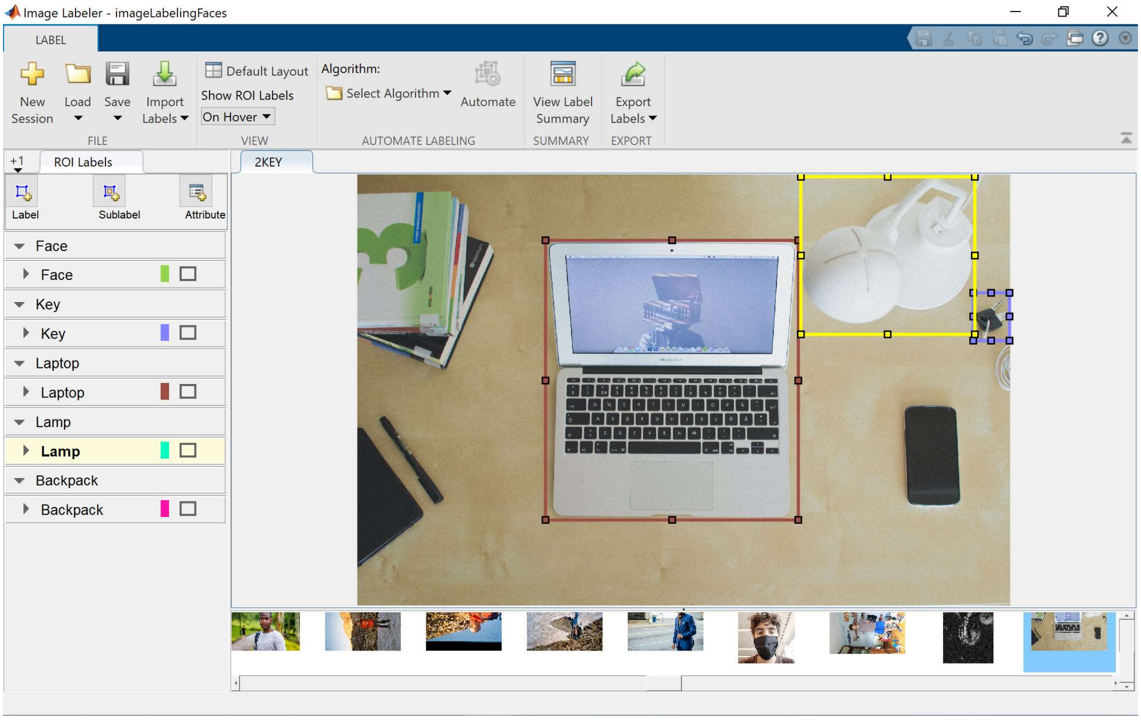


**Figure 5.2.3.** ROI labelling for multiple object classes

Considerations to take into account when documenting images:

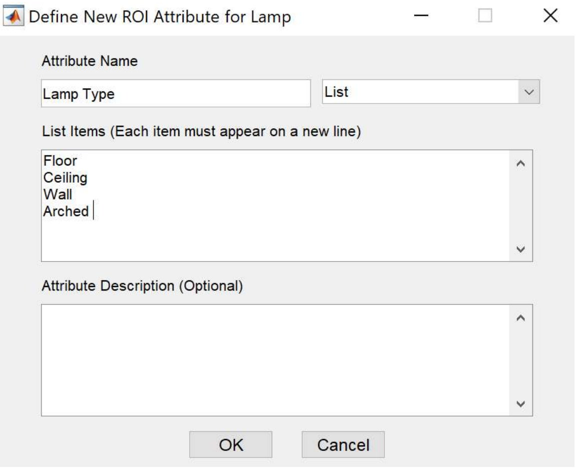
As all faces tend to have similar structures, forms and features, as it happens for keys, laptops and backpacks. We can document these four object classes with just one label for each class.

However, this doesn't happen with lamps, which have different characteristics and forms depending on their design. Therefore, not only we can document all lamps with the "Lamp" label, but add attributes to the labelling that define what type of design each lamp presents.

With the dataset we have used to train the network for detecting lamps, we can subgroup all lamps with four attributes that define their design. These four attributes are the following:

- Floor
- Ceiling
- Wall
- Arched

To add attributes select the label you want to define attributes for, go to "Attribute" and define the list of items you want to create.



Define New ROI Attribute for Lamp

Attribute Name: Lamp Type — List

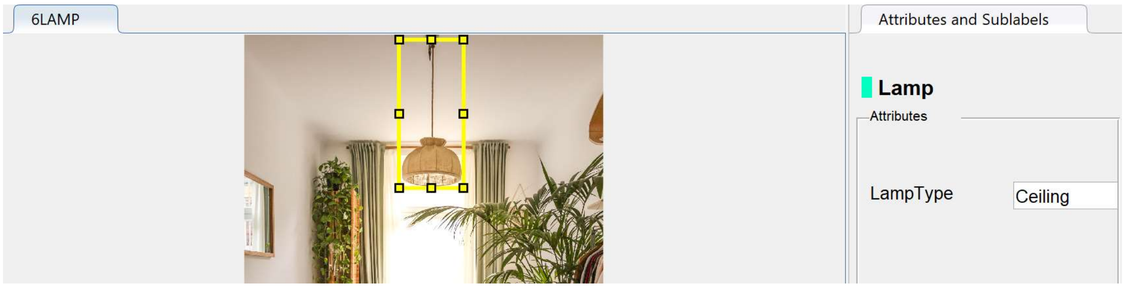
List Items (Each item must appear on a new line)

Floor
Ceiling
Wall
Arched

Attribute Description (Optional)

**Figure 5.2.4.** Creation of attributes for the documentation of lamps in the dataset

Then each lamp can be assigned to its corresponding attribute.

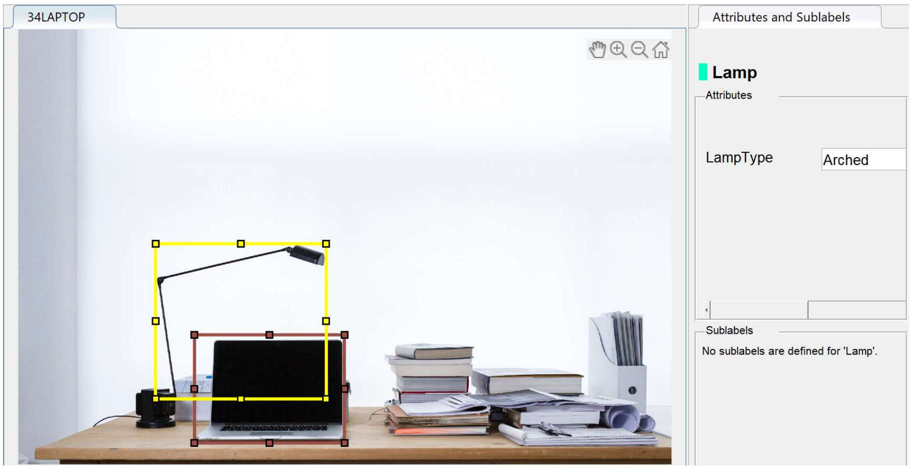


Lamp

Attributes

LampType: Ceiling

**Figure 5.2.5.** Documentation of a ROI containing a lamp with the “Ceiling” attribute as the lamp corresponds to a ceiling lamp type



Lamp

Attributes

LampType: Arched

Sublabels

No sublabels are defined for 'Lamp'.

**Figure 5.2.6.** Documentation of a ROI containing a lamp with the “Arched” attribute as the lamp corresponds to an arched lamp type

However, it is important to be aware that the RCNN network used in this project cannot be trained with datasets that contain attributes. Thus, adding attributes to labels may be a great consideration when documenting images and defining ROIs for some particular purposes, but it cannot be done if the labels are exported in order to train a convolutional object detection network.

When all documentation and labelling work has been done, by clicking "View Label Summary" a histogram that shows how many labelled ROIs each object class has in each image, can be visualized.

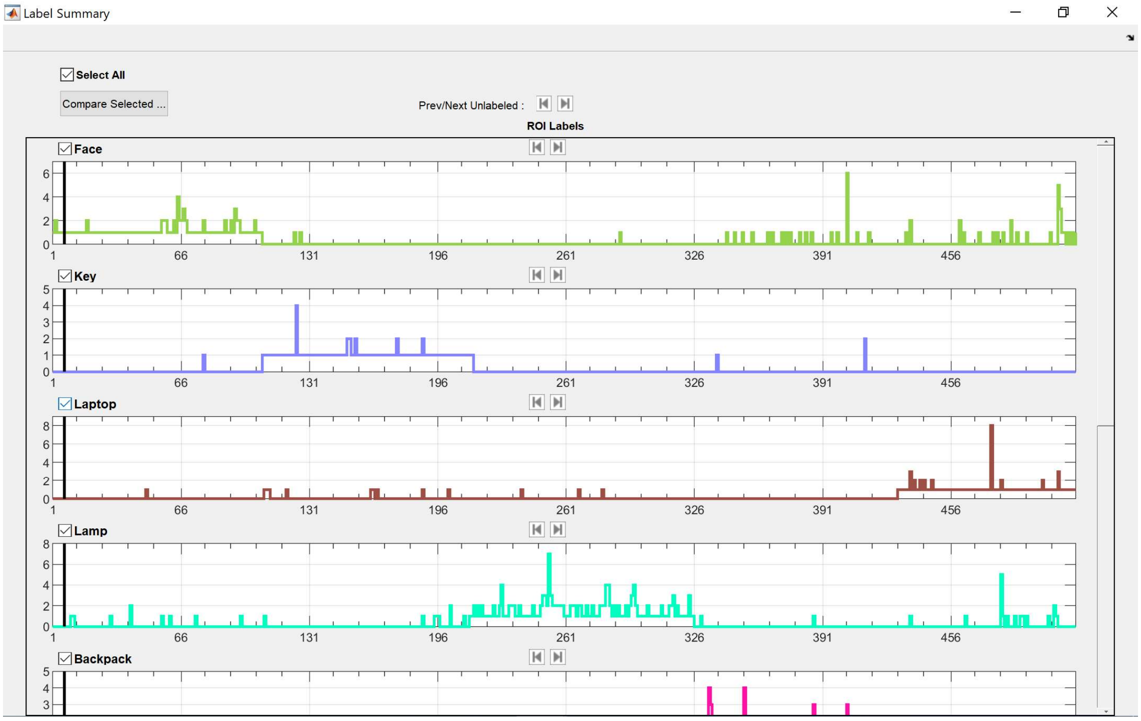


**Figure 5.2.7.** Summary histogram with the information of the appearance in the images of each of the 5 labels

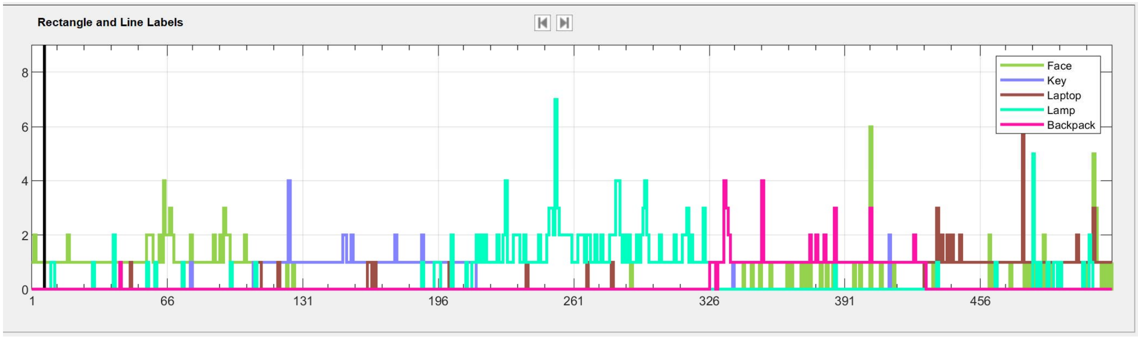
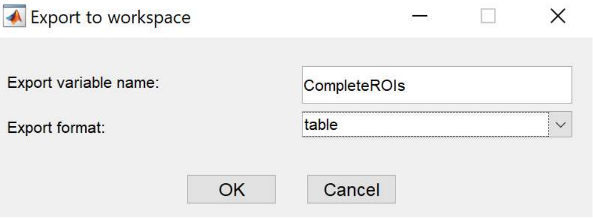


**Figure 5.2.8.** Compacted summary histogram

Finally, as previously done when we developed the face recognition neural network, we export the labels to the workplace.



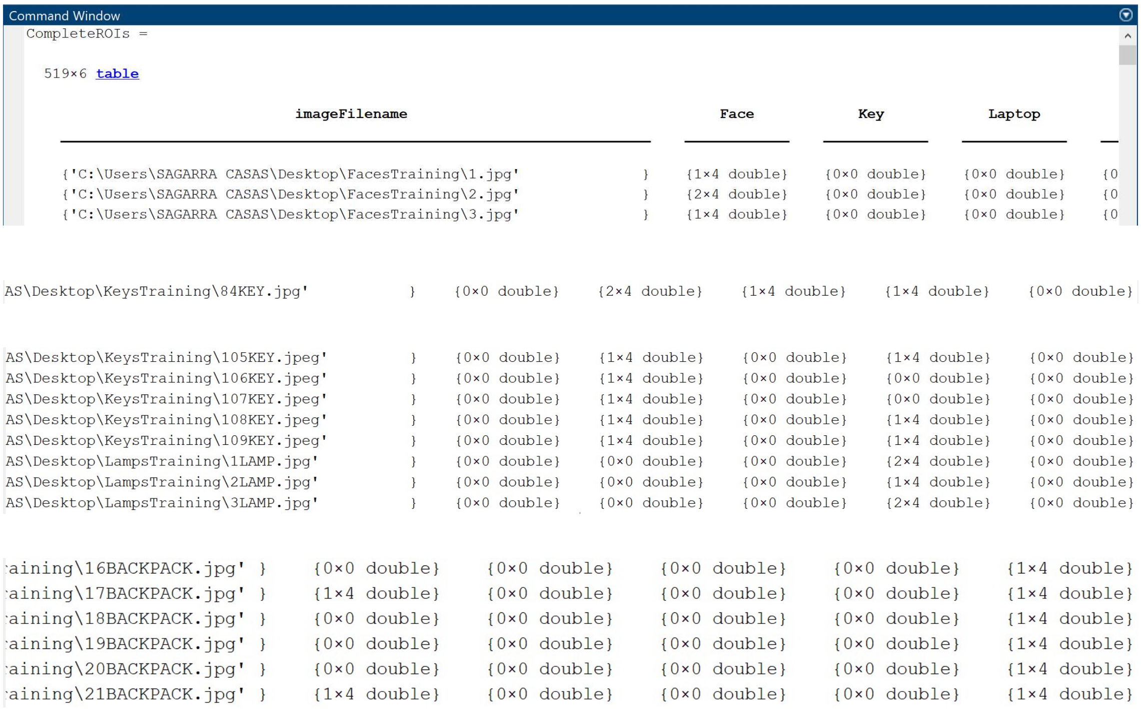
Export to workspace

Export variable name: CompleteROIs

Export format: table

**Figure 5.2.8.** Exportation of the dataset to the workspace

The following are some extracts of the generated table, which we can observe that is formed by 519 images and 5 object types. The second column corresponds to the ROIs that contain faces, the third corresponds to the key ROIs, the fourth to the laptop ROIs, the fifth to the lamp ROIs and the sixth corresponds to the ROIs that contain backpacks.



```
CompleteROIs =

  519×6 table

                         imageFilename                              Face          Key          Laptop

    ___________________________________________________________    ____________    ____________    ____________    __

    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\1.jpg'      }    {1×4 double}    {0×0 double}    {0×0 double}    {0
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'      }    {2×4 double}    {0×0 double}    {0×0 double}    {0
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\3.jpg'      }    {1×4 double}    {0×0 double}    {0×0 double}    {0
```

```
AS\Desktop\KeysTraining\84KEY.jpg'           }    {0×0 double}    {2×4 double}    {1×4 double}    {1×4 double}    {0×0 double}
```

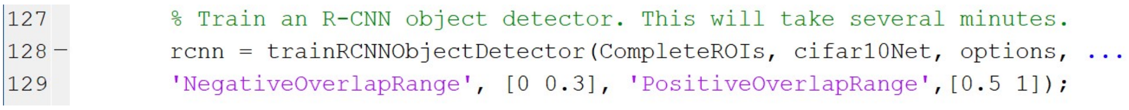
```
AS\Desktop\KeysTraining\105KEY.jpeg'         }    {0×0 double}    {1×4 double}    {0×0 double}    {1×4 double}    {0×0 double}
AS\Desktop\KeysTraining\106KEY.jpeg'         }    {0×0 double}    {1×4 double}    {0×0 double}    {0×0 double}    {0×0 double}
AS\Desktop\KeysTraining\107KEY.jpeg'         }    {0×0 double}    {1×4 double}    {0×0 double}    {0×0 double}    {0×0 double}
AS\Desktop\KeysTraining\108KEY.jpeg'         }    {0×0 double}    {1×4 double}    {0×0 double}    {1×4 double}    {0×0 double}
AS\Desktop\KeysTraining\109KEY.jpeg'         }    {0×0 double}    {1×4 double}    {0×0 double}    {1×4 double}    {0×0 double}
AS\Desktop\LampsTraining\1LAMP.jpg'          }    {0×0 double}    {0×0 double}    {0×0 double}    {2×4 double}    {0×0 double}
AS\Desktop\LampsTraining\2LAMP.jpg'          }    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}    {0×0 double}
AS\Desktop\LampsTraining\3LAMP.jpg'          }    {0×0 double}    {0×0 double}    {0×0 double}    {2×4 double}    {0×0 double}
```

```
aining\16BACKPACK.jpg'  }    {0×0 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
aining\17BACKPACK.jpg'  }    {1×4 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
aining\18BACKPACK.jpg'  }    {0×0 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
aining\19BACKPACK.jpg'  }    {0×0 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
aining\20BACKPACK.jpg'  }    {0×0 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
aining\21BACKPACK.jpg'  }    {1×4 double}    {0×0 double}    {0×0 double}    {0×0 double}    {1×4 double}
```

**Figure 5.2.9.** Specific extracts of the training dataset created for multiple object classes

## 5.2.2. Training the network

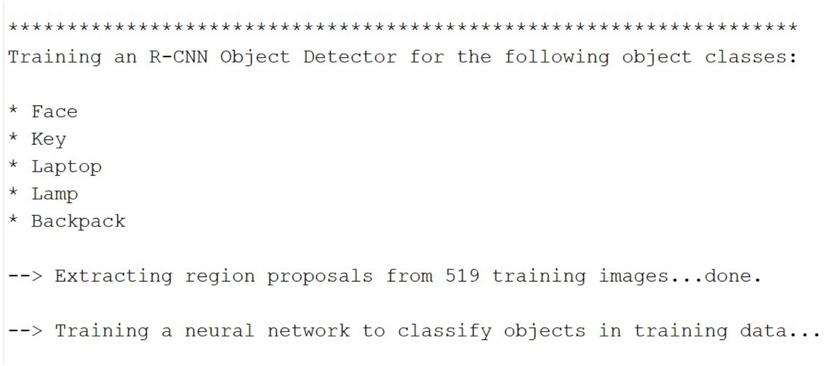
Now we finally train the network with the “CompleteROIs” dataset. The following figure corresponds to the lines of code of the transfer learning program where it can be seen that the dataset with multiple object classes is defined as the training dataset.



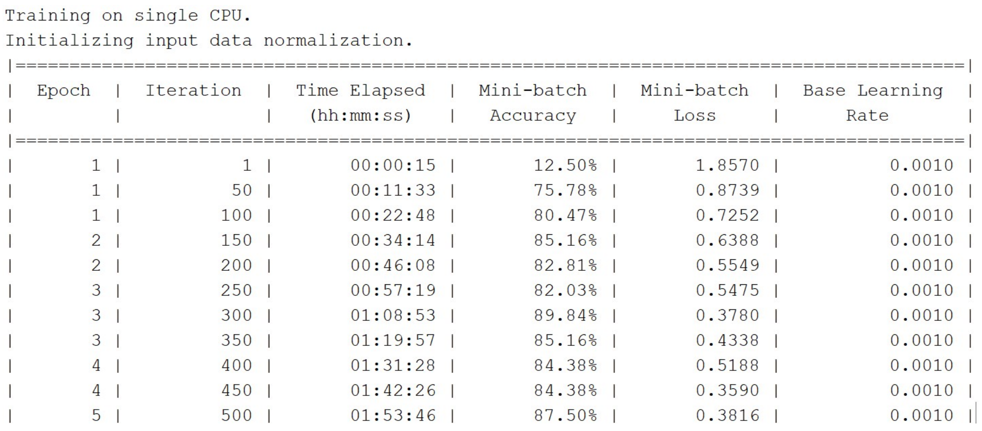
```
        % Train an R-CNN object detector. This will take several minutes.
        rcnn = trainRCNNObjectDetector(CompleteROIs, cifar10Net, options, ...
        'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1]);
```

**Figure 5.2.10.** Lines of code that correspond to the definition of the properties of the RCNN training process

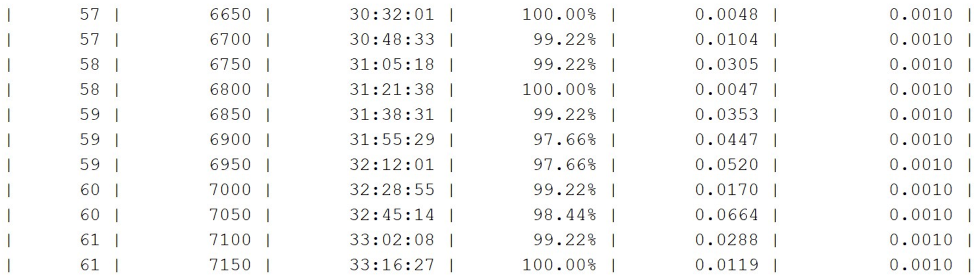
The following figures correspond to the training process, showing the steps taken during the execution of the training and its results. Not the entirety of the process is shown as a large number of iterations have been performed.



