UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Egocentric Rich Images Detection in a Serious Game for Alzheimer's Patients

Author: Adrián Dueñas Peláez

Director:	**Gabriel de Oliveira**
Co-Director:	**Marc Bolaños**
Done in:	**Departament de Matemàtiques i Informàtica. UB**
Barcelona,	**22 de juny de 2017**

Abstract

Alzheimer's is a progressive neurodegenerative disease, right now irreversible and related with ageing. It is caused by neuronal deterioration which leads to memory loss, intellectual impairment and personality disorders and behavior. Several international reports warn that Alzheimer's is the most common form of dementia, affecting between 60% and 70% of the total number of cases. In addition, it has been estimated that at least 5% of the world's population with more than 60 years old will suffer the disease at some point in their life.

Despite the social and economic impact of the disease, there is still no treatment that can cure dementia or prevent its evolution. It is for this reason that efforts to find solutions that diminish the progressive development of the disease have been growing increasingly, building a society aware of these problems. The World Health Organization recognized dementia as a public priority in the report: "Dementia: a public health priority"[27].

The TV program L·La Marató de TV3"decided to dedicate the show of the year 2013 to the neurodegenerative diseases. Granted by it, this work has been aimed at reinforcing the inclusion of the new technologies in areas related with health to promote the improvement of Alzheimer's patients.

The framework developed in this project consists on the incorporation of a wearable camera that automatically takes pictures of the day of the patient. Subsequently, a selection is made, through the detection of objects in the images and a classification using Machine Learning algorithms, of those photographs which can bring value to the cognitive reinforcement of the patient. A serious game based on this pictures has been built to enhance this behavior.

Resumen

El Alzheimer es una enfermedad neurodegenerativa progresiva, por ahora irreversible y que está relacionada con el envejecimiento. Viene provocada por un deterioro neuronal que produce pérdidas de memoria, deterioro intelectual y trastornos de personalidad y comportamiento. Varios informes internacionales alertan que el Alzheimer es la forma más común de demencia, llegando a representar entre un 60% y un 70% del total de los casos. Además, se ha calculado que al menos un 5% de la población mundial con más de 60 años sufrirá la enfermedad en algún momento de su vida.

Pese al impacto social y económico de la enfermedad, aún no existe ningún tratamiento que pueda curar la demencia o evite su evolución. Es por este motivo por el que los esfuerzos en buscar soluciones que disminuyan el progresivo desarrollo de la enfermedad han ido creciendo más y más, construyendo una sociedad concienciada. La Propia Organización Mundial de la Salud reconoció la demencia como prioridad pública en el informe "Demencia: una prioridad de salud pública"[27].

La Marató de TV3, por su parte, decidió dedicar la gala del año 2013 a las enfermedades neurodegenerativas. Gracias a esto, en este trabajo se ha pretendido apostar por la inclusión de las nuevas tecnologías en ámbitos vinculados a la sanidad para favorecer la mejora de las personas de Alzheimer.

El marco de trabajo está formado por la incorporación de una cámara portátil que automáticamente saca fotografías del día a día del paciente. Posteriormente se realiza una selección, a través de la detección de los objetos que hay en las imágenes y una clasificación utilizando algoritmos de Machine Learning, de aquellas que puedan aportar un estímulo cognitivo al paciente. Además, se ha construido un juego serio basado en estas imágenes para intentar impulsar esta posibilidad.

Resum

L'Alzheimer és una malaltia neurodegenerativa progressiva, per ara irreversible i que está relaciona amb l'envelliment. Provoca un deteriorament neuronal que produeix pérdues de memória, deteriorament intel·lectual i trastorns de personalitat i comportament. Varis informes internacionals alerten que l'Alzheimer és la forma més comú de demència, arribant a representar entre un 60% i un 70% del total dels casos.

A més a més, s'ha calculat que almenys un 5% de la població mundial amb més de 60 anys patirà la malaltia en algun moment de la seva vida.

Tot i l'impacte social i econòmic de la malaltia, encara no existeix cap tractament que pugui curar la demència o eviti la seva evolució. És per aquet motiu pel que els esforÃ§os en buscar solucions que disminueixin el progressiu desenvolupament de la malaltia han anat creixent més i més, construint una societat consienciada. La própia Organització Mundial de la Salut va reconèixer la demència com a prioritat pública a l'informe "Demència: Una prioritat de la salut pública"[27].

La Marató de TV3, per la seva part, va decidir dedicar la gala de l'any 2013 a les malalties neurodegeneratives. Grácies a això, en aquest treball s'ha pretès apostar per la inclusió de les noves tecnologies a àmbits vinculats amb la sanitat per afavorir la millorar de les persones amb Alzheimer.

El marc d'aquest treball consisteix en la incorporació d'una càmera portàtil que automàticament pren fotografies del dia a dia del malalt. Posteriorment, es realitza una selecció, mitjançant la detecció dels objectes que hi ha a la imatge i una classificació utilitzant algoritmes de Machine Learning, d'aquelles que puguin aportar un estímul cognitiu al pacient. Per altra banda, s'ha construït un joc seriós basat en aquestes fotografies per intentar impulsar aquesta possibilitat.

Acknowledgments

I would like to thank Gabriel de Oliveira and Marc Bolaños for all the support they have given me during the semester. In the same way, I would like to thank Petia for trusting me to carry out this beautiful project and all the Rememory team for the great job they are doing trying to help many families to reduce the impact of a widespread disease.

Contents

Chapter 1

Introduction

Mild cognitive impairment (MCI) is a brain syndrome. People who suffer from it experience a progressive impairment in their cognitive and mental abilities such as memory or thinking, but they are not severe enough to interfere in their daily lives. Some population-based studies claim that it affects between 3% and 19% of people over 65 years.[12] Some of them remain stables or even return to normal but more than half of them progress to dementia within five years. Therefore, it can be considered a risk stage for dementia, so its early diagnosis and continuous control are fundamental.