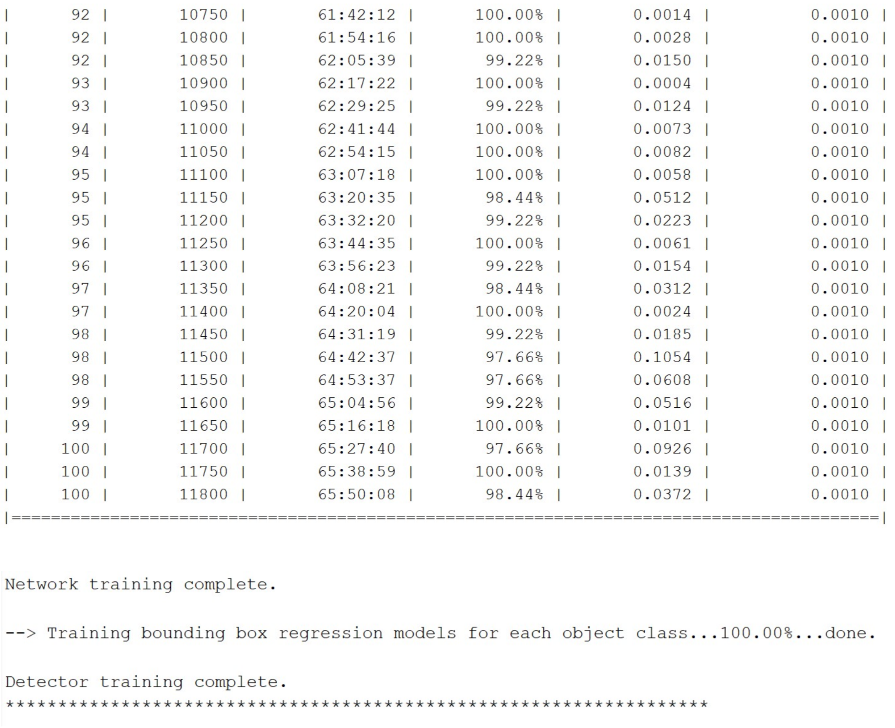
```
*******************************************************************
Training an R-CNN Object Detector for the following object classes:

* Face
* Key
* Laptop
* Lamp
* Backpack

--> Extracting region proposals from 519 training images...done.

--> Training a neural network to classify objects in training data...
```



```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:15 |       12.50% |       1.8570 |          0.0010 |
|       1 |          50 |       00:11:33 |       75.78% |       0.8739 |          0.0010 |
|       1 |         100 |       00:22:48 |       80.47% |       0.7252 |          0.0010 |
|       2 |         150 |       00:34:14 |       85.16% |       0.6388 |          0.0010 |
|       2 |         200 |       00:46:08 |       82.81% |       0.5549 |          0.0010 |
|       3 |         250 |       00:57:19 |       82.03% |       0.5475 |          0.0010 |
|       3 |         300 |       01:08:53 |       89.84% |       0.3780 |          0.0010 |
|       3 |         350 |       01:19:57 |       85.16% |       0.4338 |          0.0010 |
|       4 |         400 |       01:31:28 |       84.38% |       0.5188 |          0.0010 |
|       4 |         450 |       01:42:26 |       84.38% |       0.3590 |          0.0010 |
|       5 |         500 |       01:53:46 |       87.50% |       0.3816 |          0.0010 |
```



```
|      57 |        6650 |       30:32:01 |      100.00% |       0.0048 |          0.0010 |
|      57 |        6700 |       30:48:33 |       99.22% |       0.0104 |          0.0010 |
|      58 |        6750 |       31:05:18 |       99.22% |       0.0305 |          0.0010 |
|      58 |        6800 |       31:21:38 |      100.00% |       0.0047 |          0.0010 |
|      59 |        6850 |       31:38:31 |       99.22% |       0.0353 |          0.0010 |
|      59 |        6900 |       31:55:29 |       97.66% |       0.0447 |          0.0010 |
|      59 |        6950 |       32:12:01 |       97.66% |       0.0520 |          0.0010 |
|      60 |        7000 |       32:28:55 |       99.22% |       0.0170 |          0.0010 |
|      60 |        7050 |       32:45:14 |       98.44% |       0.0664 |          0.0010 |
|      61 |        7100 |       33:02:08 |       99.22% |       0.0288 |          0.0010 |
|      61 |        7150 |       33:16:27 |      100.00% |       0.0119 |          0.0010 |
```



```
|      92 |       10750 |       61:42:12 |      100.00% |       0.0014 |          0.0010 |
|      92 |       10800 |       61:54:16 |      100.00% |       0.0028 |          0.0010 |
|      92 |       10850 |       62:05:39 |       99.22% |       0.0150 |          0.0010 |
|      93 |       10900 |       62:17:22 |      100.00% |       0.0004 |          0.0010 |
|      93 |       10950 |       62:29:25 |       99.22% |       0.0124 |          0.0010 |
|      94 |       11000 |       62:41:44 |      100.00% |       0.0073 |          0.0010 |
|      94 |       11050 |       62:54:15 |      100.00% |       0.0082 |          0.0010 |
|      95 |       11100 |       63:07:18 |      100.00% |       0.0058 |          0.0010 |
|      95 |       11150 |       63:20:35 |       98.44% |       0.0512 |          0.0010 |
|      95 |       11200 |       63:32:20 |       99.22% |       0.0223 |          0.0010 |
|      96 |       11250 |       63:44:35 |      100.00% |       0.0061 |          0.0010 |
|      96 |       11300 |       63:56:23 |       99.22% |       0.0154 |          0.0010 |
|      97 |       11350 |       64:08:21 |       98.44% |       0.0312 |          0.0010 |
|      97 |       11400 |       64:20:04 |      100.00% |       0.0024 |          0.0010 |
|      98 |       11450 |       64:31:19 |       99.22% |       0.0185 |          0.0010 |
|      98 |       11500 |       64:42:37 |       97.66% |       0.1054 |          0.0010 |
|      98 |       11550 |       64:53:37 |       97.66% |       0.0608 |          0.0010 |
|      99 |       11600 |       65:04:56 |       99.22% |       0.0516 |          0.0010 |
|      99 |       11650 |       65:16:18 |      100.00% |       0.0101 |          0.0010 |
|     100 |       11700 |       65:27:40 |       97.66% |       0.0926 |          0.0010 |
|     100 |       11750 |       65:38:59 |      100.00% |       0.0139 |          0.0010 |
|     100 |       11800 |       65:50:08 |       98.44% |       0.0372 |          0.0010 |
|=========================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.

Detector training complete.
*******************************************************************
```

**Figure 5.2.11.** Visualization of the executed steps and their results during the training process

As it can be seen in the command window shown above, the training process has taken 65 hours and 50 minutes and 11800 iterations to complete.

However, not every epoch has been equally efficient. As the loss parameter, which as previously explained, we want it to be minimum in order to ensure the network is trained completely and correctly, rapidly decreases on the first epochs but when a certain number of iterations have been done, tends to stop the decrease and finally stagnates in a set of values around 0.01.

Moreover, there are epochs where their loss values are greater than the values corresponding to the previous ones. Which means that network finds it hard, with its architecture and the data used to train it, to find optimal weighs for the neurons that form its layers in order to be able to find accurate solutions to the posed object recognition problem.

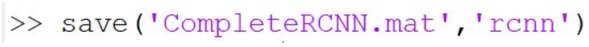
Furthermore, when the same network architecture was trained to detect faces with 104 images, the training process lasted 2 hours and 46 minutes and took 1700 iterations. Nevertheless, the training time in order to detect 5 different objects with 519 images, which is a number of categories and a dataset 5 times larger than the used for face recognition, is not 5 times greater than the necessary time for the face detection training but almost multiplies it by 24.

Therefore, we can conclude that as expected, the complexity of a RCNN training does not increase linearly but exponentially to its dataset and the number of object categories to detect and classify.

Although it would surely be a poor conclusion if we did not say that there is a possibility that, both the high training time and the inability of most iterations to improve the network performance, may be linked to the fact that the problem raised is too complex to solve by the designed convolutional network with the set of images, labels and categories used to carry out the training.

If the evaluation of the network, i.e. its test phase, does not present sufficiently satisfactory results, the partition of the problem could be considered as a possible solution. So that for each object class there is one network trained to recognize and locate objects that correspond just to this particular category. A study on which classes give more problems when carrying out the training could also be made. Eliminating those that give problems, from the set of categories of objects to detect, or replacing them by other object classes.

With all this considerations taken in mind, we proceed to the saving of the network.



```
>> save('CompleteRCNN.mat','rcnn')
```

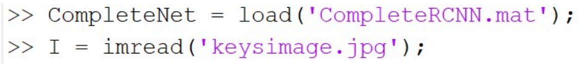
**Figure 5.2.12.** Saving of the trained RCNN for multiple object detection with the name "CompleteRCNN"

### 5.2.3. Testing the network

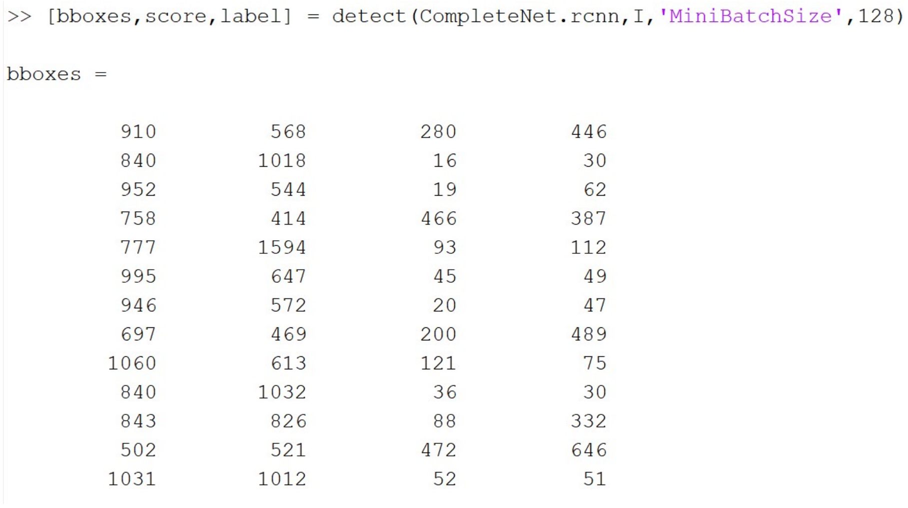
We now import testing images to the workspace and analyse them using the created RCNN. By doing this we can evaluate the accuracy and precision of the network.

As seen on the following figure, the procedure for the image analysis using the 'CompleteRCNN' is exactly the same that has been previously explained for the testing of the face detection network. We analyse an image whose file name is "keysimage.jpg" with the multiple object detection network that has just been saved.

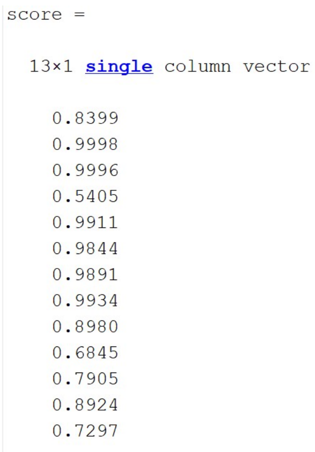
The figure below shows the executed code for the image analysis with the multiple object detector and its corresponding results.



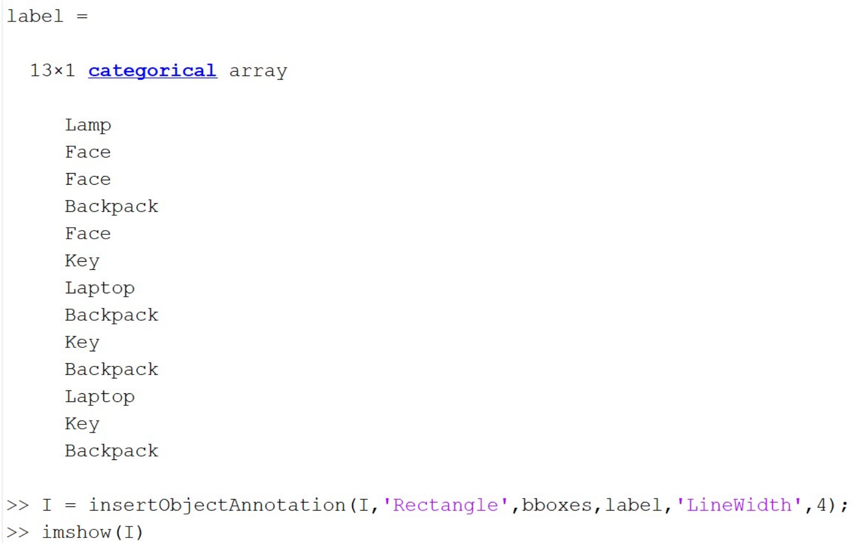
```
>> CompleteNet = load('CompleteRCNN.mat');
>> I = imread('keysimage.jpg');
```



```
>> [bboxes,score,label] = detect(CompleteNet.rcnn,I,'MiniBatchSize',128)

bboxes =

         910         568         280         446
         840        1018          16          30
         952         544          19          62
         758         414         466         387
         777        1594          93         112
         995         647          45          49
         946         572          20          47
         697         469         200         489
        1060         613         121          75
         840        1032          36          30
         843         826          88         332
         502         521         472         646
        1031        1012          52          51
```



```
score =

  13×1 single column vector

    0.8399
    0.9998
    0.9996
    0.5405
    0.9911
    0.9844
    0.9891
    0.9934
    0.8980
    0.6845
    0.7905
    0.8924
    0.7297
```



```
label =

  13×1 categorical array

     Lamp
     Face
     Face
     Backpack
     Face
     Key
     Laptop
     Backpack
     Key
     Backpack
     Laptop
     Key
     Backpack

>> I = insertObjectAnnotation(I,'Rectangle',bboxes,label,'LineWidth',4);
>> imshow(I)
```

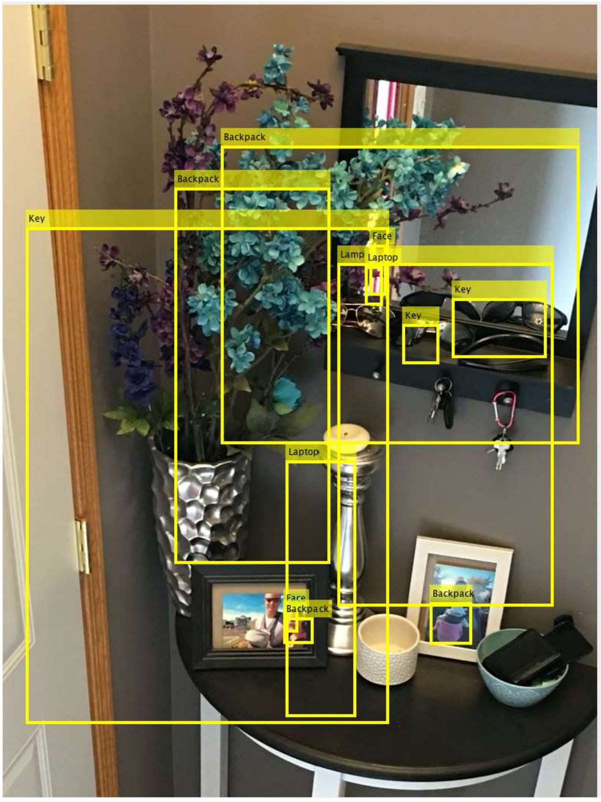


**Figure 5.2.13.** Code executed to analyse an image with the multiple object detection RCNN, and its results

As it can be seen, the result that the network suggests for this image is totally chaotic and disastrous. Not only are not the keys and faces not detected (which are the only elements that should be detected on this image) but the networks erroneously classifies some objects to categories that don't correspond to them and, moreover, gathers groups of pixels that don't form any object or that belong to various objects.

We resize the image and reanalyse it several times but the results turn out to be the same. Unlike with the face detector, the image analysis with the "CompleteRCNN" is unsatisfactory regardless of the resolution of the picture.

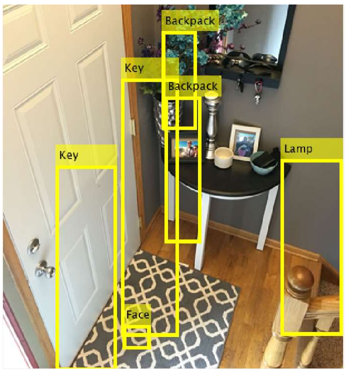


**Figure 5.2.14.** Analysis of a resized version of the original image, which results to be unsatisfactory

The following is a set of three images analysed by the ‘CompleteRCNN’:

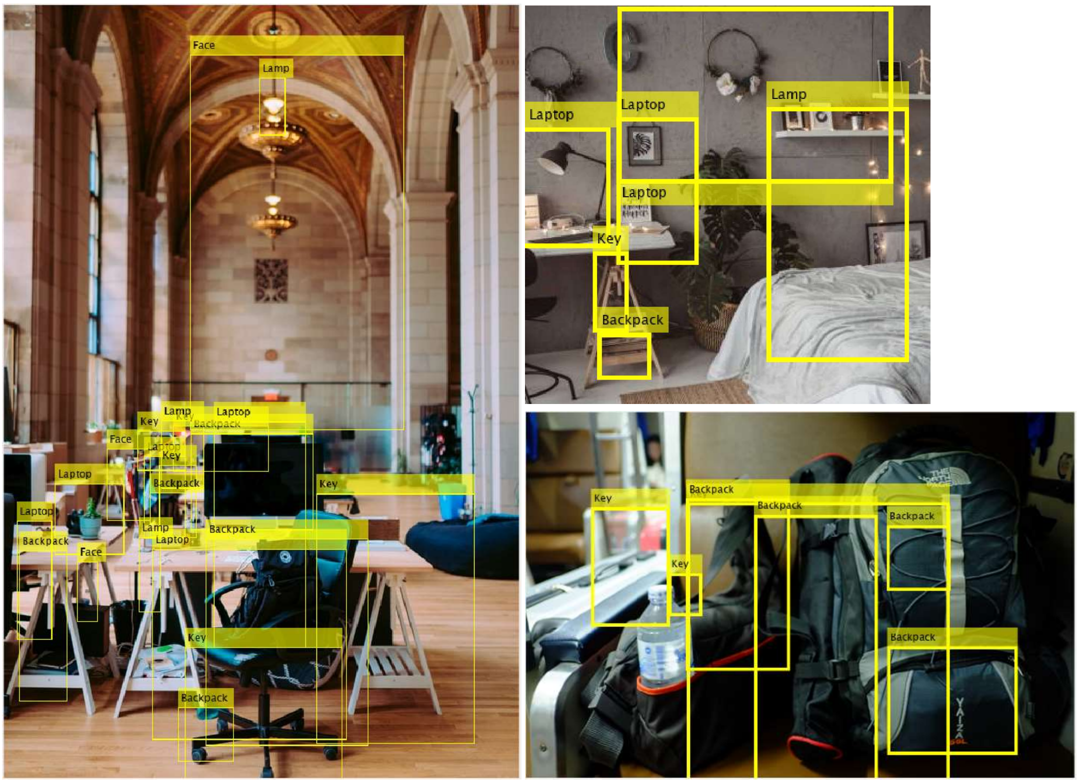


**Figure 5.2.15.** Analysis of various images with the multiple object detection network, which results to compute completely erroneous results

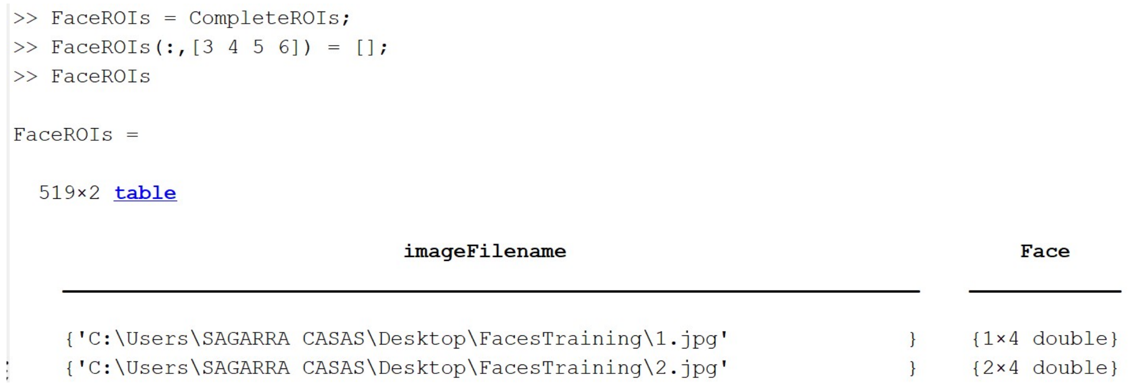
It can be clearly concluded that the created network for the recognition and location of five different object classes is not useful as it is not capable of resolving the object detection problem with enough accuracy and clarity.

Therefore, we proceed to train 5 different RCNNs, one for each object class.

## 5.3. Training and testing a single network for each object class

Instead of doing the labelling of the images again in order to obtain a single training dataset for each object class, the table “CompleteROIs” that was created grouping the five object classes can be sectioned in five tables that correspond to each class.

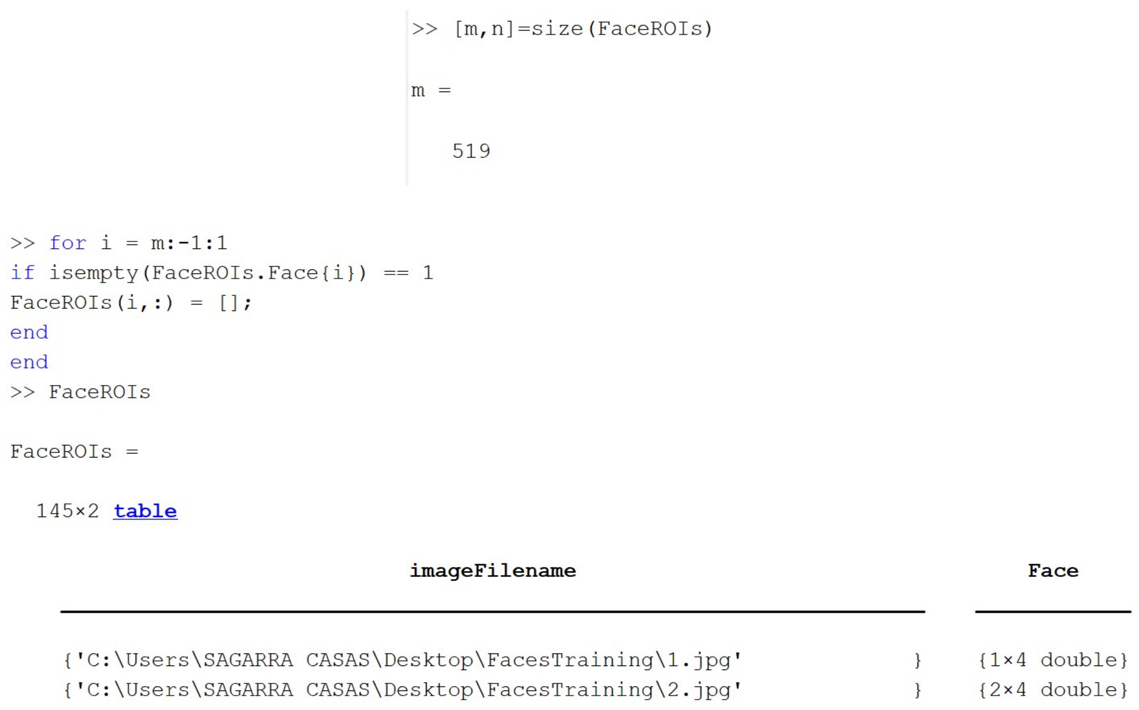
As it can be seen below, the first step is the elimination of the columns that don’t correspond to the selected class. We will first create the “FaceROIs” table.



```
>> FaceROIs = CompleteROIs;
>> FaceROIs(:,[3 4 5 6]) = [];
>> FaceROIs

FaceROIs =

  519×2 table

                          imageFilename                                  Face
    ______________________________________________________________    __________

    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\1.jpg'        }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'        }    {2×4 double}
```

**Figure 5.3.1.** Creation of the table for the training of the face detection network and elimination of the columns that are not of interest

Then, as the code shown below does, we proceed with the elimination of all those rows that don't contain any bounding box corresponding to the selected class.

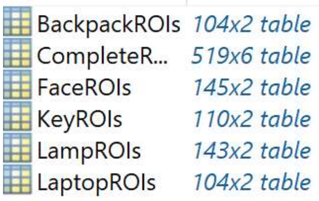


```
>> [m,n]=size(FaceROIs)

m =

   519
```

```
>> for i = m:-1:1
if isempty(FaceROIs.Face{i}) == 1
FaceROIs(i,:) = [];
end
end
>> FaceROIs

FaceROIs =

  145×2 table

                          imageFilename                                  Face
    ______________________________________________________________    __________

    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\1.jpg'        }    {1×4 double}
    {'C:\Users\SAGARRA CASAS\Desktop\FacesTraining\2.jpg'        }    {2×4 double}
```

**Figure 5.3.2.** Code that executes the elimination of the rows that are not of interest

Now that we have a table with the images and ROIs of faces, the same procedure can be done for the 4 remaining classes.



| | |
|---|---|
| BackpackROIs | 104x2 table |
| CompleteR... | 519x6 table |
| FaceROIs | 145x2 table |
| KeyROIs | 110x2 table |
| LampROIs | 143x2 table |
| LaptopROIs | 104x2 table |

**Figure 5.3.3.** Dimensions of the specific datasets for each object class

Now we train each neural network using each specific dataset.

The training processes of the key detection, lamp detection and backpack detection networks is presented in the *Annexes A2*, corresponding to the *Figure A.2.* , the *Figure A.3.* and the figure *Figure A.4* respectively.

Having tested the networks, we can divide the created NNs into two types:

- The networks trained to detect objects that have robust shapes and that tend to occupy bigger areas of the image, present good performances. However, they failure rate increases as the object occupies a smaller area of the picture.
  They work well with small images (with a height of 450 pixels), what makes them fast as less pixels to analyse result in less processing time.
  These networks are the ones that classify: faces, backpacks and laptops.

- The networks trained to detect objects that may have changing shapes depending on their orientation and on how they are placed in the environment, present irregular performances depending on the characteristics of the studied image.

  If the image is large (it is formed by a big number of pixels) and the object occupies a relatively large area of the image, the object is correctly detected although residual wrong detections are done (the wrong detections tend to have low scores). The next image shows an example of this case, where the score of the correctly detected keys is equal to 1 (which corresponds to a 100% accuracy) and there are erroneous detections.