Consorci Sanitari de Terrassa was able to promote new projects with the aim of improving the memory of Alzheimer's patients and their quality of life and also to help doctors with the treatment of this disease. To do this, some patients voluntarily were willing to hang a small wearable camera that would photograph different images (taking a photo every 30 seconds) related to their environment and their daily habits. These images will then be used to reinforce the cognitive capabilities of the patients through different activities. The reason for using images of their own life is because this produces a sensory stimulus for the patient and helps him memorize and recognize those scenes and people or objects appearing in them[6].

However, not all images can be used to carry out these activities as many will be blurred, others will be quite dark, etc. Thanks to Computer Vision we can develop algorithms which help to automatically analyze, understand and filter this large number of photographies. This analysis, in a concise manner, is necessary to make a classification between informative (or rich) and non-informative (or non-rich) images. That is, we need to separate those photographs that can represent a visual motivation for the patient: frequent places that he/she visits, familiar faces, especially sensitive objects, etc.

This classification is extremely important for the subsequent development of the project, since the evolution of the patients will depend on the rich images and will encourage the improvement of their memory. With the chosen photographs, we can built different serious games to stimulate and exercise the memory of the patient.

Our work methodology to achieve the inclusion of serious games as a medical instrument for the improvement of diseases has been, mainly, the creation of an algorithm that allows to discern between visually rich and non-rich images. This first step is one of the most important since it is the basis of the subsequent game that serves as a tool to doctors and patients.

In this work, we are going to develop a rich images detector algorithm which takes patients' egocentric images and selects those images that can stimulate his/her cognitive abilities. We achieved 80% of accuracy in our results. With this results we will also develop a serious games to let the patients exercise their memory. This instrument will support medical professionals and will have an impact on the previous methods, allowing the incorporation of technology in a field where it can give positive results, contributing to the social welfare in general, and to the optimization of processes in the treatment of Alzheimer, in particular. It allows a part of the computer vision discipline to be aimed to social projects with direct involvement on sick people allowing a more intuitive and accurate follow-up of the patient's evolution.

Chapter 2

State of the Art

Any image taken through technological devices encloses tones, colors, shadows and an endless shade that make it unique.

Currently, it is unusual to look for images by themselves, that is, by the content of the image itself. As an example, the world biggest and reference search engine: Google, is based on a description of the object we want to look for (text) and not on the graphic information itself.

2.1 Egocentric Vision

Lifelogging consists of a user continuously recording their everyday experiences via wearable sensors like cameras. This egocentric perspective allows observing the objects that the person carrying the camera looks through his eyes(see fig. 2.1 [10]. On the one hand, the data recorded offer the possibility to analyze and detect activities of daily living (ADL) or special activities, allowing to capture where, when and even how something happened. On the other hand, this data analysis also has the potential for knowing some personal habits of the wearer like behavior patterns or they could be also used to prevent diseases like obesity or depression[5] or cognitive and functional decline in elderly people[11].

Figure 2.1: Example of an egocentric image

However, visual lifelogs present a significant challenge for computer vision because of some uncontrollable variables like free motion of the camera, sudden changes in lighting conditions and image content, occluded objects or a large number of non-informative images that capture meaningless information such as walls, sky... Furthermore, visual lifelogs are increasing considerably fast so efficient methods to extract and locate relevant content are needed.

Current ADLs recognition in egocentric images invite to think that the key is the detection of the objects in the scene. However, although there have been some improvements in object recognition, it remains a challenge due to unconstrained scenarios[15].

2.2 Serious Games

Current technologies not only increase the implantation of games that make life easier for both doctors and patients in the improvement of their disease, but also stimulate them to share ideas, projects and results. By doing a simple search on the Internet we can find many projects with a common purpose: to improve the well-being of people with cognitive diseases.

A serious game aims to enhance learning through play, which allows people to acquire new knowledge or improve their skills with a more enjoyable method. Therefore, through gaming, it provides more value than just entertainment.

Some of the most common games are finding pairs in upside down images, repeat a series (this is very similar to Simon) or find the differences between to sequential images [1]. To go one step further, in this work it has been considered that including in those games the patients' own day-to-day photographs can provide a stimulus for their minds[1].

[1]https://lasaludi.info/juegos-de-memoria-para-los-pacientes-de-alzheimer.html

2.3 Object detections systems

We can find many image recognition systems such as face recognition and facial expression, signature recognition or character recognition.

Our hypothesis is that to detect a rich semantically image, the number and appearance of images matter (see 2.3). To identify objects more easily they can be divided into different categories, for example, whether they are manipulated or not, whether the object changes color or size, etc[17].

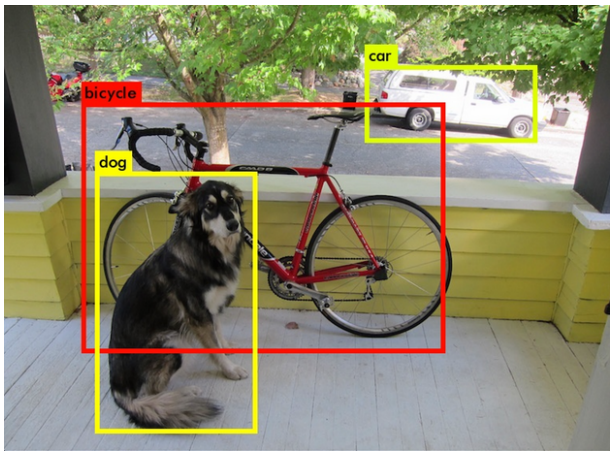

Figure 2.2: Image with its objects identified.

Some detections systems are:

- **Deformable parts models (DPM)**. It uses a sliding window to detect objects[7] and a disjoint pipeline to extract features, classify regions and predict bounding boxes. It is slow and less accurate than YOLO as it replace all independent parts with a single CNN.

- **R-CNN**. This model[9] and its variants (Fast R-CNN[8]) use region proposals to find objects in images. First, it generates potential bounding boxes [26], then applying a convolutional network extracts features and an SVM scores the boxes. Each part of this pipeline must be trained independently and consequently it is very slow.

- **Deep MultiBox**. Unlike R-CNN, it trains a CNN to predict regions of interest instead of using Selective Search but the main disadvantage is that it is not a complete detection system, it can not perform general object detection.

- **YOLO**. It replaces some independent parts of the prior detectors with a single CNN. It divide the image in a N x N grid and each cell is responsible for detecting the objects in them.

Given that, nowadays the mission is to achieve optimum results on a large scale in the shortest possible time. That is why alternatives like R-CNN[9] or Fast R-CNN[8] are difficult to use as they are relatively slow. We consider that YOLO is the best option it is able to detect objects almost in real time with a high accuracy.

Chapter 3

Rich Images Detection Algorithm

In this section, we are going to talk about the design of the algorithm itself that has been carried out in this work. It could be divided into 3 clearly differentiated parts:

Objects Detection: The first step of the algorithm is to detect objects in patients' images. It is a very important part as with them we can later measure the richness of the scene or find out if it is a familiar place for the patient.

Feature extraction: The second step is to analyze all the data obtained in the previous part and create a vector for each image that later will be the input of the classifier and will determine whether this is a rich or a non-rich image.
We considered that the number of objects, the variance of the color, whether a person is in the scene and the scale, confidence (of YOLO detection) and class of the objects detected are important features in this classification. We will detail this extraction later in the chapter.

Image classification: The last step is to create a proper classifier. First of all, it needs to be trained with the training subset. Once it is done, the classifier is ready to predict the richness of every image we will have as input.

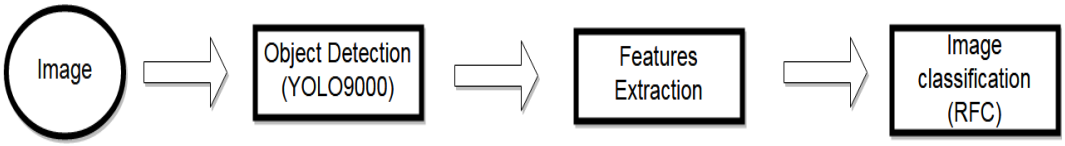

Figure 3.1: Algorithm for Rich Scene Detection

Each of these parts is detailed below.

3.1 Object Detection: YOLO9000

The first of the important technical parts is the detection of the objects that contain the different images (see fig. 3.3. The analysis of the different images is fundamental since the more objects we detect in it the richer the image is as its scene has more content and, consequently, we have much more data to base our decision on. In addition, this is a first filtering of patient's images as if we don't detect any objects or the objects detected have a very low detection confidence, this image is a candidate to be discarded. The results that we achieve here will have an important influence on the result of the classification. Some examples of the data that can be obtained are: the number of objects in the image, its type (if they are people, furniture, buildings ...), their respective locations and scale within the image or the most predominant color.

This task is time consuming. For this reason, we make use of YOLO9000 [18][19], an extremely fast and reliable object detection system. The base version can process images at 45 frames per second in real-time, but the fast one, is capable to run at 150 fps or even more. In table 3.1 we can see a comparison of its speed.

Many other detection systems make use of a sliding window over the image to detect possible objects in it. Others, like R-CNN[9] or Fast R-CNN[8] select potential bounding boxes within the image where there can be objects, then run a classifier only on these boxes and finally, a cleaning process is executed to improve bounding boxes and delete duplicates.

All these systems are somehow slow. YOLO9000 unifies all the independent components of the detection and uses a single convolutional neural network (CNN) to predict all bounding boxes from the whole image simultaneously.

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img
YOLOv2 (480×480)	77.8	59 FPS	16 ms/img

Table 3.1: Comparison of the speed of several Object Detection Systems

A CNN is a type of Artificial Neural Network that uses a variation of multilayer perceptrons (see fig.3.2). It was inspired by biological processes in which the connectivity pattern between the artificial neurons simulates the layout of the visual cortex. The receptive fields of different neurons (a restricted region of space) slightly overlap such that they cover the entire field. The response of neurons to stimuli within their receptive fields can be approximated by a convolution operation. In table 3.1 we can see how the CNN of YOLO is structured.

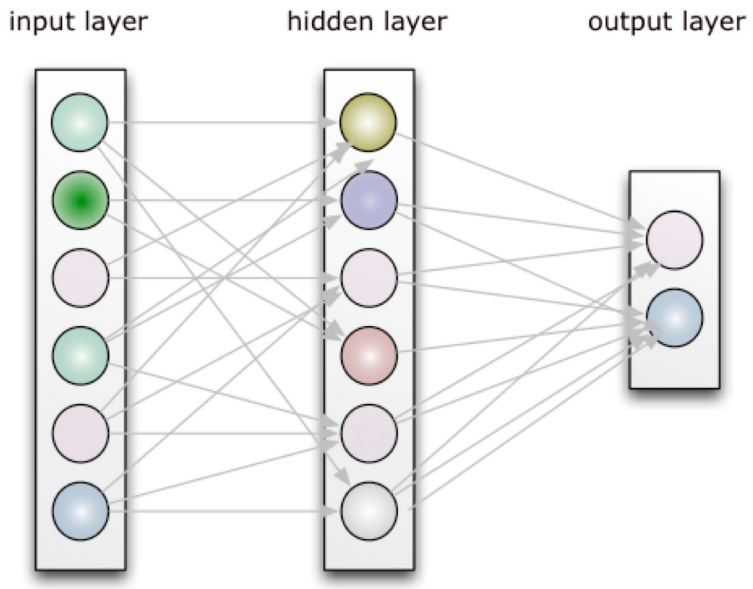

Figure 3.2: Example of a Neural Network

Unlike the other detectors, YOLO sees the whole image with all its context allowing to make more accurate detections with less errors. Firstly, it divides the image into an N x N grid. Then, it applies the Convolutional Neural Network to predict whether each grid

cell contains an object (see fig. 3.3). Finally, it filters all objects to obtain those that have a higher confidence of a certain threshold (by default the threshold is 0.25).