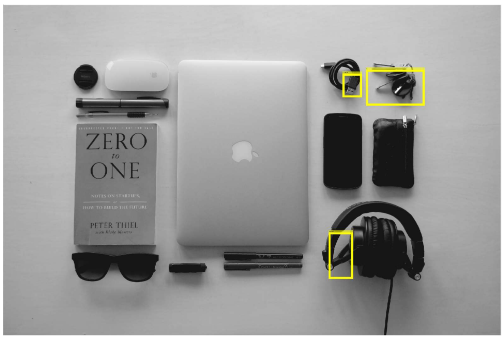


**Figure 5.3.4.** Analysis of an image with the previously mentioned features, with the key detection NN

If the image dimensions are small and the object occupies a relatively large area, the network performs in an unpredictable way. However, if the object occupies a sufficiently large area, it is correctly detected. The next image shows an example of this phenomenon, where the classification has been done incorrectly.

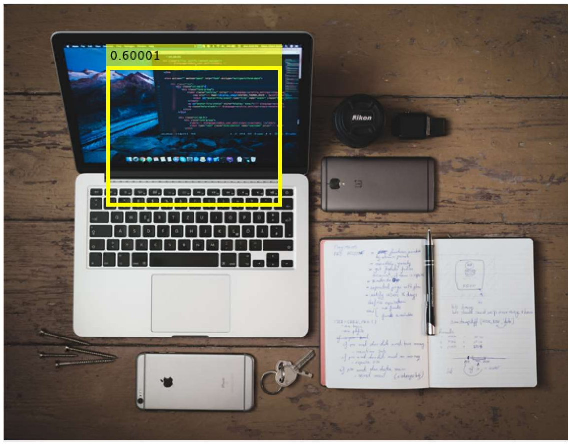


**Figure 5.3.5.** Analysis of an image with the previously mentioned features, with the key detection NN

If the area in the image that corresponds to the object is small, it is highly probable that the object is not recognized correctly. The next image shows the result of the analysis of an image where the searched object (which are keys) occupies a small area of the image. It can be observed that multiple objects that do not correspond to the class of interest are identified as false positives with a high incorrect score.

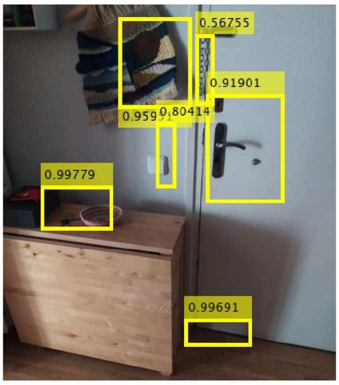


**Figure 5.3.6.** Analysis of an image with the previously mentioned features, with the key detection NN

These networks that present these behaviours are the ones that classify keys and lamps. As keys have really different shapes depending on their orientation and lamps present different forms depending on their design.

The created networks have serious problems detecting their objects of interest when the objects occupy small areas in the inspected images.

Instead of continuing performing changes in the training of the networks, we will perform changes in the method that the NNs use in order to analyse the images.

We will use these specific RCNNs from now on.

# 6. Subdividing images

As it has been observed in the previous section, when the objects to detect occupy small areas of the image, our trained RCNNs have problems recognizing them.

The next solution has been implemented: The subdivision of the images to inspect in 9 sub-images that are separately analysed by the corresponding RCNN. Therefore, in each image division objects appear bigger, occupying an area percentage of the image subdivision 9 times bigger than the area percentage they occupied in the original image. This results in a multiplication of the analysis accuracy by 9, although it also results in a computing time 9 times bigger.

The image is firstly resized to a height dimension of 1350 pixels, preserving its original dimensional relations between its height and width. This way, when dividing the image into 9 parts, each part will have a height of 450 pixels, which is the optimal height for the object detection with our RCNNs.

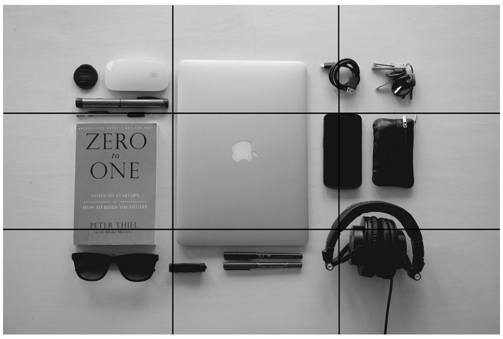


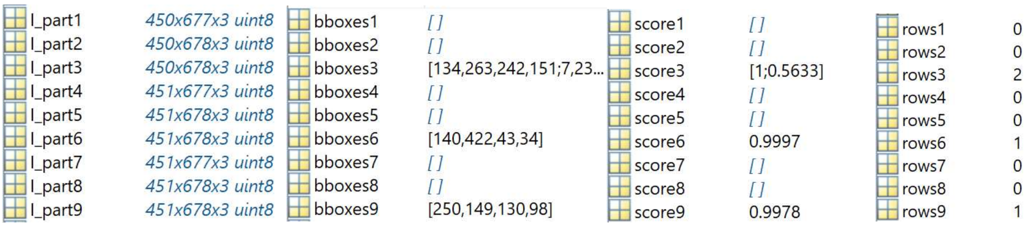
**Figure 6.1.** Division of an image in 9 parts

Moreover, if the searched object is found, it can be easily located just by relating it to the sub-image where it has been detected. We will name the top-left subdivision as the image_part1, the top-centre subdivision as the image_part2 and so on; being the bottom-centre subdivision the image_part8 and the bottom-right subdivision the image_part9.

However, this partition presents a good method for the detection of small objects as they may probably be located inside a specific subdivision, but presents limitations taking into account that sufficiently big objects will occupy pixels in multiple image parts and will therefore be detected as separate objects.

In the *Annexes A3* is presented the complete commented code that subdivides the image of interest and analyses each subdivision. The output of this code gives a result that describes the location or locations where the object that the user wants to search has been found. A possible location can be, for example, “bottom-left section of the image”.

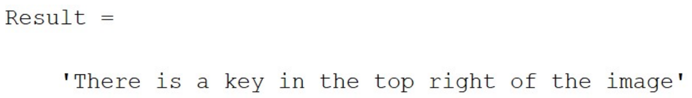
In the following figure are represented the bounding boxes and scores that have been extracted from the analysis of the 9 sub-images that form a specific image. The analysed image file name is “13KEY.jpg” and the used network for the analysis is the key detection network. On the left, the dimensions of each sub-image; on the centre left, the bounding boxes corresponding to detected keys in each sub-image; on the centre right, the scores related to each classification; on the right, the number of detections in each sub-image.



```
I_part1  450x677x3 uint8  bboxes1  []                     score1  []          rows1  0
I_part2  450x678x3 uint8  bboxes2  []                     score2  []          rows2  0
I_part3  450x678x3 uint8  bboxes3  [134,263,242,151;7,23...  score3  [1;0.5633]  rows3  2
I_part4  451x677x3 uint8  bboxes4  []                     score4  []          rows4  0
I_part5  451x678x3 uint8  bboxes5  []                     score5  []          rows5  0
I_part6  451x678x3 uint8  bboxes6  [140,422,43,34]        score6  0.9997      rows6  1
I_part7  451x677x3 uint8  bboxes7  []                     score7  []          rows7  0
I_part8  451x678x3 uint8  bboxes8  []                     score8  []          rows8  0
I_part9  451x678x3 uint8  bboxes9  [250,149,130,98]       score9  0.9978      rows9  1
```

**Figure 6.2.** Multiple variables from a segmented analysis of an image

A total of 4 objects are detected as keys in three different image sections. However, only detected objects with a greater accuracy score than 0.9999 are understood as correctly classified objects. The next figure shows the final result computed from the analysis above.

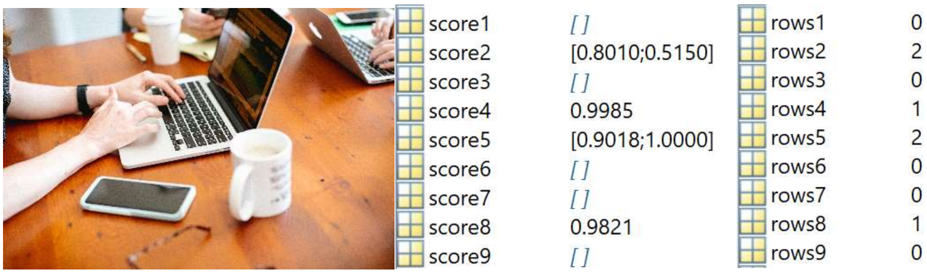


```
Result =

    'There is a key in the top right of the image'
```

**Figure 6.3.** Result computed from a segmented analysis of an image

The top right of the image corresponds to the third subdivision. The analysed image is the one shown in the *Figure 6.1.* **,** which has a key in the top right corner of it. The analysis has been successful.

The next figure shows the analysis results of another image, where laptops have been searched. On the left, the analysed image; on the centre the scores of the detected laptops in every sub-image; on the right, the number of the detected laptops in each sub-image.



```
score1  []                 rows1  0
score2  [0.8010;0.5150]    rows2  2
score3  []                 rows3  0
score4  0.9985             rows4  1
score5  [0.9018;1.0000]    rows5  2
score6  []                 rows6  0
score7  []                 rows7  0
score8  0.9821             rows8  1
score9  []                 rows9  0
```

**Figure 6.4.** Result computed from a segmented analysis of an image

The description result from the previous analysis is: “there is a laptop in the centre of the image”. The object detection model formed by the image segmentation and the RCNN has correctly detected and located a computer but has not detected a second computer present in the image. However, the approach to visually describe the environment and the detection of the searched object is satisfactory.

It is important to notice that the coordinates of the bounding boxes are not used in this method. However, before arriving to this analysis model, a faster but less accurate method was developed in order to detect and locate specific objects in images. The following is the process executed by this previous model in order to describe the location of detected objects in the analysed images:

- Compute the bounding boxes and scores of detected objects in an image.
- Filter the correctly detected objects as those that have score values greater than 0.9999.
- Find the gravity centres of the bounding boxes that contain each correctly detected object.
- Relate the coordinates of the gravity centres with the dimensions of the image. Finding the section of the image where each detected object is located.

This previous model also divides the possible location of the detected objects in 9 regions that equally divide the image. In the *Annexes A5* it is located the commented code that was implemented in the Matlab application program for the detection and location of objects in images, before developing the image subdivision method.

Through multiple empirical testing, we have concluded that the model that analyses segmented images is more accurate than the model that analyses complete images. Although the time that a segmented analysis takes to compute is of 15 seconds (in average), while the time that the complete image analysis takes to compute is of 8 seconds (in average), we have opted for the usage of the most accurate model, which is the image segmentation analysis model.

# 7. Matlab application program

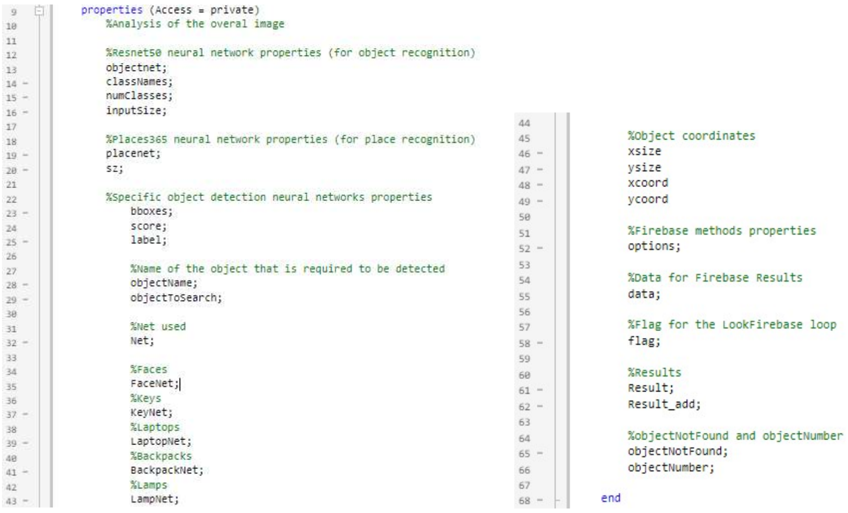
This chapter explains the functions of each part of the Matlab program that form the image analysis application that interacts with the Firebase in order to form the complete software together with the user interface programmed with Mit App Inventor.

The application is programmed using the Code View workspace of the Matlab App Designer. The program uses functions that are called during the execution of the application, which perform specific operations and have particular objectives. Functions are called by trigger events and other functions, creating a flow structure that defines the behaviour of the application.

The structure used to program this application is the one presented in the *Figure 3.2.2.* , where the architecture of the program is represented and explained. The functions of the program are interconnected as explained in this figure.

Each of the following paragraphs will correspond to the explanation and code of specific extracts and functions that together form the entirety of the Matlab program.

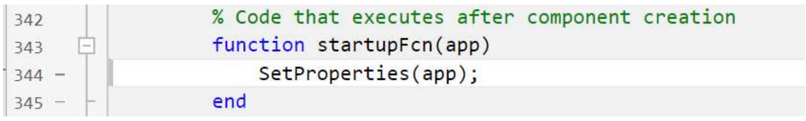
- The core of the computational tasks done by the program are performed by functions, which can be called between them and by the code itself. The first function has the objective of defining the global variables of the program. These variables are defined by just inserting their names into this private function (a private function can be accessed by the same program, but not by external programs). Global variables can be called and treated during the whole program execution, local variables can just be created and used inside a specific function. In order to call a global variable the characters "app." have to be inserted before writing the name of the variable.



```
    properties (Access = private)
        %Analysis of the overal image

        %Resnet50 neural network properties (for object recognition)
        objectnet;
        classNames;
        numClasses;
        inputSize;

        %Places365 neural network properties (for place recognition)
        placenet;
        sz;

        %Specific object detection neural networks properties
            bboxes;
            score;
            label;

            %Name of the object that is required to be detected
            objectName;
            objectToSearch;

            %Net used
            Net;

            %Faces
            FaceNet;
            %Keys
            KeyNet;
            %Laptops
            LaptopNet;
            %Backpacks
            BackpackNet;
            %Lamps
            LampNet;

            %Object coordinates
            xsize
            ysize
            xcoord
            ycoord

            %Firebase methods properties
            options;

            %Data for Firebase Results
            data;

            %Flag for the LookFirebase loop
            flag;

            %Results
            Result;
            Result_add;

            %objectNotFound and objectNumber
            objectNotFound;
            objectNumber;

    end
```

**Figure 7.1.** Global variables definition function

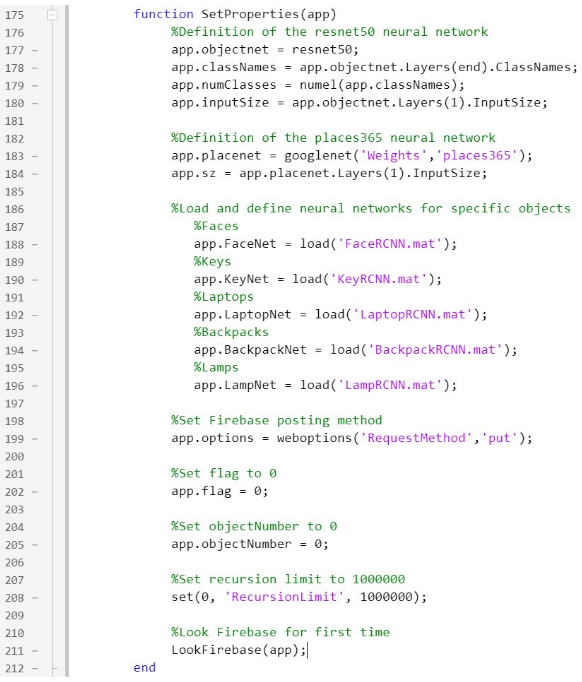
- When the application is opened the first executed function is, by default, the startupFcn(app), which calls the SetProperties(app) function.



```
        % Code that executes after component creation
        function startupFcn(app)
            SetProperties(app);
        end
```

**Figure 7.2.** Start-up function

- The SetProperties(app) configures the parameters of the deep neural networks used during the program, sets the Firebase posting method as ‘put’, initializes de value of some of the global variables and sets the value of the maximum number of times a same function can be called without clashing to 1000000. Finally this function calls the LookFirebase(app) function.



```
        function SetProperties(app)
            %Definition of the resnet50 neural network
            app.objectnet = resnet50;
            app.classNames = app.objectnet.Layers(end).ClassNames;
            app.numClasses = numel(app.classNames);
            app.inputSize = app.objectnet.Layers(1).InputSize;

            %Definition of the places365 neural network
            app.placenet = googlenet('Weights','places365');
            app.sz = app.placenet.Layers(1).InputSize;

            %Load and define neural networks for specific objects
               %Faces
               app.FaceNet = load('FaceRCNN.mat');
               %Keys
               app.KeyNet = load('KeyRCNN.mat');
               %Laptops
               app.LaptopNet = load('LaptopRCNN.mat');
               %Backpacks
               app.BackpackNet = load('BackpackRCNN.mat');
               %Lamps
               app.LampNet = load('LampRCNN.mat');

            %Set Firebase posting method
            app.options = weboptions('RequestMethod','put');

            %Set flag to 0
            app.flag = 0;

            %Set objectNumber to 0
            app.objectNumber = 0;

            %Set recursion limit to 1000000
            set(0, 'RecursionLimit', 1000000);

            %Look Firebase for first time
            LookFirebase(app);
        end
```

**Figure 7.3.** Properties initialization function

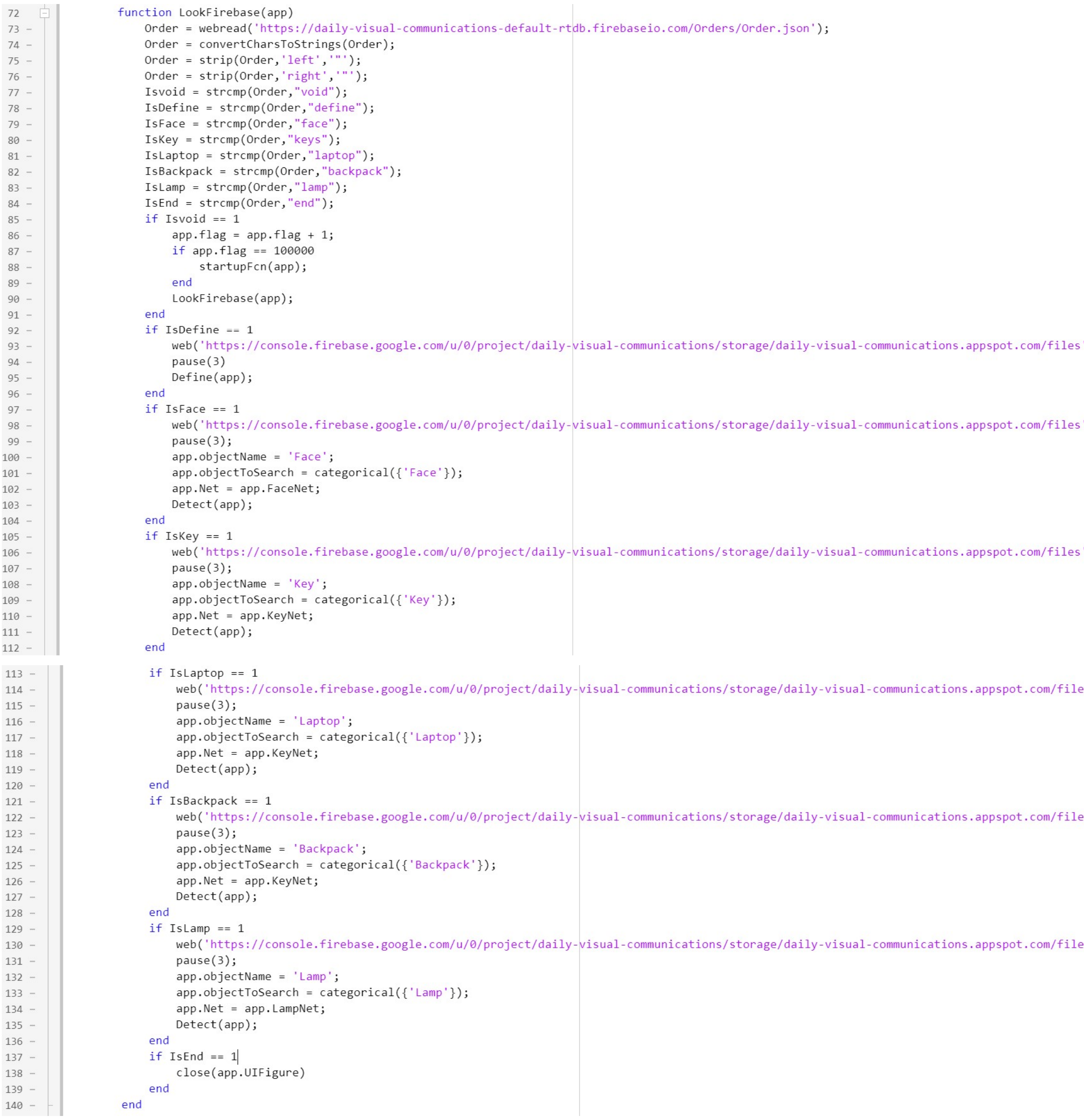
- LookFirebase(app) reads the Order value from the Firebase and eliminates its “ characters that are concatenated to the order when it is uploaded to the database by the smartphone program. Then it compares the Order with the specific words and when the word that corresponds to the Order is found a specific process is carried out.

  If the Order is equal to “void” the LookFirebase(app) is called again.

  If the Order is equal to “define” the storage database website is loaded and a pause of 3 seconds is executed in order to ensure the image has been successfully uploaded to the database. Then the Define(app) function is called.

  If the order is equal to one of the possible object classes to search, the storage database website is loaded and a pause of 3 seconds is executed in order to ensure the image has been successfully uploaded to the database. Then a set of global variables are defined as the name of the object to search and the RCNN corresponding to the detection of the specific object is defined as the operating neural network. Finally the Detect(app) function is called.