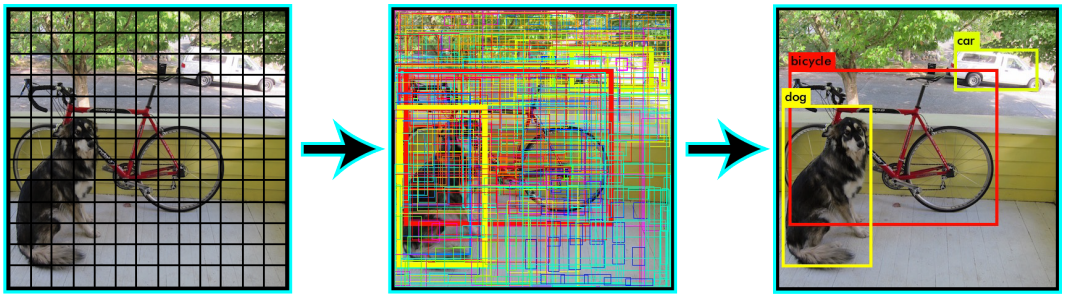

Figure 3.3: YOLO detection example

To find bounding boxes, YOLO predicts location coordinates relative to the location of the grid cell. This bounds the confidence to fall between 0 and 1 and then a logistic activation function is used to limit the network's predictions to be in this range. The network predicts 5 coordinates for each bounding box, $t_x, t_y, t_w, t_h, and t_o$. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box has width and height p_w, p_h, then the predictions correspond to:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

Figure 3.4: YOLO prediction

where b_x and b_y are the top left coordinates, b_w the width and b_h the height of the bounding box.

YOLO predicts width and height of the bounding box as offsets from cluster centroids and predicts the coordinates of the box relative to the location of filter application using a sigmoid function.

3.2 Features Extraction

Once YOLO is executed and we have all the objects within the image (with their position and size and the confidence of the detection), we should start the classification. But,

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 3.2: Model of the CNN used in YOLOv2.

right now, the classifier can not make any decision with this information; it does not understand it. Therefore, we need to transform and adapt it to its requirements. We have to build the feature vectors.

The input of this classifier must be a vector representing an image. Each position of this vector is determined by ourselves valuing what we think could be determinant for the right classification. Therefore, observing the images made by the volunteers of the project, we consider that these are very important features:

- Is there any person in this image?
- Is the image indoor or outdoor?
- Where are usually the objects, in the center or in the edges of the image?
- Is there a significant variation of the colour or it is a monochromatic image?

3.2.1 Feature vectors

With these questions in mind, for each image we save:

1. **Numbers of objects it contains.** The number can range from 0 to the maximum number of objects found in the image (no limit).
2. **Variance of color.** Like the previous one, from 0 when the image is of a single color and without a defined limit.
3. **Whether it contains people.** It can only be 0 or 1, 0 indicating there is no person and 1 that at least one. For this feature we chose some common people classes and checked if any object belongs to these classes.
4. **Object Scale.** It is a decimal between 0 and 1.
5. **Object Class.** Corresponds to the line number of the class name within the 9k.names file that varies between 1 and 9418.
6. **Object Confidence.** It corresponds to a percentage whose decimal value range between 0 and 1.

To get more information of images we made a pyramidal division of them: First, we save these features for the whole image taking into account the five most important objects of it. Then, we divide the image into 4 quadrants and get this same information (number of objects in every quadrant, color variance...) taking into account only the 3 most predominant objects of each quadrant. Finally, we make the same division and extraction but in this case for a 9-quadrant division getting the 2 most important objects of each quadrant.

The importance of objects is determined by a very simple formula:

$$i(O) = c(O) * s(O)$$

where c is the object confidence of the detection of YOLO and s is the scale of the object.

Altogether, the feature vectors have a length of 147 positions, the first 18 correspond to the data related to the whole image, the next 48 for each of the quadrants dividing the image into four and the remaining 81 when the division is nine.

Once the feature vectors of all the images have been obtained, they are placed in a matrix where each row corresponds to a different image and each column to each of the positions of the vectors (the resulting matrix will have a dimension of 10997 rows x 147 columns). Subsequently, each column is normalized. For this, maximum and minimum values of each column of the images belonging to the training set are needed. Then for each position the formula 3.1 is applied:

$$value_{normalized} = \frac{current\_value - min\_value}{max\_value - min\_value} \tag{3.1}$$

As we have been only considered the training set when finding maximum and minimum values, it is possible that in test or validation sets there are images with a higher or lower value than these two. In these cases, we force each column to only range between 0 and 1.

In chapter 5 of this work we analyze two other versions of feature vectors implemented that gave slightly worse results.

3.3 Classifying images

3.3.1 Random Forest Classifier

To implement our work we propose using a Random Forest Classifier (RFC)[4][14], a very effective method for classifying variables or objects. They are a combination of decision trees, which are represented as a tree graph that given an input, it made a series of decisions and depending on each answer, the algorithm descends for one branch or another. The leaves nodes determine which will be the classification of the input so, every decision is many important as they define the way of descent and therefore, the final classification.

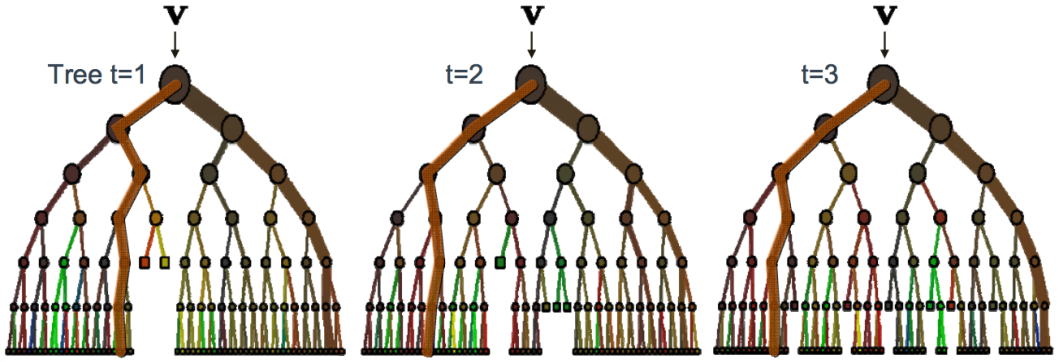

Figure 3.5: Example of a RFC with several decision trees.

The output probability of the ensemble model is:

$$p(c|v) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|v)$$

where c is each of the possible resulting class and v the input.

RFC require very little learning time and, moreover, they are very accurate. To train the classifier, first, we need to choose a training set and then different decision trees are created which will be trained with a random selection of elements of the training set or a

combination of them (with repetitions). This allows the classifier to be stronger and, very important, avoid overfitting.

Once the classifier is built, if we want to classify an image, we must apply the RFC on it and select the most common output class among all the trees.

Chapter 4

Implementation

In this chapter, we are going to define which technologies we have used, the libraries imported to get new functionalities and whether we used an IDE or any other alternative.

4.1 Technologies and Dependencies

For the part of object detection, we used YOLO9000, which is programmed in C. Although we have not programmed it, it has been necessary a modification in its code to adapt it to our needs[1]. In addition, since it is expected to work with a large number of images, its execution has been parallelized. For this reason, it is also necessary the package 'GNU parallel'[2] with all its dependencies.

For the second part, the data extraction and classification of the images have been implemented with IPython as it is easy to read and allows an agile and robust development. It let us do many tests and compare its results. Within the code the most important libraries that have been used are:

scikit-learn[16] [3] For the classifier as it contains very interesting packages for machine learning like Random Forest Classifier or Support Vector Machines among others.

NumPy [4] A very important package for scientific computing with Python. It contains among other things: powerful N-dimensional arrays and sophisticated math functions.

pandas [5] It provides powerful easy-to-use data structures and data analysis tools for Python. Widely used for statistics and datascience.

[1] This modification is explained in detail in section 4.2.1
[2] GNU: 2017 [En línea]. <https://www.gnu.org/software/parallel/> [Consulta: 12 de junio]
[3] http://scikit-learn.org/stable/
[4] NUMPY DEVELOPERS: 2017 [En línea]. <http://www.numpy.org/> [Consulta: 12 de junio]
[5] NUMFOCUS: 2017 [En línea]. <http://pandas.pydata.org/> [Consulta: 12 de junio]

cv2 (OpenCV) [6] Very interesting library for computer vision and machine learning.

plot.ly [7] It is a platform to make amazing charts and visualize data. During the tests, it was especially good when trying to compare scores between the different classifiers as we needed to make a 3D-chart and it let us navigate and move through them.

To save all the code, both YOLO and IPython, we have used GitHub, the Versions Control System (VCS) par excellence.

4.2 Executing YOLO9000

The first thing we need to do is download and compile YOLO9000[8]. In addition, we need to download the pre-trained weight file that we can download from the YOLO website[9]. This file is the result of the pre-training of the model and indicates the weights of the different convolutions within the neural network.

Now, we can execute YOLO. It can be done with the following command:

```
./darknet detector test cfg/combine9k.data cfg/yolo9000.cfg yolo9000.weights
    data/dog.jpg -thresh 0.1
```

Where:

- combine9k.data is the file that tells YOLO where the information related to the classes is like the total number, their names, labels, etc.
- yolo9000.cfg is the configuration file of YOLO in which there are indicated among other things the convolutions that it applies to the image.
- yolo9000.weights is the file mentioned above.
- data/dog.jpg is the image from which we want to detect objects.
- The parameter 'thresh' is optional and indicates the minimum confidence of the objects that YOLO will find. If we do not pass this parameter, as we mentioned before, it is 0.25 default.

There is another way to execute YOLO9000, which is very useful if we want to detect objects in multiple images. This is identical to the previous, but without passing the image path. In this case, YOLO will prompt us the paths of images one by one.

In this project, we have used the first one, but with a slight modification:

[6] OPENCV TEAM: 2017 [En línea]. <http://opencv.org/about.html> [Consulta: 12 de junio]
[7] PLOTLY: 2017 [En línea]. <https://plot.ly/> [Consulta: 16 de junio]
[8] https://pjreddie.com/darknet/yolo/
[9] https://pjreddie.com/media/files/yolo9000.weights

```
find $1 -name '* .jpg' | parallel -j+1 ./darknet detector test cfg/combine9k.data
    cfg/yolo9000.cfg yolo9000.weights {} -thresh 0.1
```

In this command, we first run find to list all the patient's images and redirect the output to the YOLO input, parallelizing its execution and allowing to detect objects of multiple images simultaneously. This exeution is really fast when trying to detect objects in a large number of images.

Once YOLO has detected all objects above a given threshold, it shows them in two possible ways:

1. If YOLO has not been compiled with the OpenCV library, it creates predictions.png file with the image and the bounding boxes for each object drawn on it. In figure 3.3 we can see what YOLO prints out in the image.
2. If it has been compiled the mentioned library, it shows us in a separate window the result that previously saved in a file.

Neither of these two alternatives is useful since the algorithm needs the position of each object and its scale. That is why a modification in the YOLO code has been necessary for this particular use.

4.2.1 YOLO output

The function that displays the results is called draw_detections() which is located in the image.c file. We commented the part where it shows the results and we forced YOLO to print the detections on the screen. In this case, the data that it prints is: the name (path), width and height of the image and the position, width, height, confidence and class for each object YOLO has found. In figure 4.1 we can see the new YOLO output for one image.