If the order is equal to “end” the program is shut down. This allows the user to stop the execution of the Matlab program although he is not near the computer.



```
function LookFirebase(app)
    Order = webread('https://daily-visual-communications-default-rtdb.firebaseio.com/Orders/Order.json');
    Order = convertCharsToStrings(Order);
    Order = strip(Order,'left','"');
    Order = strip(Order,'right','"');
    Isvoid = strcmp(Order,"void");
    IsDefine = strcmp(Order,"define");
    IsFace = strcmp(Order,"face");
    IsKey = strcmp(Order,"keys");
    IsLaptop = strcmp(Order,"laptop");
    IsBackpack = strcmp(Order,"backpack");
    IsLamp = strcmp(Order,"lamp");
    IsEnd = strcmp(Order,"end");
    if Isvoid == 1
        app.flag = app.flag + 1;
        if app.flag == 100000
            startupFcn(app);
        end
        LookFirebase(app);
    end
    if IsDefine == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/files
        pause(3)
        Define(app);
    end
    if IsFace == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/files
        pause(3);
        app.objectName = 'Face';
        app.objectToSearch = categorical({'Face'});
        app.Net = app.FaceNet;
        Detect(app);
    end
    if IsKey == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/files
        pause(3);
        app.objectName = 'Key';
        app.objectToSearch = categorical({'Key'});
        app.Net = app.KeyNet;
        Detect(app);
    end
    if IsLaptop == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/file
        pause(3);
        app.objectName = 'Laptop';
        app.objectToSearch = categorical({'Laptop'});
        app.Net = app.KeyNet;
        Detect(app);
    end
    if IsBackpack == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/file
        pause(3);
        app.objectName = 'Backpack';
        app.objectToSearch = categorical({'Backpack'});
        app.Net = app.KeyNet;
        Detect(app);
    end
    if IsLamp == 1
        web('https://console.firebase.google.com/u/0/project/daily-visual-communications/storage/daily-visual-communications.appspot.com/file
        pause(3);
        app.objectName = 'Lamp';
        app.objectToSearch = categorical({'Lamp'});
        app.Net = app.LampNet;
        Detect(app);
    end
    if IsEnd == 1
        close(app.UIFigure)
    end
end
```

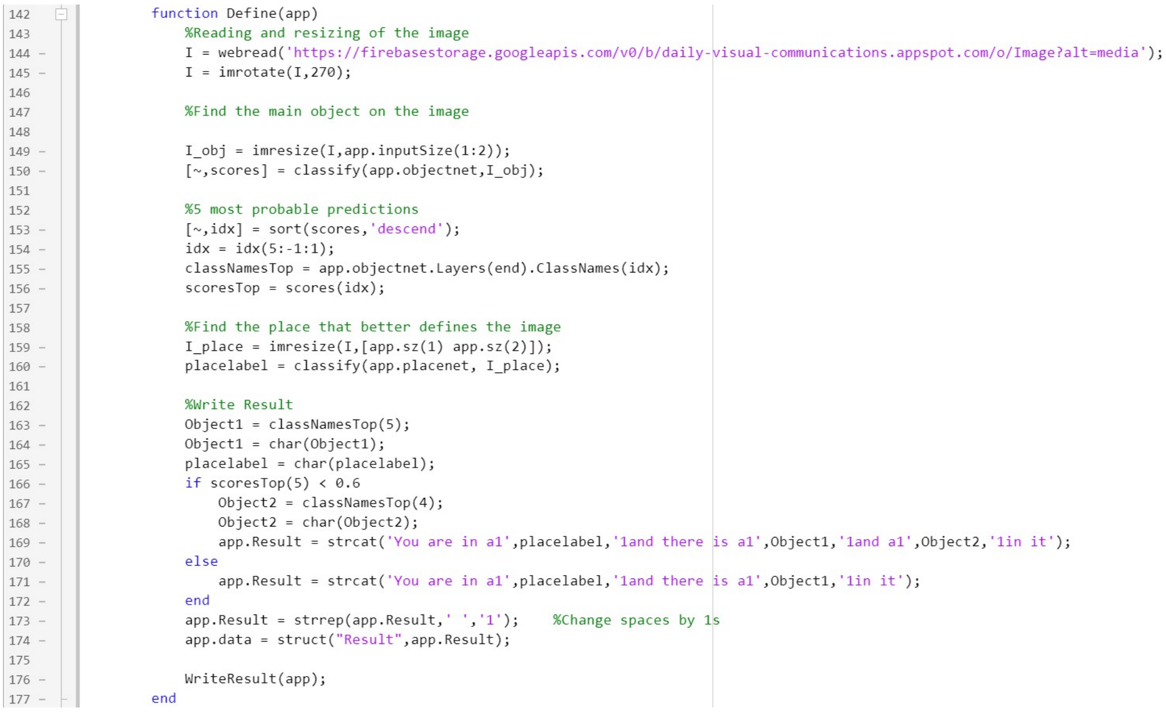
**Figure 7.4.** LookFirebase(app) function

- The Define(app) function computes the program, previously explained, for the general analysis of the image. The image is firstly downloaded from the Firebase Storage database using the “webread” command, then it is treated and analysed by the object and place classification neural networks. Subsequently, the function stores the result of the description of the image as the data that has to be uploaded to the Firebase Realtime Database. Finally, the WriteResult(app) function is called.

Notice that instead of spaces, there are ones “1s” between the words that form the result that is uploaded to the Firebase. This is because spaces cannot be uploaded to the Firebase, so they are replaced by ones before storing the result in the database. Later, “1s” are replaced by spaces by the smartphone application. This method can recall an encrypting-decrypting method.

If the main object found in the image has an accuracy value equal or greater than 60%, it is presented as the unique representative object in the image.

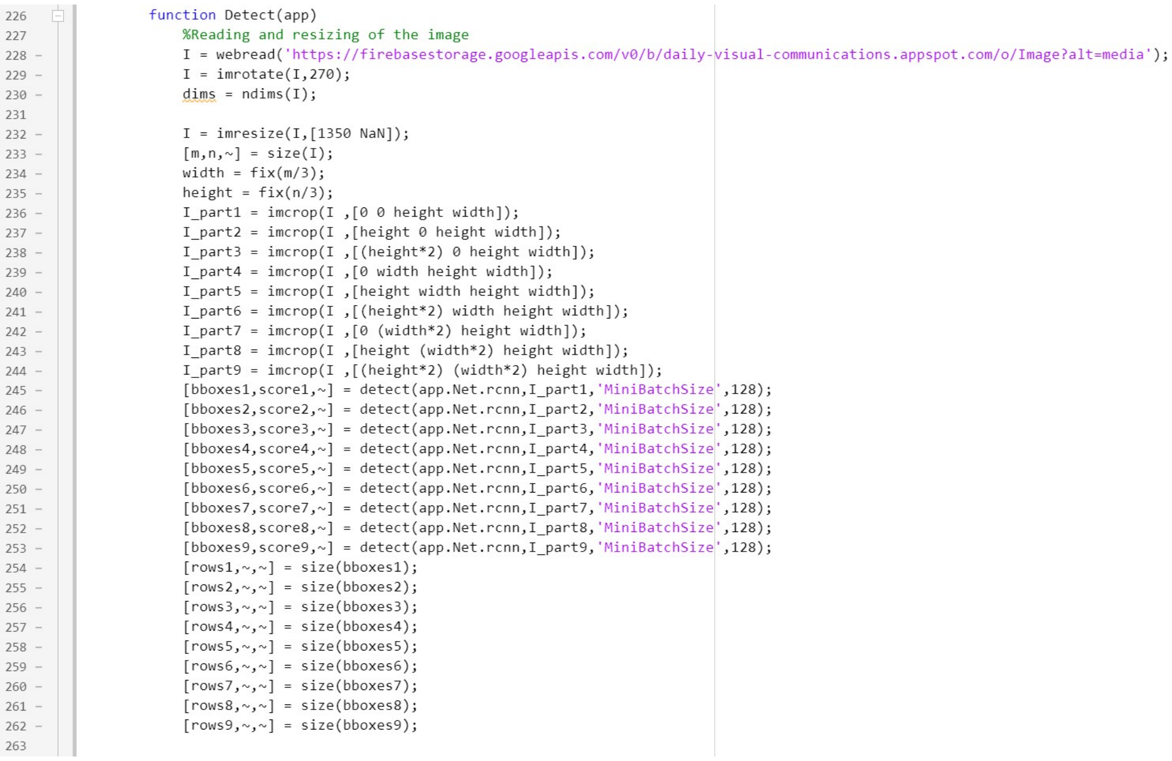
If the main object found in the image has an accuracy value lower than 60%, the two most representative objects are introduced to the result as objects contained in the inspected image.



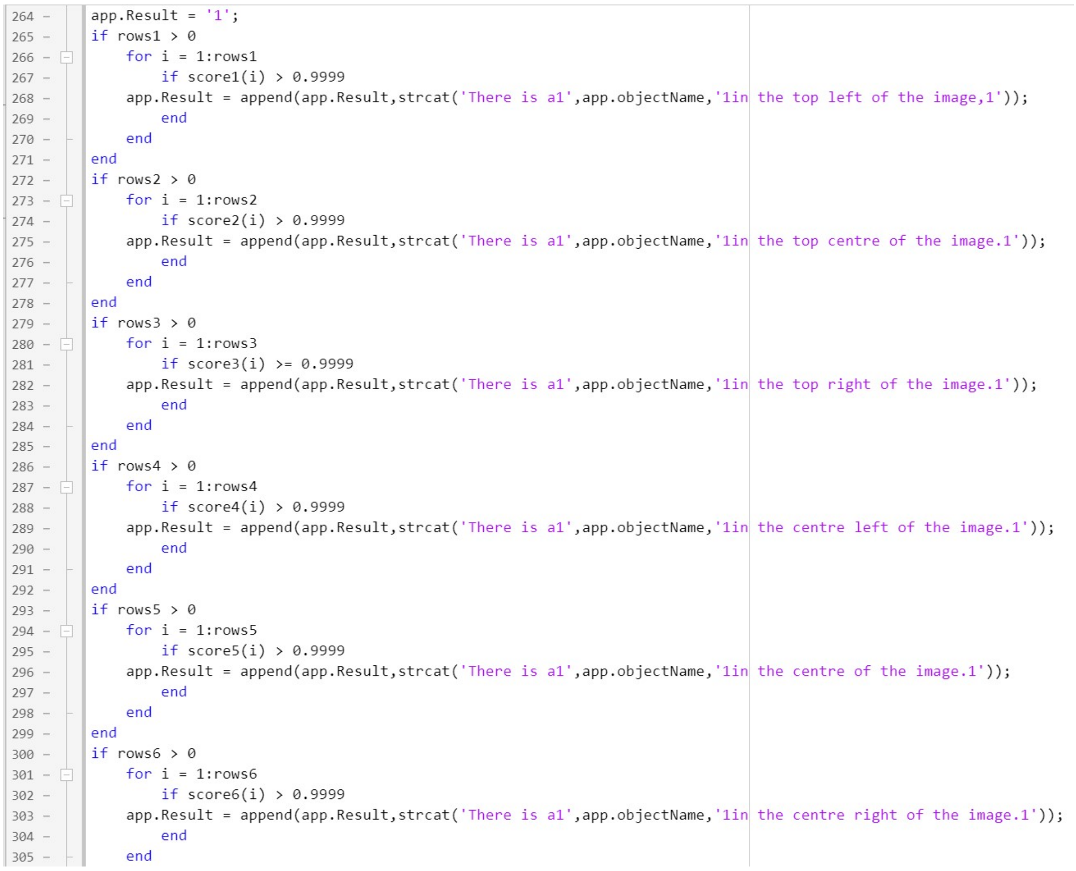
```
function Define(app)
    %Reading and resizing of the image
    I = webread('https://firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media');
    I = imrotate(I,270);

    %Find the main object on the image

    I_obj = imresize(I,app.inputSize(1:2));
    [~,scores] = classify(app.objectnet,I_obj);

    %5 most probable predictions
    [~,idx] = sort(scores,'descend');
    idx = idx(5:-1:1);
    classNamesTop = app.objectnet.Layers(end).ClassNames(idx);
    scoresTop = scores(idx);

    %Find the place that better defines the image
    I_place = imresize(I,[app.sz(1) app.sz(2)]);
    placelabel = classify(app.placenet, I_place);

    %Write Result
    Object1 = classNamesTop(5);
    Object1 = char(Object1);
    placelabel = char(placelabel);
    if scoresTop(5) < 0.6
        Object2 = classNamesTop(4);
        Object2 = char(Object2);
        app.Result = strcat('You are in a1',placelabel,'1and there is a1',Object1,'1and a1',Object2,'1in it');
    else
        app.Result = strcat('You are in a1',placelabel,'1and there is a1',Object1,'1in it');
    end
    app.Result = strrep(app.Result,' ','1');    %Change spaces by 1s
    app.data = struct("Result",app.Result);

    WriteResult(app);
end
```

**Figure 7.5.** General image analysis function

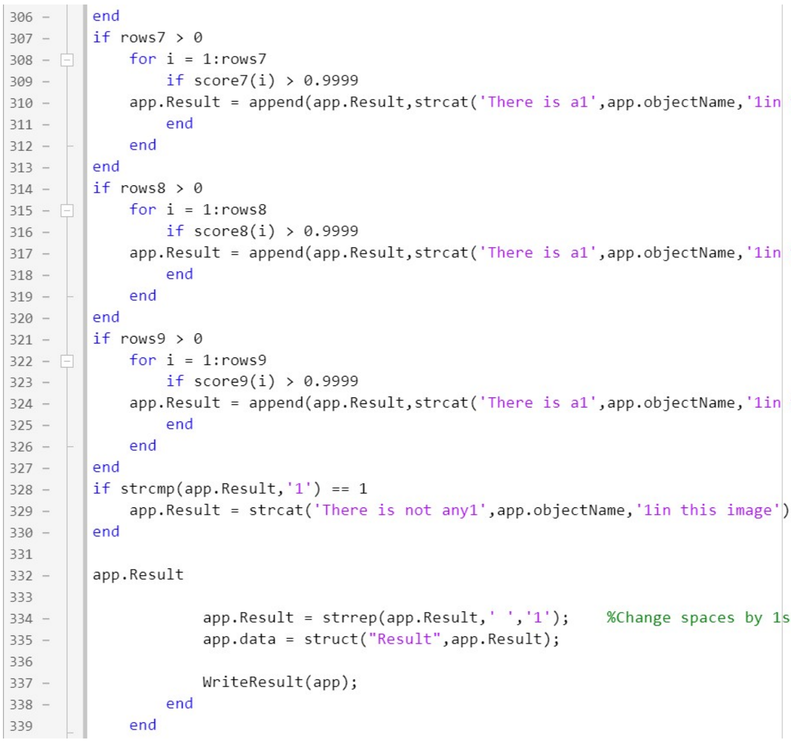
- The Detect(app) function has the same general structure than the Define(app): Downloading of the image, treatment and analysis of it (although this time a specific object is meant to be detected and located with a particular detection RCNN, which has been defined in the LookFirease(app) depending on the object to search), generation of a Result string and the finally calling of the WriteResult(app). The code corresponding to this function is explained in the *Annexes A3*.



```
    function Detect(app)
        %Reading and resizing of the image
        I = webread('https://firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media');
        I = imrotate(I,270);
        dims = ndims(I);

        I = imresize(I,[1350 NaN]);
        [m,n,~] = size(I);
        width = fix(m/3);
        height = fix(n/3);
        I_part1 = imcrop(I ,[0 0 height width]);
        I_part2 = imcrop(I ,[height 0 height width]);
        I_part3 = imcrop(I ,[(height*2) 0 height width]);
        I_part4 = imcrop(I ,[0 width height width]);
        I_part5 = imcrop(I ,[height width height width]);
        I_part6 = imcrop(I ,[(height*2) width height width]);
        I_part7 = imcrop(I ,[0 (width*2) height width]);
        I_part8 = imcrop(I ,[height (width*2) height width]);
        I_part9 = imcrop(I ,[(height*2) (width*2) height width]);
        [bboxes1,score1,~] = detect(app.Net.rcnn,I_part1,'MiniBatchSize',128);
        [bboxes2,score2,~] = detect(app.Net.rcnn,I_part2,'MiniBatchSize',128);
        [bboxes3,score3,~] = detect(app.Net.rcnn,I_part3,'MiniBatchSize',128);
        [bboxes4,score4,~] = detect(app.Net.rcnn,I_part4,'MiniBatchSize',128);
        [bboxes5,score5,~] = detect(app.Net.rcnn,I_part5,'MiniBatchSize',128);
        [bboxes6,score6,~] = detect(app.Net.rcnn,I_part6,'MiniBatchSize',128);
        [bboxes7,score7,~] = detect(app.Net.rcnn,I_part7,'MiniBatchSize',128);
        [bboxes8,score8,~] = detect(app.Net.rcnn,I_part8,'MiniBatchSize',128);
        [bboxes9,score9,~] = detect(app.Net.rcnn,I_part9,'MiniBatchSize',128);
        [rows1,~,~] = size(bboxes1);
        [rows2,~,~] = size(bboxes2);
        [rows3,~,~] = size(bboxes3);
        [rows4,~,~] = size(bboxes4);
        [rows5,~,~] = size(bboxes5);
        [rows6,~,~] = size(bboxes6);
        [rows7,~,~] = size(bboxes7);
        [rows8,~,~] = size(bboxes8);
        [rows9,~,~] = size(bboxes9);

```



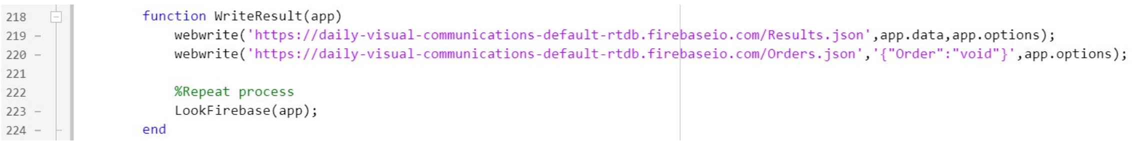
```
app.Result = '1';
if rows1 > 0
    for i = 1:rows1
        if score1(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the top left of the image,1'));
        end
    end
end
if rows2 > 0
    for i = 1:rows2
        if score2(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the top centre of the image.1'));
        end
    end
end
if rows3 > 0
    for i = 1:rows3
        if score3(i) >= 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the top right of the image.1'));
        end
    end
end
if rows4 > 0
    for i = 1:rows4
        if score4(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the centre left of the image.1'));
        end
    end
end
if rows5 > 0
    for i = 1:rows5
        if score5(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the centre of the image.1'));
        end
    end
end
if rows6 > 0
    for i = 1:rows6
        if score6(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the centre right of the image.1'));
        end
    end
```



```
end
if rows7 > 0
    for i = 1:rows7
        if score7(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the bottom left of the image.1'));
        end
    end
end
if rows8 > 0
    for i = 1:rows8
        if score8(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the bottom centre of the image.1'));
        end
    end
end
if rows9 > 0
    for i = 1:rows9
        if score9(i) > 0.9999
    app.Result = append(app.Result,strcat('There is a1',app.objectName,'1in the bottom right of the image.1'));
        end
    end
end
if strcmp(app.Result,'1') == 1
    app.Result = strcat('There is not any1',app.objectName,'1in this image');
end

app.Result

            app.Result = strrep(app.Result,' ','1');    %Change spaces by 1s
            app.data = struct("Result",app.Result);

            WriteResult(app);
        end
    end
```

**Figure 7.6.** Specific object detection and location function

- The WriteResult(app) uploads the result generated for the Define(app) or the Detect(app) functions to the Realtime Database and changes the value of the Order in the database by “void”. Finally, the LookFirebase(app) function is called in order to repeat the cycle.



```
function WriteResult(app)
    webwrite('https://daily-visual-communications-default-rtdb.firebaseio.com/Results.json',app.data,app.options);
    webwrite('https://daily-visual-communications-default-rtdb.firebaseio.com/Orders.json','{"Order":"void"}',app.options);

    %Repeat process
    LookFirebase(app);
end
```

**Figure 7.7.** WriteResult(app) function

The Matlab program, once is fully developed, is compiled and provided to the user as an executable application.

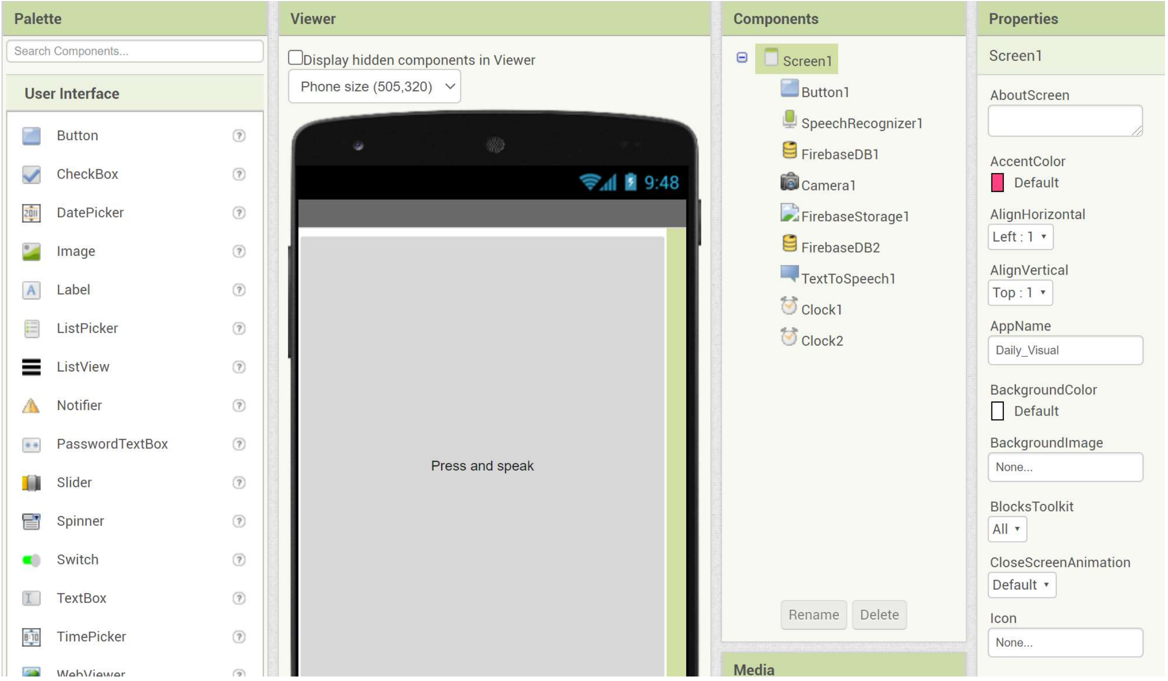
# 8. Mit App Inventror program

This chapter explains the functions of each part of the Mit App Inventor program that form the user – software interface application that interacts with the Firebase in order to form the complete software together with the image analysis programed developed on Matlab.

As explained in the chapter *3.4 Mit App Inventor*, this programming language uses visual blocks that are interconnected in order to form processes. The interconnections and functionalities of these blocks are developed in the “Blocks” workspace, which allows the programming of the processes that form the application. The design of the interface understanding it as the way elements are shown and interact in the screen is done in the “Designer” workspace.

The structure used to program this application is the one presented in the *Figure 3.2.1.* , where the architecture of the program is represented and explained. The blocks and processes of the program are interconnected as explained in this figure.

The following figure corresponds to the selection of the elements that form the application and the design of their interactions in the screen. As it can be observed, the totality of the screen is occupied by a single button. All the dialog between the user and the application will be made through this button and voice operated commands.



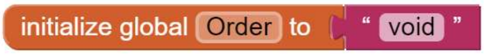
Components:
- Screen1
  - Button1
  - SpeechRecognizer1
  - FirebaseDB1
  - Camera1
  - FirebaseStorage1
  - FirebaseDB2
  - TextToSpeech1
  - Clock1
  - Clock2

Properties: Screen1
- AboutScreen:
- AccentColor: Default
- AlignHorizontal: Left : 1
- AlignVertical: Top : 1
- AppName: Daily_Visual
- BackgroundColor: Default
- BackgroundImage: None...
- BlocksToolkit: All
- CloseScreenAnimation: Default
- Icon: None...

**Figure 8.1.** Designer view of the application development environment

Each of the following paragraphs corresponds to the explanation and visual representation of processes that form the fully operational smartphone application.

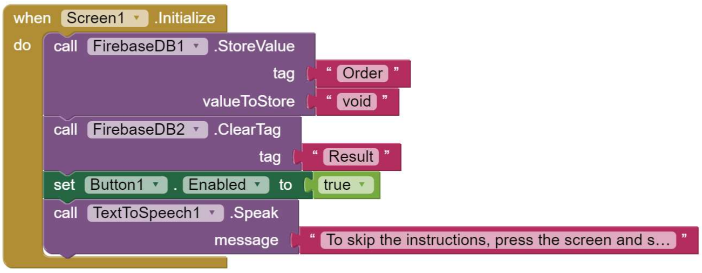
- The only global variable we will use is the Order that is to be uploaded to the Firebase. We firstly define this variable.



```
initialize global Order to " void "
```

**Figure 8.2.** Definition of the global variable "Order"

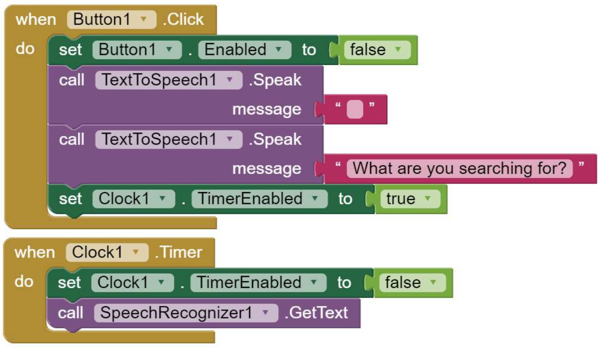
- The following blocks are the executed when the application is initialized. The order in the Firebase is set to "void" and the Result tag is eliminated. The main button of the screen is enabled so it can operate and a text-to-speech block communicates the instructions of the app to the user. The user can skip the instructions and start using the app by pressing the screen button.



```
when Screen1 .Initialize
do  call FirebaseDB1 .StoreValue
                        tag  " Order "
               valueToStore  " void "
    call FirebaseDB2 .ClearTag
                        tag  " Result "
    set Button1 . Enabled to  true
    call TextToSpeech1 .Speak
                    message  " To skip the instructions, press the screen and s… "
```

**Figure 8.3.** Initialization blocks

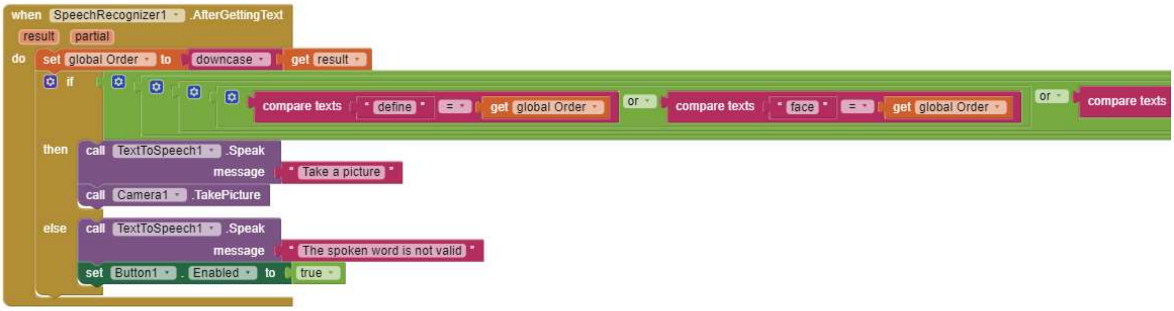
- When the button is pressed (if it is enabled), the next blocks are executed. The screen button is disabled and the message "What are you searching for" is communicated to the user. Then a clock is fired in order to compute a pause of 2 seconds to let the previous message be communicated. After this pause, the speech recognition is executed in order to transcribe the order from the user.



```
when Button1 .Click
do  set Button1 . Enabled to  false
    call TextToSpeech1 .Speak
                    message  "  "
    call TextToSpeech1 .Speak
                    message  " What are you searching for? "
    set Clock1 . TimerEnabled to  true

when Clock1 .Timer
do  set Clock1 . TimerEnabled to  false
    call SpeechRecognizer1 .GetText
```

**Figure 8.4.** Blocks that get the order from the user

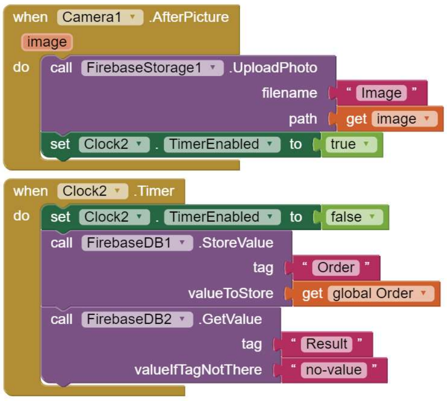
- If the order corresponds to one of correct order commands, the camera is opened in order to let the user take a picture. If the order is not correct, a warning message is communicated to the user and the program returns to the previous explained blocks corresponding to the *Figure 8.4.* The order characters are set to lowercase in order to analyse it easier.



```
when SpeechRecognizer1 .AfterGettingText
  result  partial
do  set global Order to downcase get result
    if  compare texts " define " = get global Order  or  compare texts " face " = get global Order  or  compare texts
    then  call TextToSpeech1 .Speak
                  message " Take a picture "
          call Camera1 .TakePicture
    else  call TextToSpeech1 .Speak
                  message " The spoken word is not valid "
          set Button1 . Enabled to true
```

**Figure 8.5.** Blocks that perform the analysis of the order and open the camera

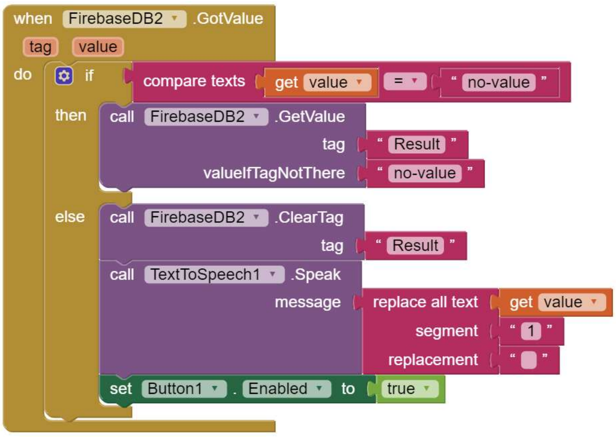
- The next blocks upload the taken image to the Storage database once the user has taken the picture and perform a pause of 4 seconds in order to ensure that the image has been correctly uploaded. Then, the word corresponding to the Order is written into the Firebase Realtime database and a block that reads the value of the Result from the firebase is executed. If the Result is not found in the database, the word "no-value" is assigned as the Result value.



```
when Camera1 .AfterPicture
  image
do  call FirebaseStorage1 .UploadPhoto
                   filename " Image "
                       path get image
    set Clock2 . TimerEnabled to true

when Clock2 .Timer
do  set Clock2 . TimerEnabled to false
    call FirebaseDB1 .StoreValue
                        tag " Order "
               valueToStore get global Order
    call FirebaseDB2 .GetValue
                        tag " Result "
          valueIfTagNotThere " no-value "
```

**Figure 8.6.** Blocks that upload the image and Order to the Firebase and that read the Result

- The next block set evaluates the read Result variable from the database. If the value is equal to "no-value" it means that the Result has not been yet posted to the database and therefore, the block that reads the Result from the firebase is executed again. If the Result value is not equal to "no-value" it means that the Result was successfully posted to the Firebase and downloaded by the read block. In this case, the Result is again eliminated from the database and aurally communicated to the user. Finally the button is again enabled in order to wait for the user to order the analysis of another image.



```
when FirebaseDB2 .GotValue
  tag  value
do  if  compare texts  get value  =  " no-value "
    then  call FirebaseDB2 .GetValue
                         tag  " Result "
           valueIfTagNotThere  " no-value "
    else  call FirebaseDB2 .ClearTag
                         tag  " Result "
          call TextToSpeech1 .Speak
                     message  replace all text  get value
                                        segment  " 1 "
                                    replacement  "   "
          set Button1 . Enabled to  true
```

**Figure 8.7.** Blocks that evaluate the Result read from the Firebase and, in case it exists, communicate it to the user

The program .apk file can be extracted from the same Mit App Inventor workspace, obtaining the complete functional and integrated application. So the smartphone application can be provided to the user.

# 9. Results

## 9.1. Analysis of the Neural Networks performances

The analysis of the performances of each developed and trained neural network has been done as the networks have been developed. Thus, methods in order to improve their accuracy have been designed and implemented during the development of this project.

### 9.1.1. About the errors in RCNNs

A false negative means that the network does a null detection (not detecting any object in the image) when the object is present in the analysed image. A false positive means that the network detects and classifies an object as the interest object when the object is not present in the analysed image. A location error occurs when the object is detected in the image but wrongly detected.

The next table represents the number of false negatives, false positives and location errors analysis performed by each of the five RCNNs over a training set of 40 images, where 30 images contained the corresponding objects and 10 did not, for each dataset. The percentage is calculated as following:

- $False\ positives\ (\%) = \frac{Number\ of\ false\ positives}{10} \cdot 100$ (Eq. 9.1)

  This number represents the percentage of images that do not contain the object of interest which are erroneously analysed.

- $False\ negatives\ (\%) = \frac{Number\ of\ false\ negatives}{30} \cdot 100$ (Eq. 9.2)
- $Location\ errors\ (\%) = \frac{Number\ of\ location\ errors}{30} \cdot 100$ (Eq. 9.3)

  Adding the false negatives percentage and location errors percentage, we get the percentage of images that contain the object of interest which are correctly analysed.

The analysis of the images have been done with the RCNN and the previous treatment of the analysed images with the method of image subdivision in 9 parts. Therefore, it is important to take into account that the performance accuracy values obtained during the testing of the RCNNs correspond to the accuracy values of the whole image subdivision – RCNNs methods.

| | Faces RCNN | Keys RCNN | Lamps RCNN | Laptops RCNN | Backpacks RCNN |
|---|---|---|---|---|---|
| False positives | 2 (20%) | 7 (70%) | 5 (50%) | 4 (40%) | 7 (70%) |
| False negatives | 8 (26.7%) | 1 (3.33%) | 6 (20%) | 10 (33.3%) | 3 (10%) |
| Location errors | 7 (23.3%) | 8 (26.7%) | 2 (6.67%) | 9 (30%) | 10 (33.3%) |

**Figure 9.1.** Number and percentages of wrongly analysed image and their error type

It can be clearly observed in the obtained results from the testing that our trained RCNNs tend to perform much more false positives than false negatives and location errors.

The next figure shows some of the test images used to evaluate the performance of the key RCNN.

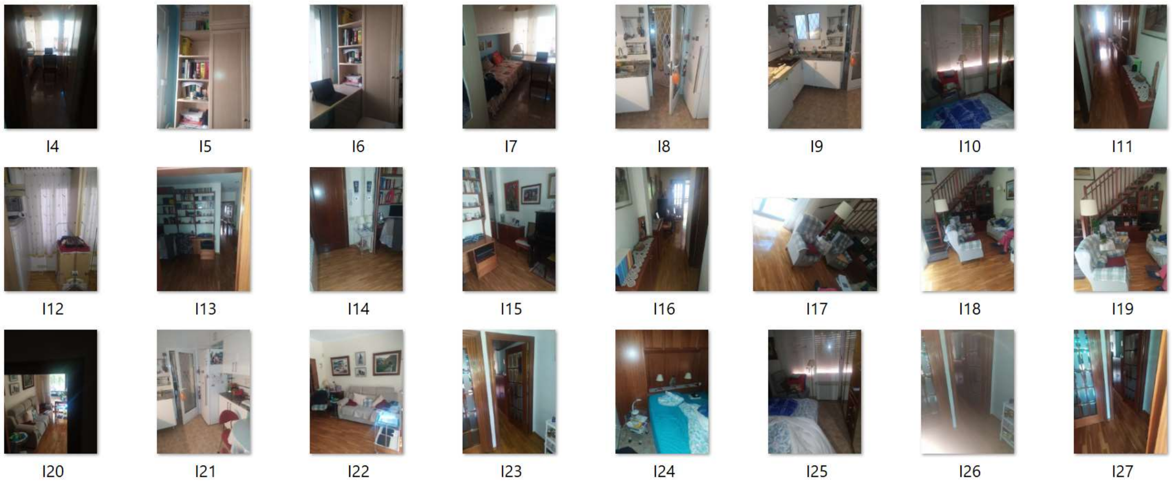


**Figure 9.1.** Sample of the test dataset for key detection RCNN

For each network, its accuracy is calculated as follows:

$$Network\ Accuracy = 100 - \frac{FalsePositives(\%) + FalseNegative\ \ (\%) + LocationErrors(\%)}{3} \qquad \text{(Eq. 9.4)}$$

The next table shows the accuracy that each network has performed in the testing process.

| | Faces RCNN | Keys RCNN | Lamps RCNN | Laptops RCNN | Backpacks RCNN |
|---|---|---|---|---|---|
| Accuracy (%) | 76.7 | 66.7 | 74.4 | 65.6 | 62.2 |

**Figure 9.3.** Accuracy of each RCNN, using the image subdivision method

The average accuracy of the trained RCNNs is a 69.1%.

### 9.1.2. About the accuracy of the pretrained ResNet-50 network and the pretrained GoogleNet network

These networks do not have neither false positives nor false positives. They simply correctly classify an image into the place that best describes it or not.

A test dataset with 60 images corresponding to different domestic indoor spaces has been used to test the accuracy of this two networks.

For each of the networks the performance results have been the following:

| | ResNet50 | GoogleNet |
|---|---|---|
| Accuracy (%) | 73.4 | 61.9 |

**Figure 9.3.** Accuracy of ResNet50 and GoogleNet pretrained CNNS

The larger part of the erroneous classifications correspond to descriptions that can be greatly related to the features and characteristics of the actual analysed space. Being this two neural networks, in terms of usefulness to the user, better than the RCNNs for specific object detection and location.

### 9.1.3. About the image analysis computing time

Each time the user performs an analysis of his or her environment with the software developed in this project, he gets a response. This response is highly accurate in the general description of an environment, making the user capable of knowing where he is, consequently improving his or her mobility in new or unknown surroundings.

In the computer where this project has been developed and with the Internet connections of my working place, general description analysis of an image tends to last a time lapse between 10 and 14 seconds. The analysis time in general descriptions of images tends to be 2 to 4 seconds lower than in specific object detections and locations. The CPU of the computer where this project has been developed is an IntelCore i5-4300U.

The time each image analysis takes to compute is directly dependent on the speed of the CPU and the GPU of the computer, as well as its RAM capacity and other efficiency parameters. The time between the user’s order and the Result communication also depends in the speed of the internet communications. The computation time can be dropped to 6 seconds in computers with highly efficient GPUs and can increase drastically in computers with small GPUs or in weak Internet connections.

## 9.2. Interview to experts

In order to determine whether the application is really useful for people with visual impairment and in what extent, the functionality and performance of the developed software in this project has been analysed and discussed by Felipe Yagüe and Julian Bruscantini, which are experts on methodologies for people with visual impairment and have visual impairment.

I have gone to the association of support to people with visual impairment “Associació Discapacitat Visual Catalunya” and interviewed Felipe Yagüe and Julian Bruscantiny, who currently work on the association. The following is an analysis of the results presented as the points that the two experts have discussed about the utility and performance of the application.

Felipe Yagüe is a psychologist and a psychotherapist graduated in psychology in the “Universitat de Barcelona” (UB) and with the master’s degree “Special needs of students with visual impairment” in the U.N.E.D. He is specialized in adjustment to low vision and blindness and from 2005 practises as a psychologist in the “Associació Discapacitat Visual Catalunya”. He has severe visual difficulties, conserving the 10% of his peripheral vision.

Julian Bruscantini is a communicator and a social trainer specialized in visual impairment. He has a large trajectory helping people with visual difficulties adapt to the society and practises as a trainer and educator in the “Associació Discapacitat Visual Catalunya”. He has a visual impairment, conserving a little part of his peripheral vision.

“If a person with visual impairment is in a place that is not familiar, it is highly probable that he or she prefers to be aware of what surrounds him or her by using a mobile phone application that takes 20 seconds to do the description rather than by touching the unknown objects that surrounds him or her.” (Felipe Yagüe).

Julian Bruscantini and Felipe Yagüe agree that the general description mode for the analysis of environments is useful. However, sometimes a room with paintings in it can be classified as a museum and a room with tools in it can be classified as a garage. This classifications are effectively wrong but still let the user know what type of objects he or she may find in the analysed environment, as the place classifier network bases its classifications on the multiple objects and features that form the inspected space. Thus, independently of being the classification wrong or correct, the general description mode usually gives an accurate definition of the environment.

About the detection and location of specific objects, Julian Bruscantini says, the user of the application should be familiarized with it (in other words, having used it a sufficiently large amount of time) in order to develop ease, confidence and accuracy in its usage. "Despite a 30% of the times an analysis is done the application tells you that the object you were searching for is in the image although it is not there or wrongly locates it in a region of the image that does not correspond to the actual location of the object, it really helps you be aware of what type of objects are located in the environment you took the photo of. This is because although a laptop may be detected in a place where there is not any laptop, the user rapidly knows that in this place there may be an object that has similar features than a laptop (like a folder or a tablet device), resulting in a consequently awareness of the type of objects he might have in front. However this demands a familiarization of the user with the behaviour of the application" (Julian Bruscantini).

"It is true that sometimes if the object does not occupy a sufficiently big area in the taken picture, it might probably not be detected. However, let's say there are 4 backpacks in an environment that you want to inspect. Only that the application tells you there is one backpack or an object that recalls a backpack, lets you know that in front of you there is a type of object that can probably be a backpack, which may be a group of backpacks or a single one. Moreover, if you previously did a general analysis of a place and it communicated to you that you were in a work office, and you search for computers, if the application tells you the location of a computer you can suppose that there are more computers in the environment that may be far away and therefore, not detected. A specific detection of an object in a space can derive in a general understanding of the environment." (Felipe Yagüe).

"The specific capability of detecting faces in an image is completely useful when wanting to know whether there are people in a picture and where they are located in it. The software can work as a photography descriptor too." (Julian Bruscantini).

Both Felipe and Julian agree that a great improvement on the application would be to analyse live videos or camera pans in order to pass the camera of the mobile phone through the surrounding spaces and get a warning when the searched object is found. In order to carry out this improvement, the speed of the communications between the computer and the mobile phone and specially the speed of the image analysis should highly increase.

Both of them would only use this application in few occasions, but they would find it really helpful on the moment of using it. They recognize that an improvement of the accuracy of the object detection and location mode would be great. This improvement of accuracy of the image analysis can be done by training the RCNN with more complex and deep datasets, by using more epochs in the training processes and restructuring the learning transfer algorithm in order to make it more precise. These modifications would not affect the overall software that forms the application.

The Internet – based interconnections between the computer and the smartphone make the user capable of having the computer program being executed in a part of the world and being at the same analysing environments and spaces with his mobile phone in an another place on the world.

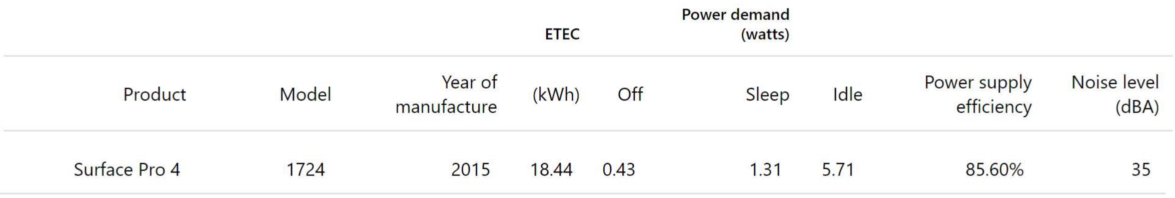
# 10. Environmental impact analysis

The major part of the environmental impact generated by this project goes to the development processes for the creation of the software. The use of the software created in this project also results in an environmental impact.

Software in no case consumes energy. However, the hardware where devices and systems where software is developed and executed do consume energy. In turn, any energy consumption by an electronic device has a repercussion in our natural environment. The effects of electric energy consumptions tend to be, if not in the totality of the cases, in the great majority of the cases, harmful to the natural resources and environments of our planet and therefore, to our lives.

## 10.1. Environmental impact generated during the development of the project

The computer used to design and implement all the technology present in this project is a Surface Pro 4 created and commercialized by Microsoft. The following table corresponds to the efficiency and acoustic information of the device.



| | | | ETEC | | Power demand (watts) | | | |
|---|---|---|---|---|---|---|---|---|
| Product | Model | Year of manufacture | (kWh) | Off | Sleep | Idle | Power supply efficiency | Noise level (dBA) |
| Surface Pro 4 | 1724 | 2015 | 18.44 | 0.43 | 1.31 | 5.71 | 85.60% | 35 |

**Figure 10.1.** Power consumption and efficiency of the Surface Pro 4

As it can be observed in the table above, the device has two main operation modes for which has different power demands: Sleep (or standby) mode and Idle (or On Off) mode. The idle mode is understood as the mode where the computer is fully opened and working.

However, when working, the power demand of the computer varies depending on the CPU usage and consumption of resources and on other factors such as memory usage and GPU usage and power consumption.

During the development of this project three Idle modes can be distinguished: The low consumption mode, the moderate consumption mode and the high consumption mode.

- Low consumption mode is that where the computer uses between a 1 and a 20% of the CPU resources. It is associated to a user level use of the computer (i.e. Internet access, document editing, etc.).
- Moderate consumption mode is that where the computer uses between a 21% and a 30% of the CPU resources. It is associated to moderate computational processes carried out by demanding programs (i.e. Computing mathematical operations in Matlab).
- High consumption mode is that where the computer uses more than a 30% of the CPU resources. It is associated to complex computational processes carried out by demanding programs (i.e. Training a RCNN in Matlab).

The next figure is the graphic representation of the percentage of the CPU that is used during 60 specific seconds in the computer where this project has been developed. From second 0 to second 34 the computer operates in low consumption mode, from second 34 to second 38 the computer operates in moderate consumption mode, from second 38 to second 50 the computer operates in high consumption mode, from second 50 to second 60 the computer operates in low consumption mode.

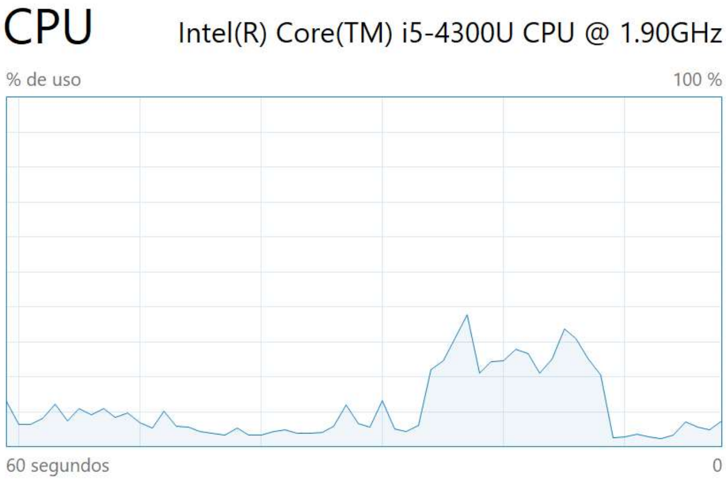


**Figure 10.2.** CPU usage of the device during 60 specific seconds

The next image shows the CPU usage histogram (in percentage) during another 60 specific seconds. Where three mathematical operations are firstly computed and an image analysis operation is finally computed.

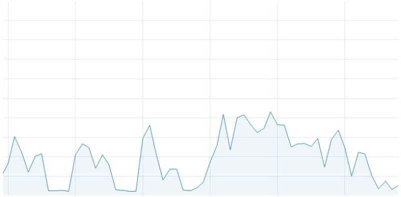


**Figure 10.3.** CPU usage of the device during 60 specific seconds

Moreover, when computing graphical operations like image analysis or RCNN training (related to the high consumption mode), the GPU power consumption increases drastically. The next figure shows the graphic representation of the percentage of the GPU that is used during 60 specific seconds in this computer. A first demanding graphical operation is computed, then the computer works in low consumption mode, subsequently another demanding graphical operation is computed and finally the computer works in low consumption mode again.

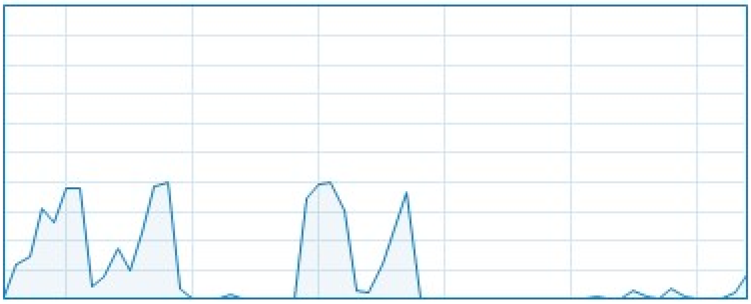


**Figure 10.4.** GPU usage of the device during 60 specific seconds

We will make the following assumptions related to power consumption and computer usage modes:

- In low mode the power consumption of the computer is a 100% of the Idle power demand. Which results in a power consumption of 5.71 Watts.
- In moderate mode the power consumption of the computer is a 150% of the Idle power demand. Which results in a power consumption of 8.865 Watts.
- In moderate mode the power consumption of the computer is a 200% of the Idle power demand. Which results in a power consumption of 11.42 Watts.

The relations between power consumption and computer operating modes shown above are not only based on the usage of the CPU and the GPU, because sometimes a demanding usage of the CPU may not result in an increment of the computer power consumption. This is due that the CPU can be operating with the maximum amount of its resources but may just be computing integer arithmetical operations that do not increase the power demand of the overall computer.

However, during the development of this project has been observed that when operating in the high consumption mode (i.e. doing complex graphical and algorithmic operations) the power demand of the computer doubles the power demand of the computer when operating in low consumption mode, as it had to be connected to the current (in other words, charging) the double amount of time.

The final power consumption of the system is greater than the power demand of the computer as the power supply efficiency is not a 100% but an 85.6%. And at the end of the day, the environmental impact is the one generated by the power supply when consuming energy from the domestic electric net.