Subsequently, when we run YOLO this output will redirect to a file, trying to follow the JSON format.


```
"data/dog.jpg": {
    "w": 768,
    "h": 576,
    "object": [
        {"x": 69, "y": 77, "w": 30, "h": 40, "confidence": 0.252034, "tag": "artifact"},
        {"x": 417, "y": 70, "w": 290, "h": 102, "confidence": 0.739928, "tag": "truck"},
        {"x": 659, "y": 116, "w": 42, "h": 49, "confidence": 0.101576, "tag": "car"},
        {"x": 144, "y": 140, "w": 132, "h": 107, "confidence": 0.143146, "tag": "bicycle"},
        {"x": 112, "y": 124, "w": 453, "h": 331, "confidence": 0.258882, "tag": "bicycle"},
        {"x": 117, "y": 227, "w": 227, "h": 301, "confidence": 0.814127, "tag": "feline"}
    ]
}
```


Figure 4.1: Example of the output with the modification we made

4.3 Running the classifier

Once all the necessary data has already been obtained, we can start predicting the class of the images.

First, the classifier needs to be declared. This is achieved with the following command:

```
rfc = RandomForestClassifier(n\_estimators=n, n\_jobs=m)
```

Where n is the number of trees we want to declare and m is the number jobs to run in parallel, for both train and predict.

Once it is created, it must be trained (or fit) to find relationships between features. But we can not do this with the whole dataset as it would create a "perfect" classifier for these particular images but we do not know its behavior with any image that will come afterwards. For this reason we need to train this classifier with a subset of our data and check its score with the other part of it (we can know this predicting the class of the images and comparing results with the real classes). This is done with the following code:

```
rfc.fit(X_train,Y_train)
```

where rfc is the classifier mentioned above, X_train corresponds to the feature vectors of the training set and Y_train corresponds to the real class of each image after the manual classification done by the doctors. This way, the classifier can find which are the main features knowing in advance the result of the classification of the training set images.

At this point, the classifier is ready to start predicting the richness of any image we pass as input. To do this, the following command is executed:

```
rfc.predict(X_validation)
```

where X_validation is the feature vector of the image whose class we want to predict. It returns the result of the predictions that, in our case, is 1 (it is probably a rich image) or 0 (it is not).

Chapter 5

Results

Once the algorihm is defined, the next step to look for the best result adjusting variables like the threshold of YOLO or the number of decision trees in RFC or making an analysis of the images to deduce what can be an important features to make a good classification.

5.1 Dataset

5.1.1 Egocentric Images

To create any algorithm, it is necessary to have a good dataset with a sufficiently large and representative sample of images to find patterns and common features. This will be used to train the classifier and to find which are the most important feature and which of them are more decisive to determine the final decision (rich or non-rich).

In this case, the dataset used is composed of photographs taken by the different members of the research team on their day to day over a period of two weeks.

As we said before, the wearable camera takes a picture every thirty seconds automatically (although you can force an image at a certain time if it is deemed appropriate). Therefore, considering that it is a seven-people sample and that on average each camera takes 1500 images per day approximately, we have a dataset of 10997 photographs (6399 rich and 4598 non-rich).

To favor the creation of an algorithm to detect which images are rich or not, we have had the collaboration of several doctors of the department in charge of the treatment of MCI patients. These professionals have performed a manual classification of all images in the dataset. The task developed has been essential for the design of the automatic classifier. Without it, it would not be possible to carry out the training phase and therefore, neither extract the relevant information to find patterns or detect which characteristics are the most determinant for subsequent classification.

The following images are examples of egocentric rich images (see 5.1) and non-rich images (see 5.2). We observe that rich photographies show known people or recognizable places. However, non-rich images are meaningless or dark images (that can hardly be seen) that it is difficult to recognize anything.

Figure 5.1: Rich images

Figure 5.2: Non-rich images

5.1.2 Data split

The dataset must be split into three subsets to allow the realization of each phase of the algorithm.

As we said before, the images belong to a period of 15 days. Due to this, and since the photographs taken during the same day can be very similar, we proceeded to randomly separate the different days into three different sets.

1. Firstly, the training set consists of 60% of the days, in this case 9. This set will be used during the training phase of the classifier. From these images, the classifier will draw all the decisions.
2. Secondly, 20% of the days, in this case 3, are part of the validation set. This set is used to evaluate the different trainings performed with the images of the previous group. The purpose of this set is to evaluate the performance of the different models and get the one that has a greater success percentage (precision and recall).
3. Finally, the remaining 20% belongs to the test set and this is the final evaluation of the classifier that has obtained the highest percentage of success with the previous set. Its result is an estimation of how good it can become when used in a real-life scenario.

5.2 Evaluation Metrics

In order to evaluate the different results and compare them to get the best one, we make use of the f1-score (or f-measure) [21] metric.

5.2.1 F1-score

This metric considers Precision and Recall[25] to make a weighted average of them getting its best value at 1 and the worst at 0:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} = 2 * \frac{precision * recall}{precision + recall}$$

where *precision* is the quotient between the number of True Positives objects and the number of predicted positive elements; and *recall* is the quotient between the number of True Positives objects and the number of real positive elements.

To get this score in our code we imported this metric from *sklearn* and computed it with the following command:

```
f1_score(Y_validation,prediction_rf)
```

where $Y\_validation$ are the real classes of the images in validation set and $prediction_rf$ are the predicted classes of the classifier for the same set of images.

5.3 Grid search

One of the first parameters that we can adjust is the threshold of YOLO. As the detection of objects in images compared to other actions is slower, we avoided the execution of YOLO for several thresholds making a single execution of YOLO with the threshold 0.01 in order to obtain the maximum possible number of candidates to be objects. This way, we have executed YOLO only once and it serves us for all the tests that we do.

Subsequently, the data would be filtered to simulate the execution of YOLO with the following thresholds: 0.01, 0.05, 0.1, 0.15, 0.25, 0.35 and 0.5, and each result will be saved in a different file that has the JSON format allowing a fast read with the pandas library saving this data in a dictionary.