The energy consumption of the computer depending on the time of usage is described by the following equations:

Energy – power relation:

$$1J = 1W \cdot 1s \quad \text{(Eq. 10.1)}$$

Energy consumption of the computer:

$$E = (Time\ at\ low\ mode \cdot low\ mode\ consumption + \ Time\ at\ moderate\ mode \cdot moderate\ mode\ consumption + \ Time\ at\ high\ mode \cdot high\ mode\ consumption) \cdot Efficiency\ of\ the\ power\ supply \qquad \text{(Eq. 10.2)}$$

Taking into account that these project has been developed in 4 months (120 days) and that the computer has been used an average of 4 daily hours, the total usage time is 480 hours. This 480 hours can be divided into the working modes of the compute as follows:

$$Low\ Consumption\ Mode\ Time = 40\%\ of\ the\ total\ usage\ time = 0.4 \cdot 480\ h = 192\ h \quad \text{(Eq. 10.3)}$$

$$Moderate\ Consumption\ Mode\ Time = 20\%\ of\ the\ total\ usage\ time = 0.2 \cdot 480\ h = 96\ h \quad \text{(Eq. 10.4)}$$

$$High\ Consumption\ Mode\ Time = 40\%\ of\ the\ total\ usage\ time = 0.4 \cdot 480\ h = 192\ h \quad \text{(Eq. 10.5)}$$

Thus, the total energy consumed by the computer in this project is:

$$Total\ Energy\ Consumed = (192h \cdot 0.00571\ kW + 96\ h \cdot 0.008865\ kW + 192\ h \cdot 0.01142\ kW) \cdot \frac{1}{0.856} = \ 4.8364\ \mathrm{kWh} \qquad \text{(Eq. 10.6)}$$

Therefore, the energy impact to the environment that has result from the development of this project is equal to 4.8364 kWh.

Moreover, we have to take into account the fact that using Internet consumes energy by itself as it demands the computational resources of its servers. Also, each time a variable is stored to the cloud, which can go from a JSON variable in a Firebase to a PDF document in Google Drive, physical resources are used from the servers. Therefore, the simple usage of Internet and computer systems has a wide environmental impact over multiple sectors.

It has to be taken into account the fact that both a computer and a smartphone need to be working at the same time in order for the application to fully work. The previous analysis done for the computer environmental impact can be equally done for the smartphone, which due to the lower device consumption and the lower computational demand of the smartphone application, will lead in a lower energy consumption for the phone than for the computer. The resource and energy consumption of both the computer and the phone at the moment of using the application results in a direct impact in the electric network that is used to power the devices and, therefore, in an environmental impact.

However, the footprint that this project leaves in the environment is not greater than the one that the vast majority of AI – based software resources leave. The fact of occupying cloud memory storage space is not new to this project, and the usage of Internet resources is not very alarming as single images and a pair of JSON strings are updated to a Firebase every 20 seconds if the software is used incessantly with a high rate of analysed images.

# Conclusions

The application works, as each of the three elements that form the totality of the software (phone application, computer program and Firebase databse) are completely functional and are successfully interconnected and structured so that the information transmission between them is fluid and robust. The structure of the developed software is efficient and permits the rapid and organized communications between the computer program and the smartphone application, being the Firebase the core tool of the communications.

The general definition of spaces works better than the specific detection and location of objects. This is due to the fact that the neural networks used for the general description of environments are pretrained CNNs that have been tested, validated and improved for a much longer time and by more researchers and developers than the RCNNs used to detect and locate specific objects, which have been developed with less and smaller datasets and fewer resources as the training, testing and research processes of this networks have been done, in their totality, in this project.

The image analysis done by the application is helpful to its user despite some analysis may be performed erroneously because it gives him or her a general understanding of the physical characteristics of the place he or she is located in. The chances of an object of being correctly detected and located are sufficiently high to consider using this application when the user wants to know whether a specific object is in front of him or not. The chances of an environment of being correctly described is sufficiently high to consider using this application when the user wants to get a general definition of his or her surroundings.

Of course, more object classes can be introduced to the software as possible objects to detect, as this is a scalable application. The work falls on creating an appropriate dataset and using it to train the new RCNN with the learning transfer method. These extensions of the software require computational time and resources.

The image segmentation model for object detection provides an efficient solution to the problems that were firstly contemplated when analysing images, as the analysis are much more reliable with the use of this method although the networks used for the analysis are the same.

The Mit App Inventor program presents a software – user interface that is characterized by its ease of use. The Matlab program presents a software that follows a complex structure in order to treat and organize data and neural networks and developing their interactions in order to perform efficient analysis of the images to inspect. The Firebase is correctly organized, resulting an ideal communication bridge between the computer and the smartphone.

The most significant improvements over this project are done by investigating the neural networks that are used in order to create more robust detection models. However, the Mit App Inventor program design and implementation, the Matlab Application program development and the structuration of the elements that compute the communication operations present efficient and functional performances that provide robust solutions to a great part of the objectives presented at the beginning of this project.

The fully accomplished objectives in this project are:

- The creation of a software application completely adapted for its use by people with visual dysfunction. Thus, a fully voice-operated software.
- The programming of software that imports the images to be analysed, passes them through the AI networks and converts the conclusions drawn from the image inspection into results interpretable by the user.
- The design and implementation of an efficient infrastructure that connects the different elements and platforms that may form the system. The connection has to be fast, must let each element send and receive different types of information and must not be interrupted or annulated when no information is being sent.
- Treatment of cloud storage services, in order to create the infrastructure previously mentioned.

The partially accomplished objectives in this projects, which do not present a decisive need of further research but do present the necessity of escalating the work done in order to generate a more robust and reliable system are:

- The implementation and structuration of already trained networks in order to carry out general analysis of images, classifying the inspected images by the place that they represent (bedroom, square, kitchen, etc.) and recognizing the object that occupies the largest area on the picture.
- The collection, selection, documentation and treatment of data in order to constitute appropriate datasets for the training and testing of the networks.

The partially accomplished objectives in this projects, which present a high need of further research and the necessity of escalating the work done in order to generate a more robust and reliable system are:

- The designing, training and validating of convolutional networks so that they are capable of detecting, classifying and locating different classes of objects in the analysed images.

It is also important to remark that the development of the software that presents the core subject of this project and its posterior usage by its user does not generate a bigger environmental impact than the impact that is generated by other applications and software based on Artificial Intelligence, object detection and database usage.

The developed software is designed to be used by a single user, if it is wanted to be commercialized or extended, registration methods should be developed over the Firebase communications in order to make different users use different repositories of the database, so that information is not overlapped and loosed. However, with this extension of the project the Matlab and the Mit App Inventor programs would remain almost intact and neural networks would remain completely intact, just changing communication protocols with the Firebase.

# Budget and economic analysis

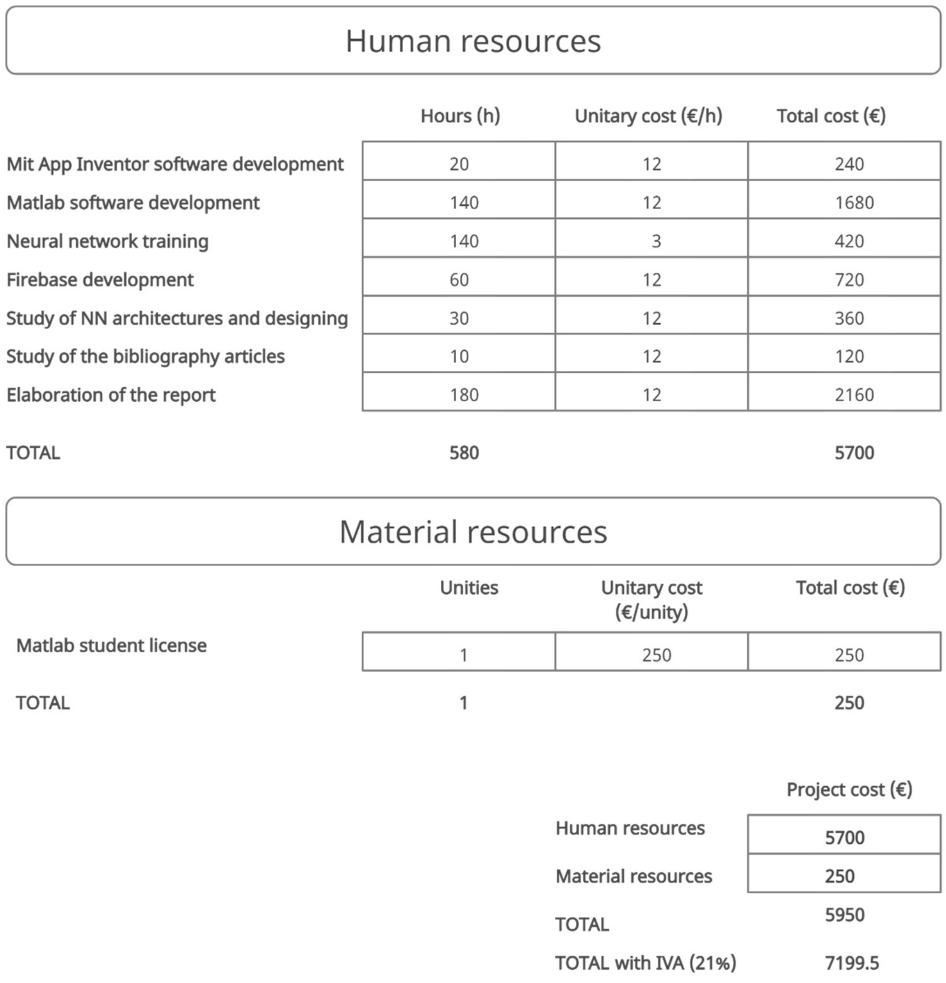
## Economic analysis of the project development

The material costs are represented by the electric energy used to supply the computer where the entirety of the project has been developed, the cost represented by the monthly payment of Internet services and the cost of the Matlab license. It would be incorrect to assume that the cost of the energy extracted from the electric net for the development of this project has been null or practically null, but it is understood as a domestic payment that is not contemplated as a significant cost of this project, as well as the cost of the Internet services. The usage of Matlab has not represented a direct cost on the developer, Pau Besses Casas, as the software license has been paid by the university, "Escola d'Enginyeria de Barcelona Est, Universitat Politècnica de Catalunya". The used license is the education license, which is used for academic use and has an annual cost of 250 €.

Therefore, the software costs of this project are: 250 € for each year of development, while being a university student. If the project is wanted to be further developed in the future, it has to be taken into account that Matlab license will have to be paid, and the license cost depends on the user's situation during each year of development.

We also have to take into account the human resources invested in this project. I personally value the human work done by myself during the development of this project for a price of 12€/h. I have valued the hours that the computer alone has taken to perform neural network training for a price of 3€/h, as I just had the work of supervising the process.

The next table shows the cost distribution and analysis of the project:



## Human resources

| | Hours (h) | Unitary cost (€/h) | Total cost (€) |
|---|---|---|---|
| Mit App Inventor software development | 20 | 12 | 240 |
| Matlab software development | 140 | 12 | 1680 |
| Neural network training | 140 | 3 | 420 |
| Firebase development | 60 | 12 | 720 |
| Study of NN architectures and designing | 30 | 12 | 360 |
| Study of the bibliography articles | 10 | 12 | 120 |
| Elaboration of the report | 180 | 12 | 2160 |
| **TOTAL** | **580** | | **5700** |

## Material resources

| | Unities | Unitary cost (€/unity) | Total cost (€) |
|---|---|---|---|
| Matlab student license | 1 | 250 | 250 |
| **TOTAL** | **1** | | **250** |

| | Project cost (€) |
|---|---|
| Human resources | 5700 |
| Material resources | 250 |
| TOTAL | 5950 |
| TOTAL with IVA (21%) | 7199.5 |

# Bibliography

The next articles and sites correspond to the information sources that are cited along this document.

1. Add Firebase to your Android project. (n.d.). Firebase. Retrieved March 1, 2021, from https://firebase.google.com/docs/android/setup
2. Economic impacts of artificial intelligence (AI) - Think Tank. (n.d.). © European Union, 2016 - Source: European Parliament. Retrieved March 3, 2021, from https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI(2019)637967
3. ImageNet. (n.d.). Dataset. Retrieved March 11, 2021, from https://www.image-net.org/
4. Khosla, A. (n.d.). Places2: A Large-Scale Database for Scene Understanding. Matlab. Retrieved May 10, 2021, from http://places2.csail.mit.edu/download.html
5. M. (2018, November 6). [Free] FirebaseStorage Extension v2.0 (New Version). Community. https://community.thunkable.com/t/free-firebasestorage-extension-v2-0-new-version/8544
6. MATLAB - El lenguaje del cálculo técnico. (n.d.). MATLAB & Simulink. Retrieved April 10, 2021, from https://es.mathworks.com/products/matlab.html
7. MATLAB App Designer. (n.d.). MATLAB. Retrieved April 10, 2021, from https://es.mathworks.com/products/matlab/app-designer.html
8. MATLAB Live Editor. (n.d.). MATLAB. Retrieved April 10, 2021, from https://es.mathworks.com/products/matlab/live-editor.html
9. MIT App Inventor | Explore MIT App Inventor. (n.d.). Mit App Iventor. Retrieved April 20, 2021, from https://appinventor.mit.edu/
10. Pretrained Deep Neural Networks - MATLAB & Simulink - MathWorks España. (n.d.). Deep Learning. Retrieved March 23, 2021, from https://es.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html
11. Szegedy, C. (2014, September 17). Going Deeper with Convolutions. ArXiv.Org. https://arxiv.org/abs/1409.4842
12. Veen, F. (2019, April 27). The Neural Network Zoo. The Asimov Institute. https://www.asimovinstitute.org/neural-network-zoo/

The next articles and sites have been visited for the elaboration of this report.

Cell Arrays of Character Vectors - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved April 16, 2021, from https://es.mathworks.com/help/matlab/matlab_prog/cell-arrays-of-strings.html

Classify Image Using GoogLeNet - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved April 7, 2021, from https://es.mathworks.com/help/deeplearning/ug/classify-image-using-googlenet.html

Convert table to cell array - MATLAB table2cell - MathWorks España. (n.d.). Matlab. Retrieved April 8, 2021, from https://es.mathworks.com/help/matlab/ref/table2cell.html

Create YOLO v2 object detection network - MATLAB yolov2Layers - MathWorks España. (n.d.). Matlab. Retrieved February 28, 2021, from https://es.mathworks.com/help/vision/ref/yolov2layers.html#mw_b1e7b78e-2ce3-4d2f-ad8d-f5acd725f170_sep_mw_58aefdb4-f28e-4bf2-bd56-b393c24116d1

CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). CNN. Retrieved May 11, 2021, from https://cs231n.github.io/convolutional-networks/

Getting Started with YOLO v2 - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved February 24, 2021, from https://es.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html

Ground truth label data - MATLAB - MathWorks España. (n.d.). Matlab. Retrieved March 24, 2021, from https://es.mathworks.com/help/vision/ref/groundtruth.html

Guardar variables del espacio de trabajo en un archivo - MATLAB save - MathWorks España. (n.d.). Matlab. Retrieved April 2, 2021, from https://es.mathworks.com/help/matlab/ref/save.html

how to save and reuse a trained neural network - MATLAB Answers - MATLAB Central. (n.d.). Matlab. Retrieved May 8, 2021, from https://es.mathworks.com/matlabcentral/answers/264160-how-to-save-and-reuse-a-trained-neural-network

Label images for computer vision applications - MATLAB - MathWorks España. (n.d.). Matlab. Retrieved March 24, 2021, from https://es.mathworks.com/help/vision/ref/imagelabeler-app.html

Load Ground Truth Signals to Label - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved March 23, 2021, from https://es.mathworks.com/help/driving/ug/load-ground-truth-signals-to-label.html

ResNet-50 convolutional neural network - MATLAB resnet50 - MathWorks España. (n.d.). Matlab. Retrieved March 28, 2021, from https://es.mathworks.com/help/deeplearning/ref/resnet50.html

save matrix. . ..load matrix - MATLAB Answers - MATLAB Central. (n.d.). Matlab. Retrieved April 14, 2021, from https://es.mathworks.com/matlabcentral/answers/114004-save-matrix-load-matrix

Speech-to-Text Transcription - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved April 8, 2021, from https://es.mathworks.com/help/audio/ug/speech2text.html

Train Object Detector Using R-CNN Deep Learning - MATLAB & Simulink - MathWorks España. (n.d.). Matlab. Retrieved April 13, 2021, from https://es.mathworks.com/help/vision/ug/object-detection-using-deep-learning.html

What Is MATLAB Mobile for Android? - Video. (n.d.). MATLAB. Retrieved June 1, 2021, from https://es.mathworks.com/videos/matlab-mobile-for-android-overview-72032.html

How to replace character in a word or a string. (2020, September 5). MIT App Inventor Community. https://community.appinventor.mit.edu/t/how-to-replace-character-in-a-word-or-a-string/15785

Pricing and Licensing. (n.d.). MATLAB & Simulink. Retrieved June 10, 2021, from https://es.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=home

Sending data to Firebase from Matlab. (2018, November 26). Stack Overflow. https://stackoverflow.com/questions/53483531/sending-data-to-firebase-from-matlab

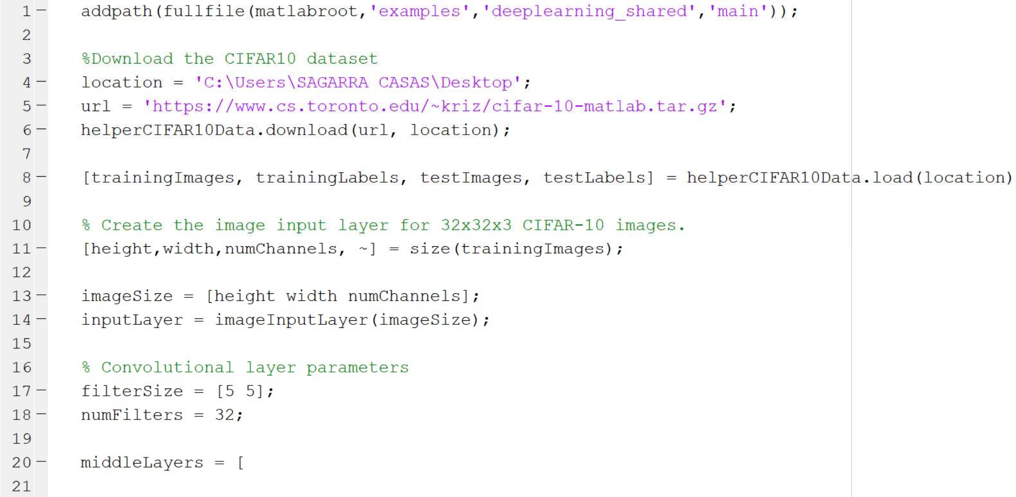
Specify parameters for RESTful web service - MATLAB weboptions - MathWorks España. (n.d.). Matlab. Retrieved March 19, 2021, from https://es.mathworks.com/help/matlab/ref/weboptions.html#d123e1479915

Uploading data from matlab to firebase. (2019, March 24). Stack Overflow. https://stackoverflow.com/questions/55325083/uploading-data-from-matlab-to-firebase

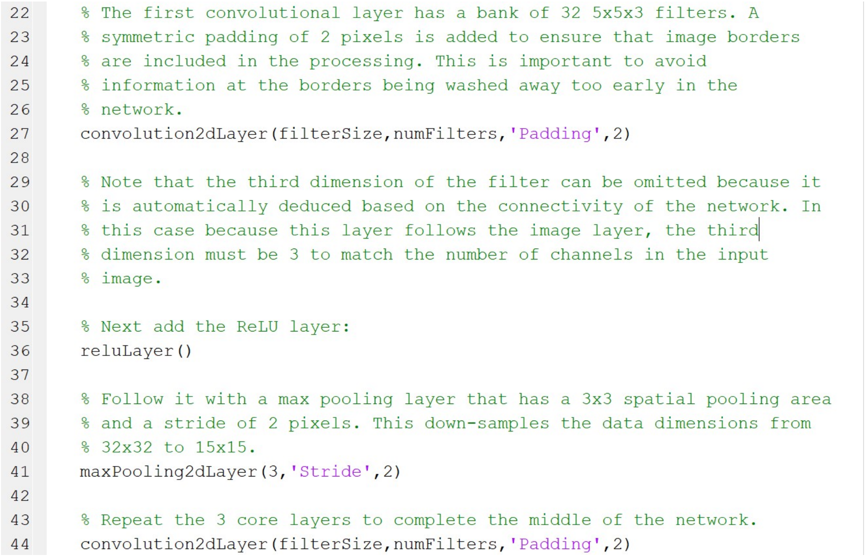
# Annexes

## Training performances of the key detection and lamp detection networks

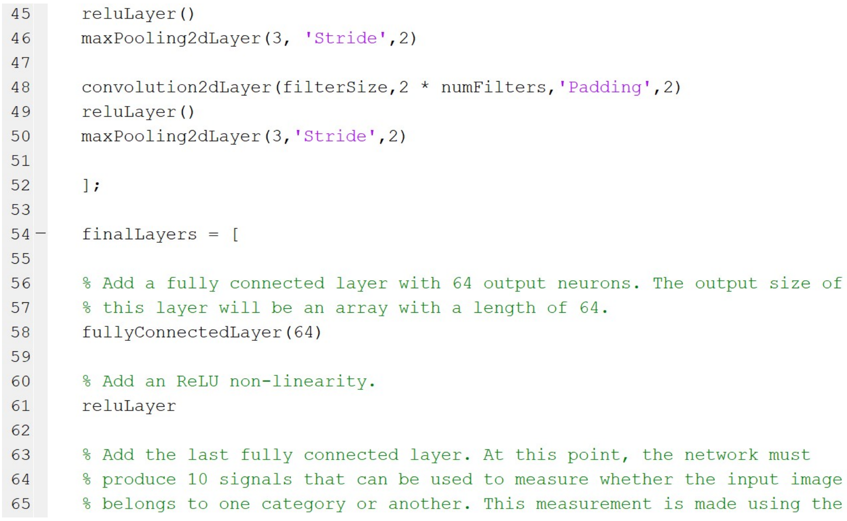
## A1. Complete commented Matlab code for the development, training and testing of CNN using the CIFAR10 dataset.



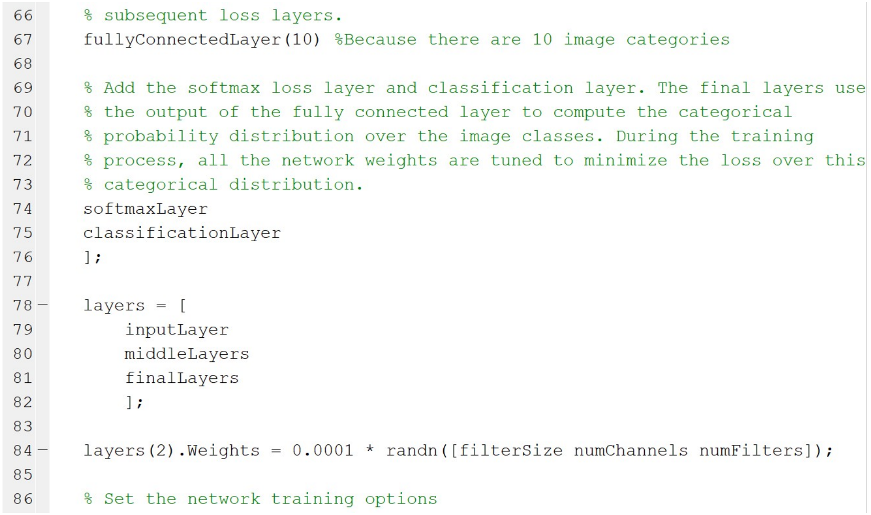
```
addpath(fullfile(matlabroot,'examples','deeplearning_shared','main'));

%Download the CIFAR10 dataset
location = 'C:\Users\SAGARRA CASAS\Desktop';
url = 'https://www.cs.toronto.edu/~kriz/cifar-10-matlab.tar.gz';
helperCIFAR10Data.download(url, location);

[trainingImages, trainingLabels, testImages, testLabels] = helperCIFAR10Data.load(location)

% Create the image input layer for 32x32x3 CIFAR-10 images.
[height,width,numChannels, ~] = size(trainingImages);

imageSize = [height width numChannels];
inputLayer = imageInputLayer(imageSize);

% Convolutional layer parameters
filterSize = [5 5];
numFilters = 32;

middleLayers = [

```



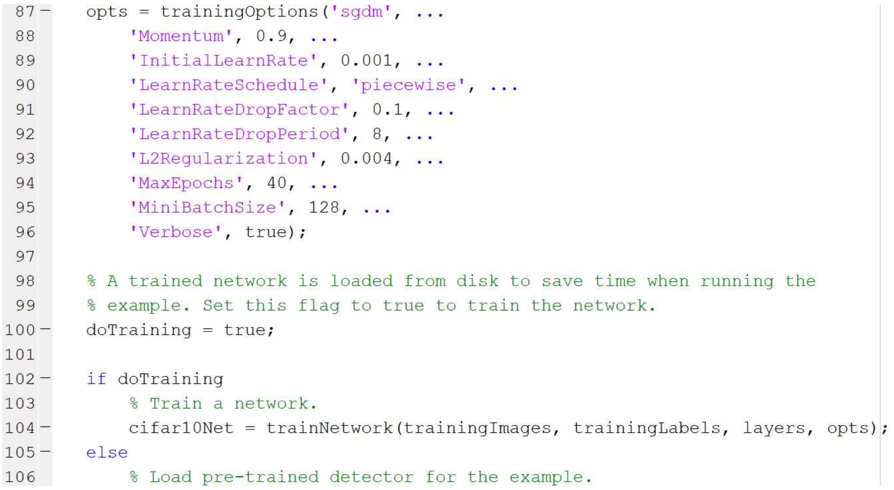
```
% The first convolutional layer has a bank of 32 5x5x3 filters. A
% symmetric padding of 2 pixels is added to ensure that image borders
% are included in the processing. This is important to avoid
% information at the borders being washed away too early in the
% network.
convolution2dLayer(filterSize,numFilters,'Padding',2)

% Note that the third dimension of the filter can be omitted because it
% is automatically deduced based on the connectivity of the network. In
% this case because this layer follows the image layer, the third
% dimension must be 3 to match the number of channels in the input
% image.

% Next add the ReLU layer:
reluLayer()

% Follow it with a max pooling layer that has a 3x3 spatial pooling area
% and a stride of 2 pixels. This down-samples the data dimensions from
% 32x32 to 15x15.
maxPooling2dLayer(3,'Stride',2)

% Repeat the 3 core layers to complete the middle of the network.
convolution2dLayer(filterSize,numFilters,'Padding',2)
```



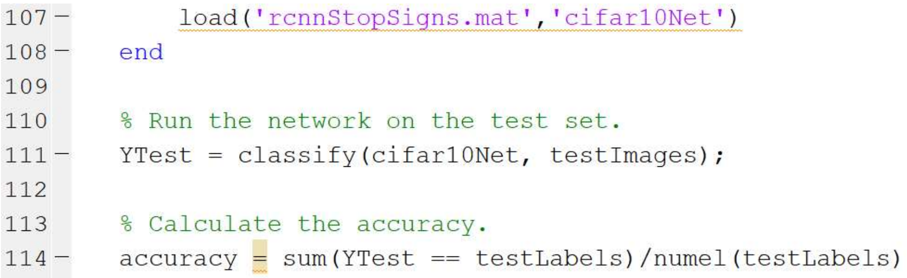
```
reluLayer()
maxPooling2dLayer(3, 'Stride',2)

convolution2dLayer(filterSize,2 * numFilters,'Padding',2)
reluLayer()
maxPooling2dLayer(3,'Stride',2)

];

finalLayers = [

% Add a fully connected layer with 64 output neurons. The output size of
% this layer will be an array with a length of 64.
fullyConnectedLayer(64)

% Add an ReLU non-linearity.
reluLayer

% Add the last fully connected layer. At this point, the network must
% produce 10 signals that can be used to measure whether the input image
% belongs to one category or another. This measurement is made using the
```



```
% subsequent loss layers.
fullyConnectedLayer(10) %Because there are 10 image categories

% Add the softmax loss layer and classification layer. The final layers use
% the output of the fully connected layer to compute the categorical
% probability distribution over the image classes. During the training
% process, all the network weights are tuned to minimize the loss over this
% categorical distribution.
softmaxLayer
classificationLayer
];

layers = [
    inputLayer
    middleLayers
    finalLayers
    ];

layers(2).Weights = 0.0001 * randn([filterSize numChannels numFilters]);

% Set the network training options
```



```
opts = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'MaxEpochs', 40, ...
    'MiniBatchSize', 128, ...
    'Verbose', true);

% A trained network is loaded from disk to save time when running the
% example. Set this flag to true to train the network.
doTraining = true;

if doTraining
    % Train a network.
    cifar10Net = trainNetwork(trainingImages, trainingLabels, layers, opts);
else
    % Load pre-trained detector for the example.
```

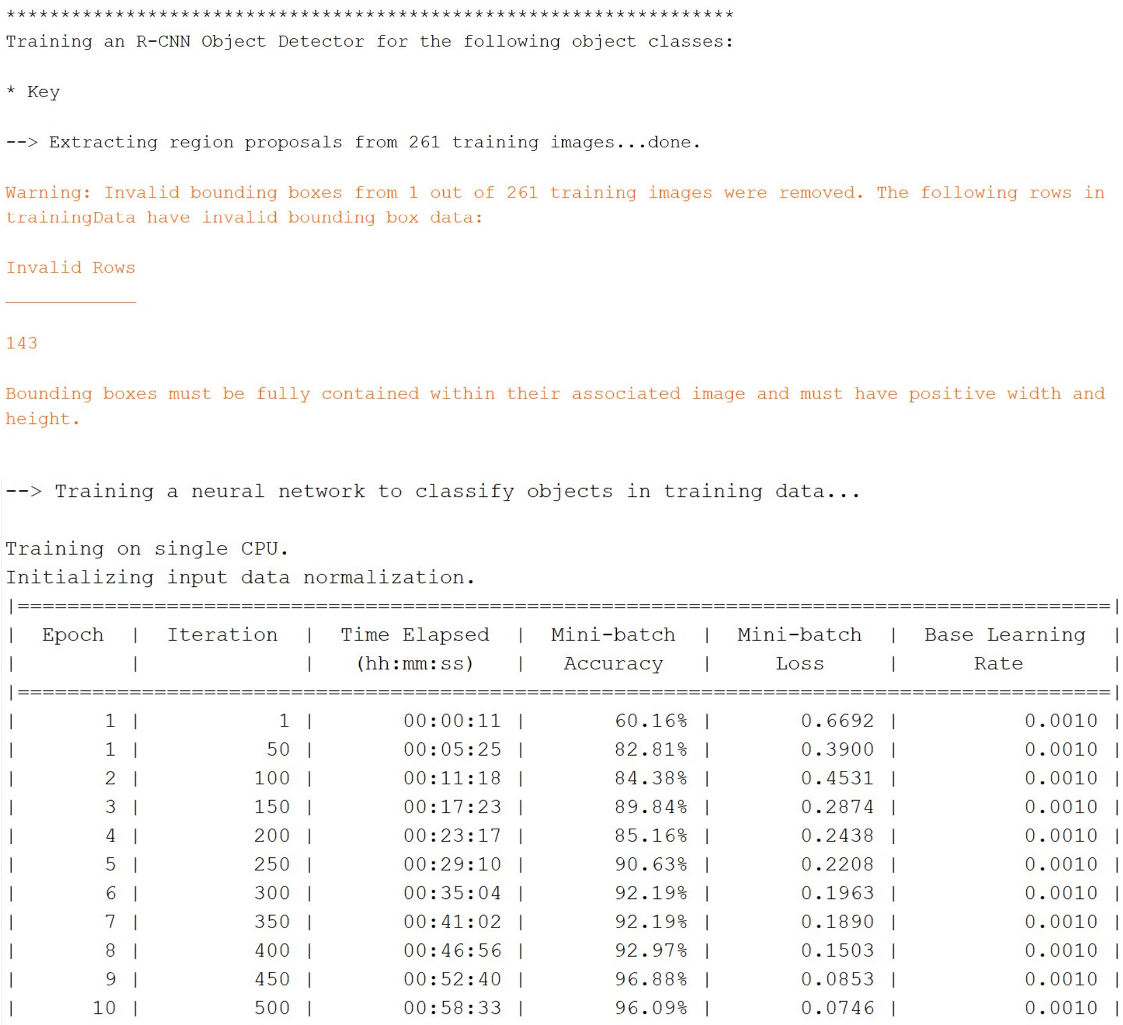


```
        load('rcnnStopSigns.mat','cifar10Net')
    end

    % Run the network on the test set.
    YTest = classify(cifar10Net, testImages);

    % Calculate the accuracy.
    accuracy = sum(YTest == testLabels)/numel(testLabels)
```

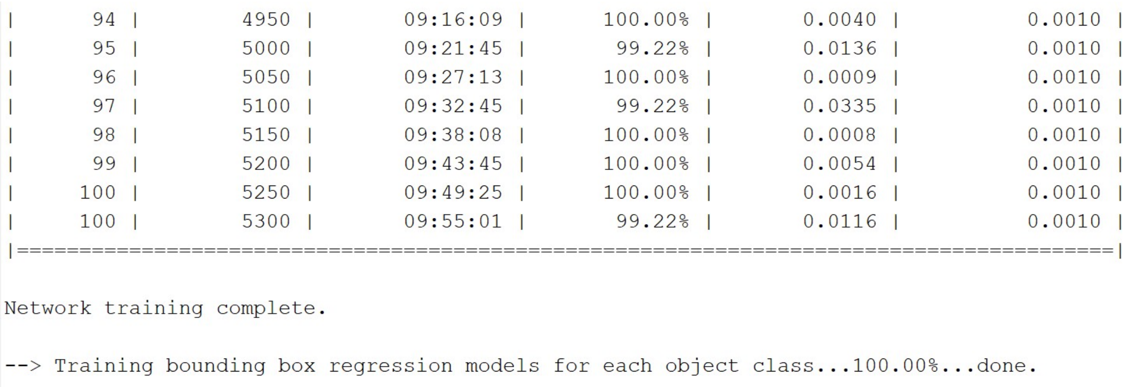
**Figure A.1.** CNN development for the posterior computation of transfer learning

## A2. Training performances of the key detection, lamp detection and backpack detection networks

The next figure shows extracts of the code corresponding to the steps that have been computed during the training process of the RCNNs for the detection of keys, lamps and backpacks and the results of the performances of each epoch.

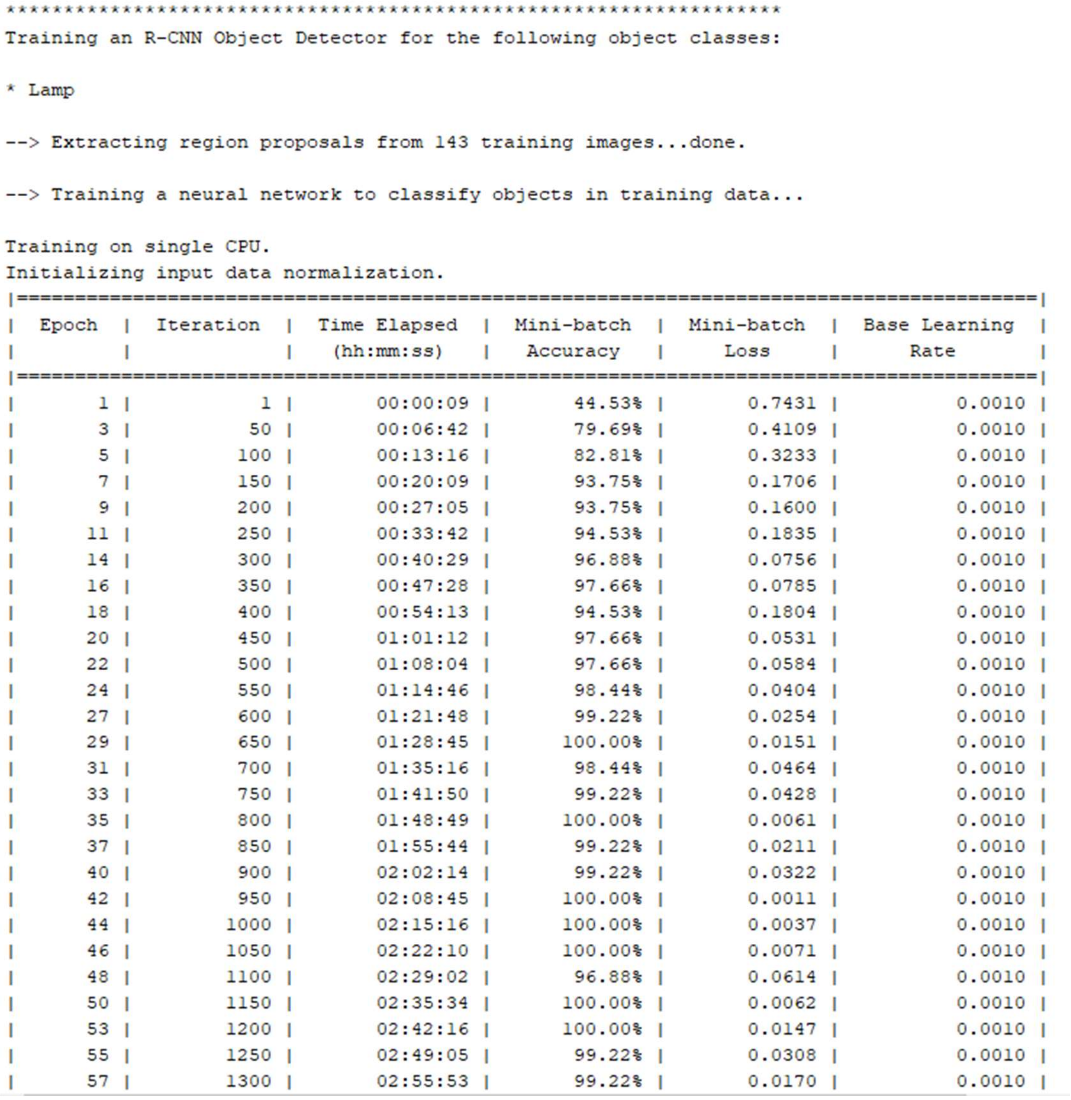


```
*******************************************************************
Training an R-CNN Object Detector for the following object classes:

* Key

--> Extracting region proposals from 261 training images...done.

Warning: Invalid bounding boxes from 1 out of 261 training images were removed. The following rows in
trainingData have invalid bounding box data:

Invalid Rows
____________

143

Bounding boxes must be fully contained within their associated image and must have positive width and
height.


--> Training a neural network to classify objects in training data...

Training on single CPU.
Initializing input data normalization.
|======================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|======================================================================================|
|       1 |           1 |       00:00:11 |       60.16% |       0.6692 |          0.0010 |
|       1 |          50 |       00:05:25 |       82.81% |       0.3900 |          0.0010 |
|       2 |         100 |       00:11:18 |       84.38% |       0.4531 |          0.0010 |
|       3 |         150 |       00:17:23 |       89.84% |       0.2874 |          0.0010 |
|       4 |         200 |       00:23:17 |       85.16% |       0.2438 |          0.0010 |
|       5 |         250 |       00:29:10 |       90.63% |       0.2208 |          0.0010 |
|       6 |         300 |       00:35:04 |       92.19% |       0.1963 |          0.0010 |
|       7 |         350 |       00:41:02 |       92.19% |       0.1890 |          0.0010 |
|       8 |         400 |       00:46:56 |       92.97% |       0.1503 |          0.0010 |
|       9 |         450 |       00:52:40 |       96.88% |       0.0853 |          0.0010 |
|      10 |         500 |       00:58:33 |       96.09% |       0.0746 |          0.0010 |
```

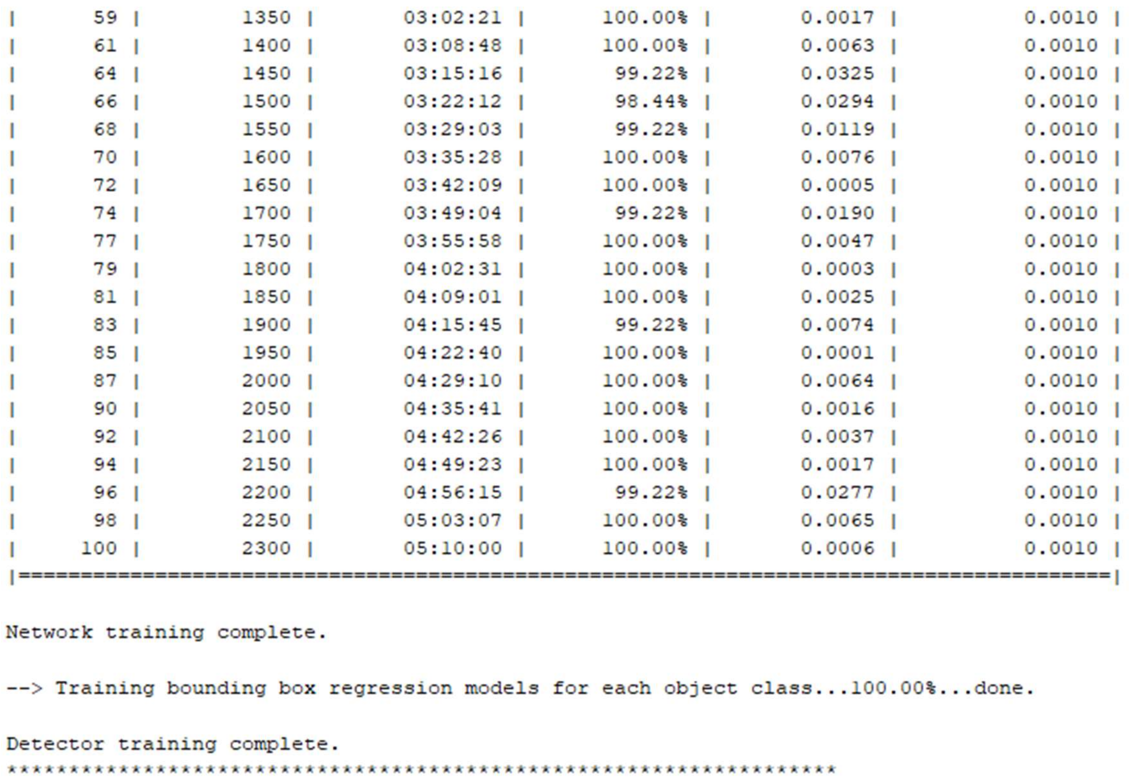


```
|      94 |         4950 |       09:16:09 |      100.00% |         0.0040 |            0.0010 |
|      95 |         5000 |       09:21:45 |       99.22% |         0.0136 |            0.0010 |
|      96 |         5050 |       09:27:13 |      100.00% |         0.0009 |            0.0010 |
|      97 |         5100 |       09:32:45 |       99.22% |         0.0335 |            0.0010 |
|      98 |         5150 |       09:38:08 |      100.00% |         0.0008 |            0.0010 |
|      99 |         5200 |       09:43:45 |      100.00% |         0.0054 |            0.0010 |
|     100 |         5250 |       09:49:25 |      100.00% |         0.0016 |            0.0010 |
|     100 |         5300 |       09:55:01 |       99.22% |         0.0116 |            0.0010 |
|=========================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.
```

**Figure A.2.** Extracts of the training process for the key detection RCNN

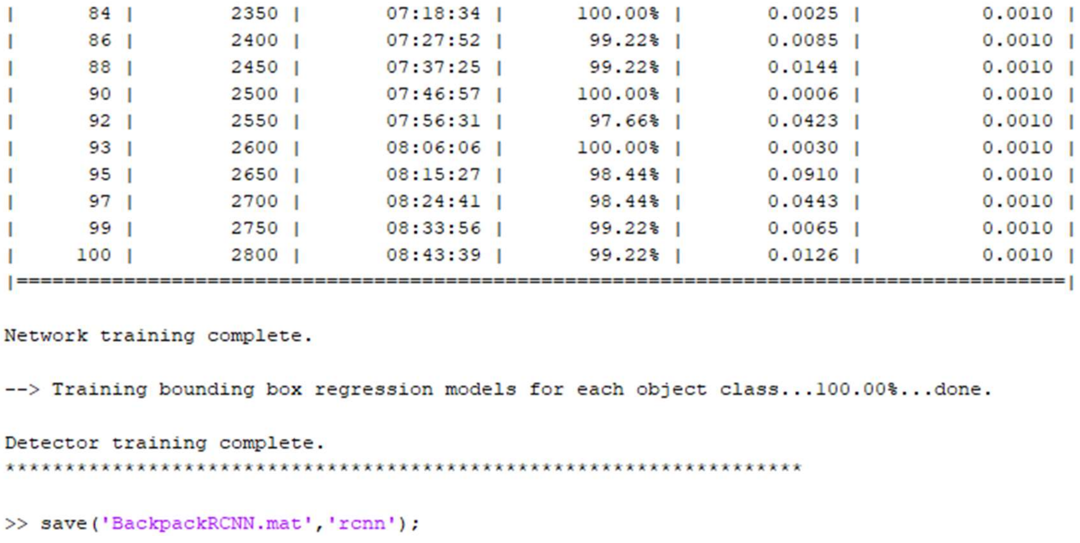


```
*******************************************************************
Training an R-CNN Object Detector for the following object classes:

* Lamp

--> Extracting region proposals from 143 training images...done.

--> Training a neural network to classify objects in training data...

Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:09 |       44.53% |       0.7431 |          0.0010 |
|       3 |          50 |       00:06:42 |       79.69% |       0.4109 |          0.0010 |
|       5 |         100 |       00:13:16 |       82.81% |       0.3233 |          0.0010 |
|       7 |         150 |       00:20:09 |       93.75% |       0.1706 |          0.0010 |
|       9 |         200 |       00:27:05 |       93.75% |       0.1600 |          0.0010 |
|      11 |         250 |       00:33:42 |       94.53% |       0.1835 |          0.0010 |
|      14 |         300 |       00:40:29 |       96.88% |       0.0756 |          0.0010 |
|      16 |         350 |       00:47:28 |       97.66% |       0.0785 |          0.0010 |
|      18 |         400 |       00:54:13 |       94.53% |       0.1804 |          0.0010 |
|      20 |         450 |       01:01:12 |       97.66% |       0.0531 |          0.0010 |
|      22 |         500 |       01:08:04 |       97.66% |       0.0584 |          0.0010 |
|      24 |         550 |       01:14:46 |       98.44% |       0.0404 |          0.0010 |
|      27 |         600 |       01:21:48 |       99.22% |       0.0254 |          0.0010 |
|      29 |         650 |       01:28:45 |      100.00% |       0.0151 |          0.0010 |
|      31 |         700 |       01:35:16 |       98.44% |       0.0464 |          0.0010 |
|      33 |         750 |       01:41:50 |       99.22% |       0.0428 |          0.0010 |
|      35 |         800 |       01:48:49 |      100.00% |       0.0061 |          0.0010 |
|      37 |         850 |       01:55:44 |       99.22% |       0.0211 |          0.0010 |
|      40 |         900 |       02:02:14 |       99.22% |       0.0322 |          0.0010 |
|      42 |         950 |       02:08:45 |      100.00% |       0.0011 |          0.0010 |
|      44 |        1000 |       02:15:16 |      100.00% |       0.0037 |          0.0010 |
|      46 |        1050 |       02:22:10 |      100.00% |       0.0071 |          0.0010 |
|      48 |        1100 |       02:29:02 |       96.88% |       0.0614 |          0.0010 |
|      50 |        1150 |       02:35:34 |      100.00% |       0.0062 |          0.0010 |
|      53 |        1200 |       02:42:16 |      100.00% |       0.0147 |          0.0010 |
|      55 |        1250 |       02:49:05 |       99.22% |       0.0308 |          0.0010 |
|      57 |        1300 |       02:55:53 |       99.22% |       0.0170 |          0.0010 |
```



```
|      59 |        1350 |      03:02:21 |      100.00% |       0.0017 |         0.0010 |
|      61 |        1400 |      03:08:48 |      100.00% |       0.0063 |         0.0010 |
|      64 |        1450 |      03:15:16 |       99.22% |       0.0325 |         0.0010 |
|      66 |        1500 |      03:22:12 |       98.44% |       0.0294 |         0.0010 |
|      68 |        1550 |      03:29:03 |       99.22% |       0.0119 |         0.0010 |
|      70 |        1600 |      03:35:28 |      100.00% |       0.0076 |         0.0010 |
|      72 |        1650 |      03:42:09 |      100.00% |       0.0005 |         0.0010 |
|      74 |        1700 |      03:49:04 |       99.22% |       0.0190 |         0.0010 |
|      77 |        1750 |      03:55:58 |      100.00% |       0.0047 |         0.0010 |
|      79 |        1800 |      04:02:31 |      100.00% |       0.0003 |         0.0010 |
|      81 |        1850 |      04:09:01 |      100.00% |       0.0025 |         0.0010 |
|      83 |        1900 |      04:15:45 |       99.22% |       0.0074 |         0.0010 |
|      85 |        1950 |      04:22:40 |      100.00% |       0.0001 |         0.0010 |
|      87 |        2000 |      04:29:10 |      100.00% |       0.0064 |         0.0010 |
|      90 |        2050 |      04:35:41 |      100.00% |       0.0016 |         0.0010 |
|      92 |        2100 |      04:42:26 |      100.00% |       0.0037 |         0.0010 |
|      94 |        2150 |      04:49:23 |      100.00% |       0.0017 |         0.0010 |
|      96 |        2200 |      04:56:15 |       99.22% |       0.0277 |         0.0010 |
|      98 |        2250 |      05:03:07 |      100.00% |       0.0065 |         0.0010 |
|     100 |        2300 |      05:10:00 |      100.00% |       0.0006 |         0.0010 |
|=========================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.

Detector training complete.
*******************************************************************
```