Once we have all the possible objects detected, we calculated the feature vectors for each of the previous thresholds saving each set in a different DataFrame and in a different file so we did not have to calculate them again. With this we obtained 7 different feature vectors for each image (1 for each threshold).

With all feature vectors of all the images computed, we proceed to classify the images. For this, different classifiers with different number of estimators were trained following the commands indicated in section 4.3. In our case, $m = 4$ (to take profit of the multi-core processor) and to find the best classifier we have executed it for the following values of n:

- From 10 to 100 with a 10-value step $(10, 20, 30, \ldots, 90)$.

- From 100 to 500 with a 20-value step $(100, 120, \ldots, 500)$.

The result of the score for each classifier for the threshold 0.01 (which gave us the best scores) is indicated in the figure 5.3:

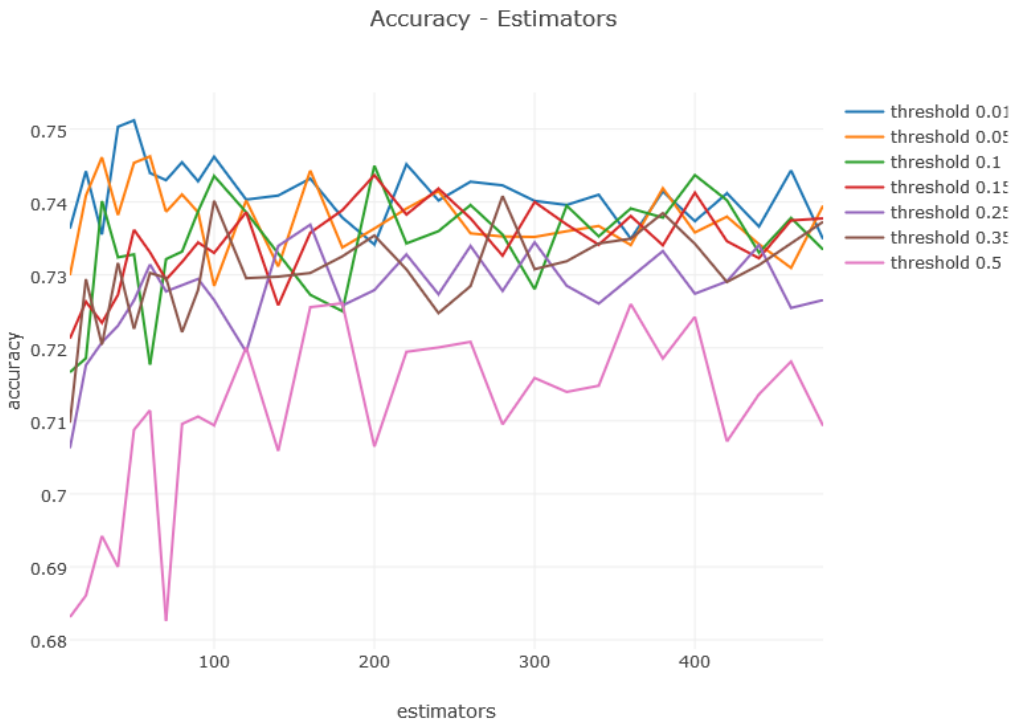

Figure 5.3: Results of predictions

where the accuracy is the result of evaluating the metric f1-score for each classifier.

We note that for 50 estimators we have given the maximum score that is around 0.751. Consequently, the score of this classifier is checked for the test set, which gives about 0.791. Therefore, we can conclude that after all the tests performed, the classifier with 50 estimators and threshold of YOLO 0.01 is the one that gives us the best results.

If we observe some results of the classifier we see how the images in figure 5.5 have been correctly classified as rich images and the images in figure 5.4 as not rich. In rich photographies appear clearly recognizable people, which is very positive for the patient. On the other hand, negative ones are made to the sky and they do not bring any useful information to the patient.

Figure 5.4: Examples of True Negatives images

Figure 5.5: Examples of True Positives images

However, looking at images in 5.6, which were misclassified (they are considered non-rich by the doctors but they were classified as rich), we observe that in the first one it may be the patient's living room and YOLO detected a lot of objects but they are not important enough to create a meaningful scene. In the second one, the body of a person person appears but he is not recognizable as his face is occluded. We strongly think that to solve this misclassification we can add to our algorithm other computer vision systems like face detectors or we can also weight the classes to avoid the fact that all the classes have the same importance.

Figure 5.6: Examples of False Positives images

In the last case, in figure 5.7 we have False Negatives images, in the ground truth they are Positives but were classified as Negatives. In the first one, we see someone's living room with his/her curtains, the iron, etc. And in the second one we can see a store window. In both images YOLO did not detected as objects as prior images. We think again that weighting objects may be a solution for these situations.

Figure 5.7: Examples of False Negatives images

5.4 Word2Vec

Apart from the previous tests, different tests have been carried out to try to improve the results. All of them are based on WORD2VEC, a 300-position vector database created by Google, which represent words in a way that they are close in this 300-dimensional space if their meaning is related.

The first step to use this new feature was to get the vector of all classes that can detect YOLO (which were 9418 classes) and those classes that are not present in its vocabulary substitute their vector for the vector of a neutral class, in our case, "thing".

5.4.1 Similarities

An advantage that gives us WORD2VEC is that it allows us to obtain the similarity between two words depending on how close their vectors are. This is achieved by the following formula:

```
model.similarity(word1,word2)
```

which returns the cosine distance between the two vectors.

Therefore, we believe that the feature which says if there is a person in the image (or quadrant) could be replaced by this similarity since it is a more real approximation to what we are looking for.

After replacing this feature, recalculate all feature vectors and re-predict the class of them, we have obtained a slightly lower result than those discussed previously. In figure 5.8, we can see the scores for the same classifiers as before but with the new version of feature vectors.