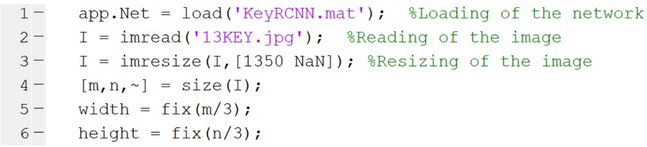
**Figure A.3.** Extracts of the training process for the lamp detection RCNN



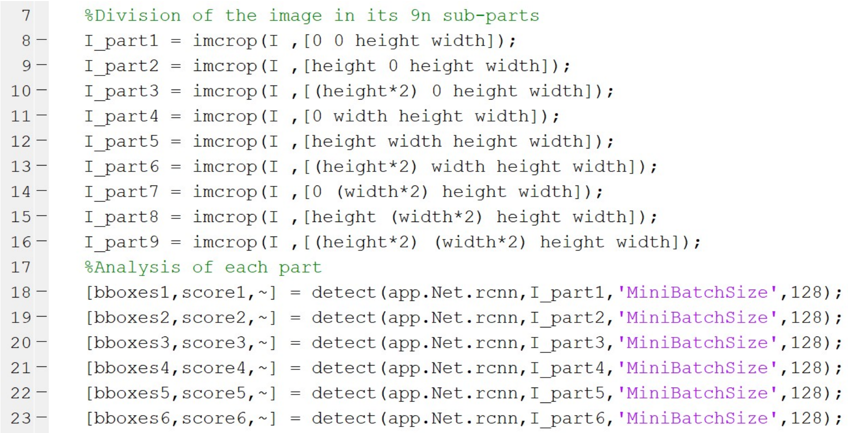
```
|      84 |        2350 |      07:18:34 |      100.00% |       0.0025 |         0.0010 |
|      86 |        2400 |      07:27:52 |       99.22% |       0.0085 |         0.0010 |
|      88 |        2450 |      07:37:25 |       99.22% |       0.0144 |         0.0010 |
|      90 |        2500 |      07:46:57 |      100.00% |       0.0006 |         0.0010 |
|      92 |        2550 |      07:56:31 |       97.66% |       0.0423 |         0.0010 |
|      93 |        2600 |      08:06:06 |      100.00% |       0.0030 |         0.0010 |
|      95 |        2650 |      08:15:27 |       98.44% |       0.0910 |         0.0010 |
|      97 |        2700 |      08:24:41 |       98.44% |       0.0443 |         0.0010 |
|      99 |        2750 |      08:33:56 |       99.22% |       0.0065 |         0.0010 |
|     100 |        2800 |      08:43:39 |       99.22% |       0.0126 |         0.0010 |
|=========================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.

Detector training complete.
*******************************************************************

>> save('BackpackRCNN.mat','rcnn');
```

**Figure A.4.** Final extract of the training process for the backpack detection RCNN

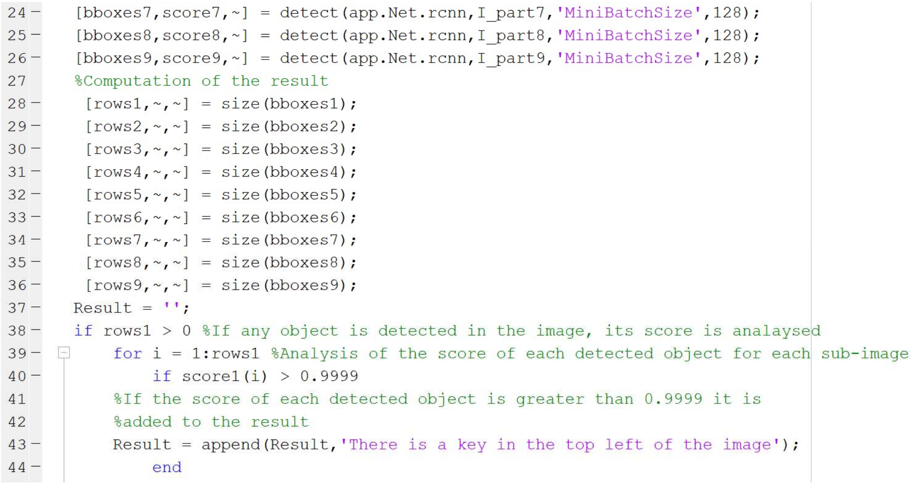
# A3. Commented Matlab code for the subdivision of images in 9 parts, their analysis with a RCNN and the output of a description of the location of the found objects



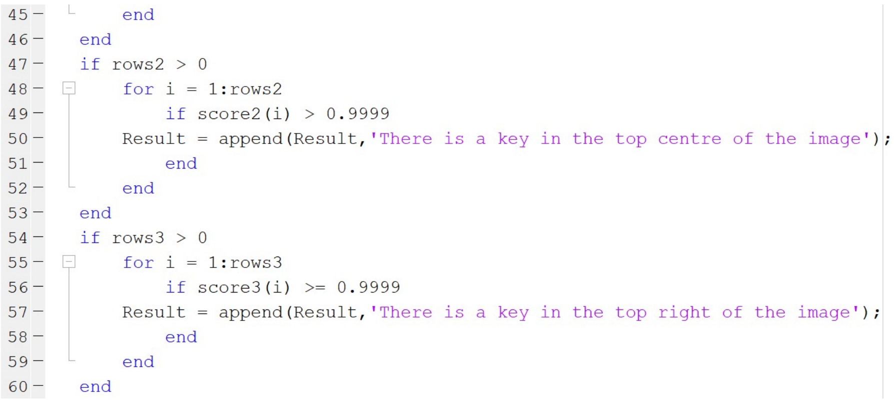
```
app.Net = load('KeyRCNN.mat');  %Loading of the network
I = imread('13KEY.jpg');  %Reading of the image
I = imresize(I,[1350 NaN]); %Resizing of the image
[m,n,~] = size(I);
width = fix(m/3);
height = fix(n/3);
```



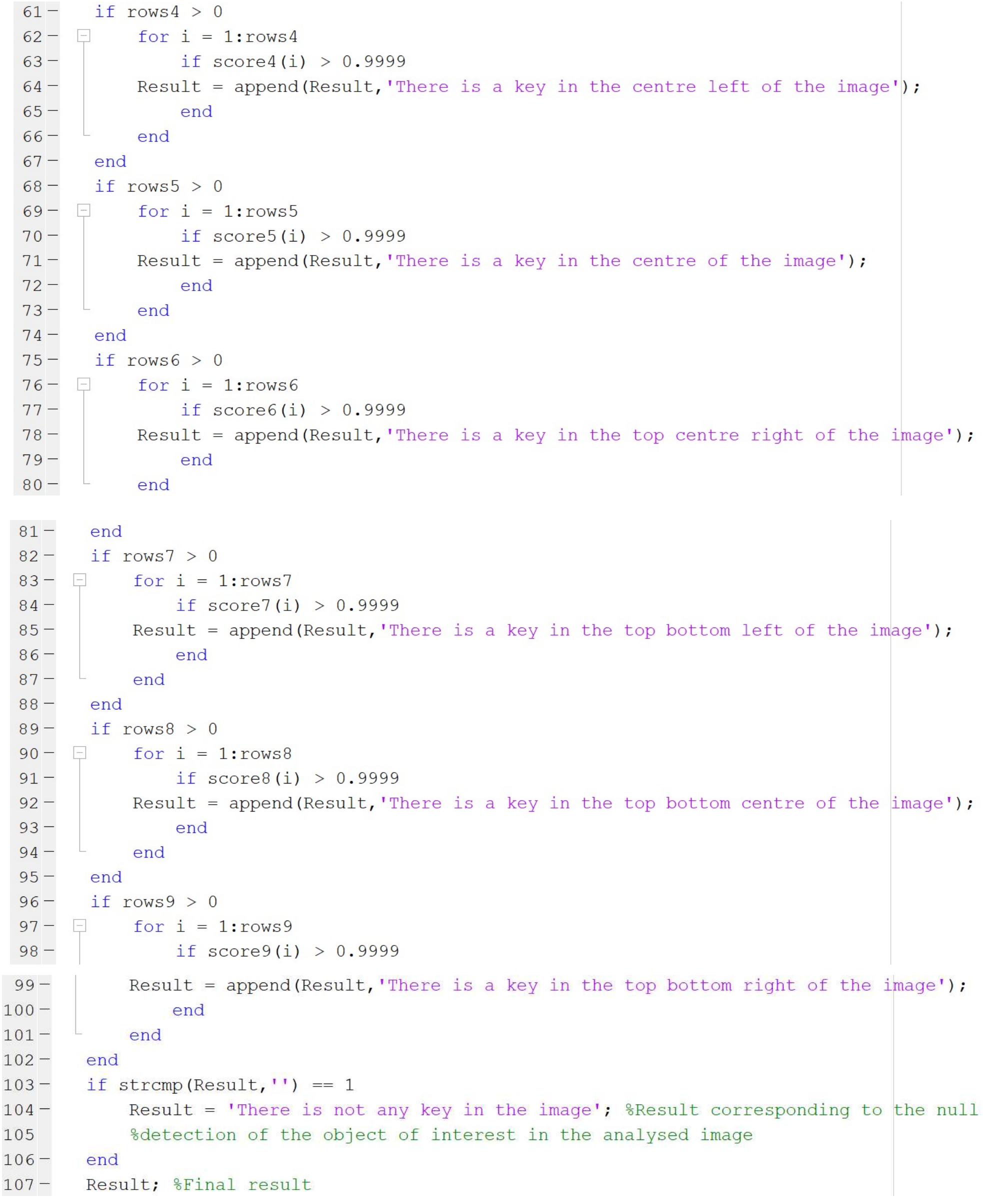
```
%Division of the image in its 9n sub-parts
I_part1 = imcrop(I ,[0 0 height width]);
I_part2 = imcrop(I ,[height 0 height width]);
I_part3 = imcrop(I ,[(height*2) 0 height width]);
I_part4 = imcrop(I ,[0 width height width]);
I_part5 = imcrop(I ,[height width height width]);
I_part6 = imcrop(I ,[(height*2) width height width]);
I_part7 = imcrop(I ,[0 (width*2) height width]);
I_part8 = imcrop(I ,[height (width*2) height width]);
I_part9 = imcrop(I ,[(height*2) (width*2) height width]);
%Analysis of each part
[bboxes1,score1,~] = detect(app.Net.rcnn,I_part1,'MiniBatchSize',128);
[bboxes2,score2,~] = detect(app.Net.rcnn,I_part2,'MiniBatchSize',128);
[bboxes3,score3,~] = detect(app.Net.rcnn,I_part3,'MiniBatchSize',128);
[bboxes4,score4,~] = detect(app.Net.rcnn,I_part4,'MiniBatchSize',128);
[bboxes5,score5,~] = detect(app.Net.rcnn,I_part5,'MiniBatchSize',128);
[bboxes6,score6,~] = detect(app.Net.rcnn,I_part6,'MiniBatchSize',128);
```



```
[bboxes7,score7,~] = detect(app.Net.rcnn,I_part7,'MiniBatchSize',128);
[bboxes8,score8,~] = detect(app.Net.rcnn,I_part8,'MiniBatchSize',128);
[bboxes9,score9,~] = detect(app.Net.rcnn,I_part9,'MiniBatchSize',128);
%Computation of the result
 [rows1,~,~] = size(bboxes1);
 [rows2,~,~] = size(bboxes2);
 [rows3,~,~] = size(bboxes3);
 [rows4,~,~] = size(bboxes4);
 [rows5,~,~] = size(bboxes5);
 [rows6,~,~] = size(bboxes6);
 [rows7,~,~] = size(bboxes7);
 [rows8,~,~] = size(bboxes8);
 [rows9,~,~] = size(bboxes9);
Result = '';
if rows1 > 0 %If any object is detected in the image, its score is analaysed
    for i = 1:rows1 %Analysis of the score of each detected object for each sub-image
        if score1(i) > 0.9999
    %If the score of each detected object is greater than 0.9999 it is
    %added to the result
    Result = append(Result,'There is a key in the top left of the image');
        end
```



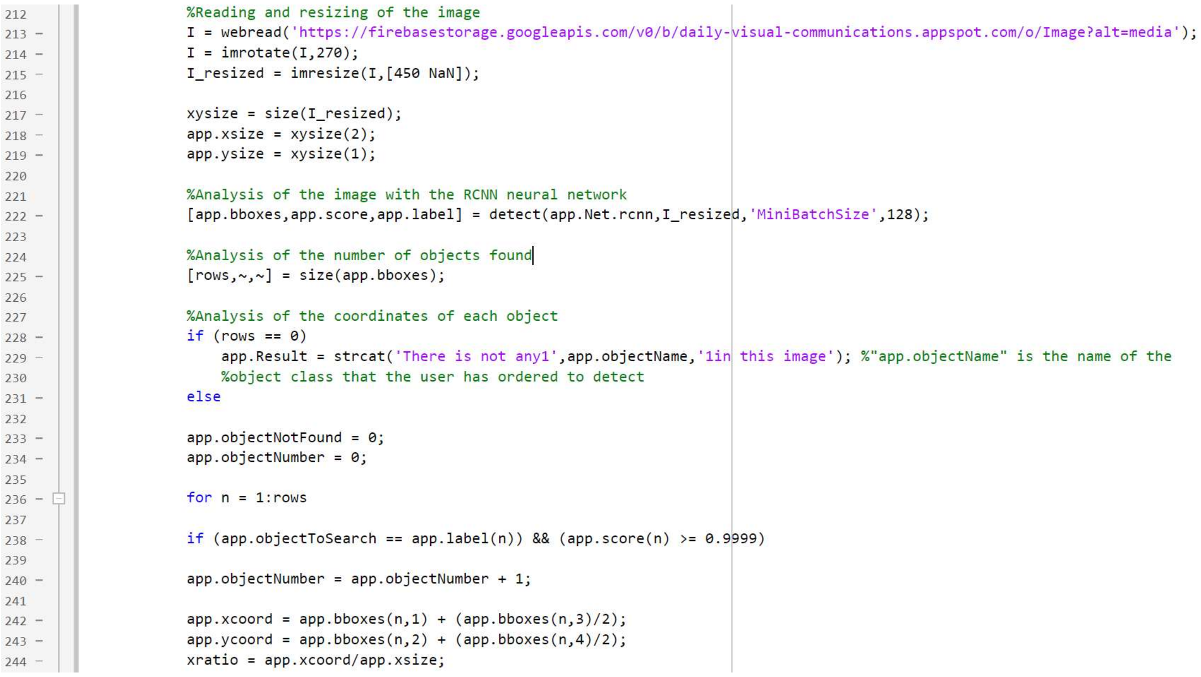
```
    end
end
if rows2 > 0
    for i = 1:rows2
        if score2(i) > 0.9999
    Result = append(Result,'There is a key in the top centre of the image');
        end
    end
end
if rows3 > 0
    for i = 1:rows3
        if score3(i) >= 0.9999
    Result = append(Result,'There is a key in the top right of the image');
        end
    end
end
```



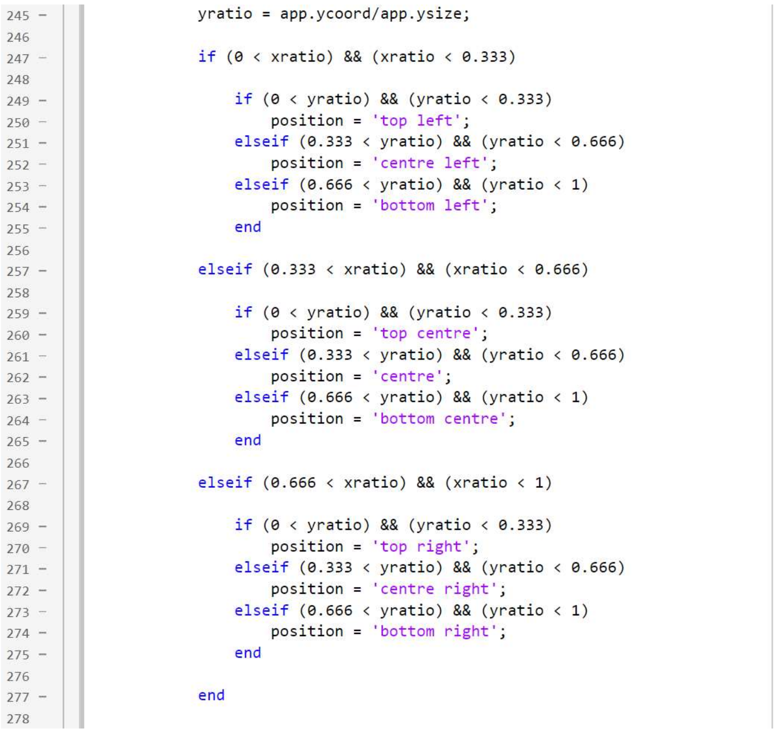
```
if rows4 > 0
    for i = 1:rows4
        if score4(i) > 0.9999
    Result = append(Result,'There is a key in the centre left of the image');
        end
    end
end
if rows5 > 0
    for i = 1:rows5
        if score5(i) > 0.9999
    Result = append(Result,'There is a key in the centre of the image');
        end
    end
end
if rows6 > 0
    for i = 1:rows6
        if score6(i) > 0.9999
    Result = append(Result,'There is a key in the top centre right of the image');
        end
    end
end
if rows7 > 0
    for i = 1:rows7
        if score7(i) > 0.9999
    Result = append(Result,'There is a key in the top bottom left of the image');
        end
    end
end
if rows8 > 0
    for i = 1:rows8
        if score8(i) > 0.9999
    Result = append(Result,'There is a key in the top bottom centre of the image');
        end
    end
end
if rows9 > 0
    for i = 1:rows9
        if score9(i) > 0.9999
    Result = append(Result,'There is a key in the top bottom right of the image');
        end
    end
end
if strcmp(Result,'') == 1
    Result = 'There is not any key in the image'; %Result corresponding to the null
    %detection of the object of interest in the analysed image
end
Result; %Final result
```

**Figure A.5.** Code for the segmented analysis of images

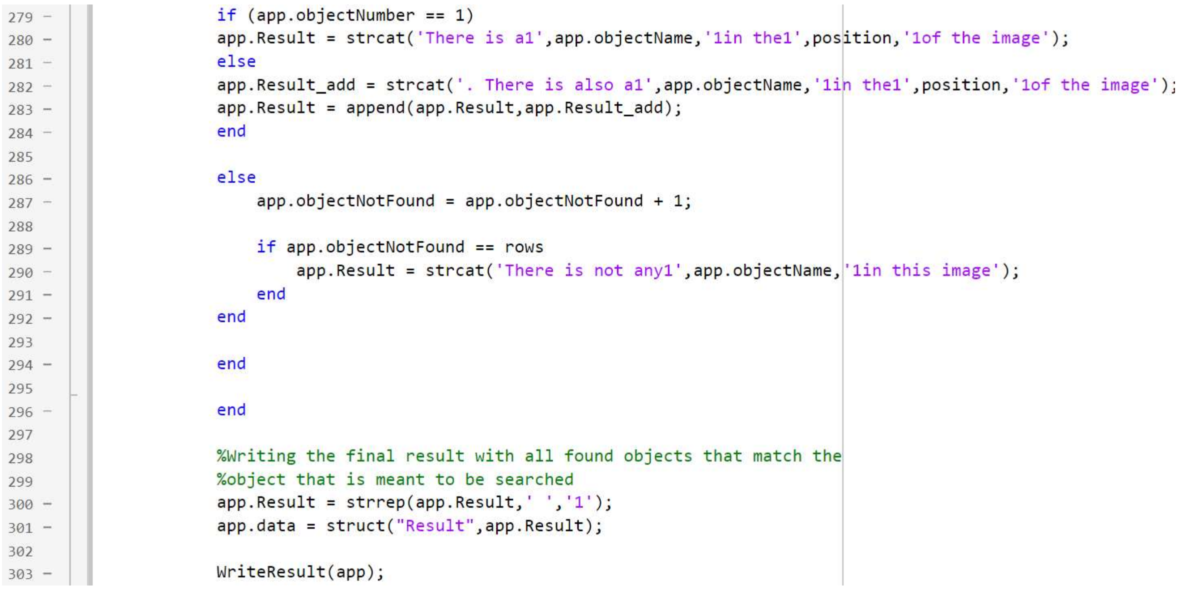
# A4. Matlab code that corresponds to the firstly developed object detection and location model, before the creation of the image subdivision system



```
%Reading and resizing of the image
I = webread('https://firebasestorage.googleapis.com/v0/b/daily-visual-communications.appspot.com/o/Image?alt=media');
I = imrotate(I,270);
I_resized = imresize(I,[450 NaN]);

xysize = size(I_resized);
app.xsize = xysize(2);
app.ysize = xysize(1);

%Analysis of the image with the RCNN neural network
[app.bboxes,app.score,app.label] = detect(app.Net.rcnn,I_resized,'MiniBatchSize',128);

%Analysis of the number of objects found
[rows,~,~] = size(app.bboxes);

%Analysis of the coordinates of each object
if (rows == 0)
    app.Result = strcat('There is not any1',app.objectName,'1in this image'); %"app.objectName" is the name of the
    %object class that the user has ordered to detect
else

app.objectNotFound = 0;
app.objectNumber = 0;

for n = 1:rows

if (app.objectToSearch == app.label(n)) && (app.score(n) >= 0.9999)

app.objectNumber = app.objectNumber + 1;

app.xcoord = app.bboxes(n,1) + (app.bboxes(n,3)/2);
app.ycoord = app.bboxes(n,2) + (app.bboxes(n,4)/2);
xratio = app.xcoord/app.xsize;
```



```
yratio = app.ycoord/app.ysize;

if (0 < xratio) && (xratio < 0.333)

    if (0 < yratio) && (yratio < 0.333)
        position = 'top left';
    elseif (0.333 < yratio) && (yratio < 0.666)
        position = 'centre left';
    elseif (0.666 < yratio) && (yratio < 1)
        position = 'bottom left';
    end

elseif (0.333 < xratio) && (xratio < 0.666)

    if (0 < yratio) && (yratio < 0.333)
        position = 'top centre';
    elseif (0.333 < yratio) && (yratio < 0.666)
        position = 'centre';
    elseif (0.666 < yratio) && (yratio < 1)
        position = 'bottom centre';
    end

elseif (0.666 < xratio) && (xratio < 1)

    if (0 < yratio) && (yratio < 0.333)
        position = 'top right';
    elseif (0.333 < yratio) && (yratio < 0.666)
        position = 'centre right';
    elseif (0.666 < yratio) && (yratio < 1)
        position = 'bottom right';
    end

end
```



```
            if (app.objectNumber == 1)
            app.Result = strcat('There is a1',app.objectName,'1in the1',position,'1of the image');
            else
            app.Result_add = strcat('. There is also a1',app.objectName,'1in the1',position,'1of the image');
            app.Result = append(app.Result,app.Result_add);
            end

            else
                app.objectNotFound = app.objectNotFound + 1;

                if app.objectNotFound == rows
                    app.Result = strcat('There is not any1',app.objectName,'1in this image');
                end
            end

            end

            end

            %Writing the final result with all found objects that match the
            %object that is meant to be searched
            app.Result = strrep(app.Result,' ','1');
            app.data = struct("Result",app.Result);

            WriteResult(app);
```

**Figure A.6.** Code for the analysis of images, previous to the apparition of the segmented analysis model