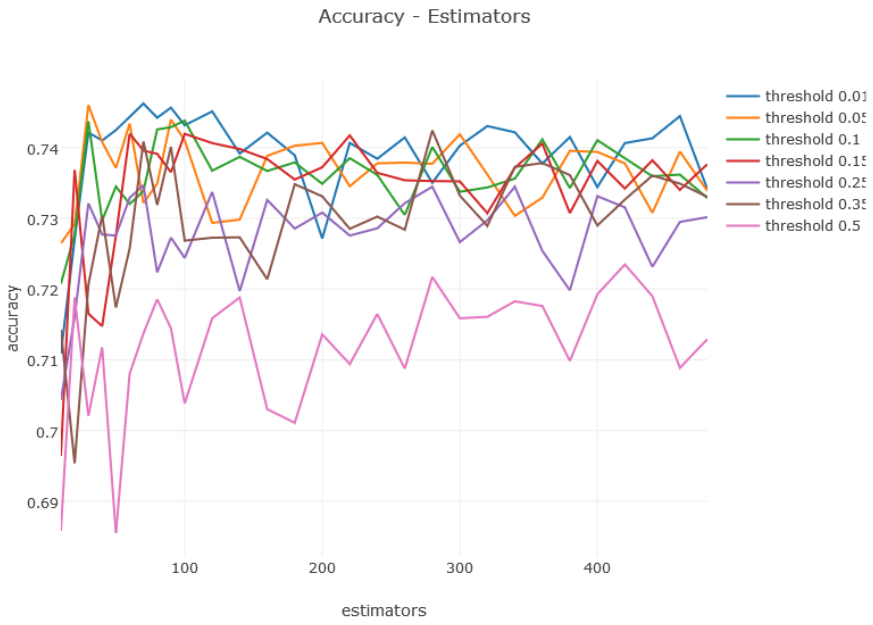

Figure 5.8: Results of predictions for the second version of feature vectors.

We can see that in this case the best score is 0.746 with 70 estimators, a slightly worse value than before. For the test set, the score for this concrete classifier is about 0.781. Although it is a lower score, it's a very small variation (only 0.01), so we can not be sure if this new version of feature vectors is worse than before.

5.4.2 Class vectors

The second test is based on the same database (Word2Vec) as the previous one but replaces a different part on the feature vectors. Besides the similarity mentioned above, we also substitute the class of the most predominant objects (in the image and in each quadrant) by its WORD2VEC vectors. As it is not practical to directly replace the 300-positions vectors (since we would have vectors of more than 10,000 positions), we applied PCA to reduce the dimension of these vectors by losing the minimum possible variance and, therefore, maintaining the greater meaning possible. It has been tested for different length values: 1 and 20. But in both cases it gave us worse results. We think that this scores are a consequence of the fact that the most important features before have lost their importance as now this vectors are too long.

Again, all feature vectors have been recalculated and the classes of the validation images have been predicted and as in the previous case. These are the results:

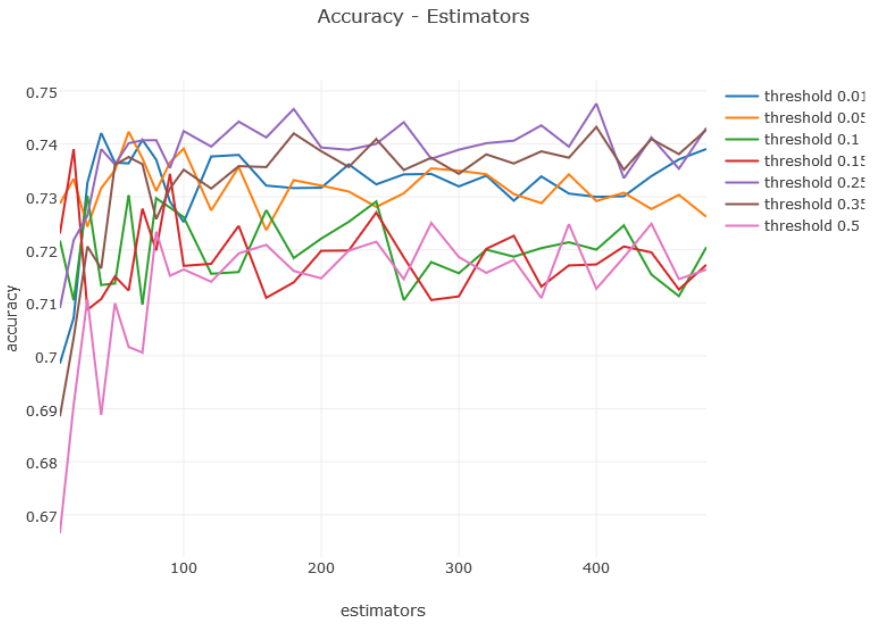

Figure 5.9: Results of predictions for the third version of feature vectors.

We can see now that the best score is reached for 400 estimators and threshold of

YOLO 0.25: 0.747. Again, this is a slightly worse value than before but in this case, for test set, we got 0.752 wich is a reasonably variaton to afirm that in this case, this version is worse than the first one.

In addition to the tests explained in this chapter, we have done the comparison with Support Vector Machines (SVM) doing a Grid Search on the variables *C* and *gamma*. As summary, in the following table we try to show the results:

	Precision	**Recall**	**f1-score**
RFC	0.79	0.79	0.79
RFC + Word2Vec	0.78	0.78	0.78
RFC + Word2Vec + PCA	0.74	0.75	0.75
SVM	0.68	0.67	0.68

Table 5.1: Comparison of the results

Chapter 6

Applications: Serious Games

In this chapter we are going to see two applications for our Rich Images Detector:

6.1 Simon

Simon is a memory game created in the year 1978 with the aim of following a sequence of colored lights that lit randomly and whose difficulty is increasing as levels are advanced.

In this project, we have considered its inclusion as a serious game, although with slight variations from the original as for example the replacement of the lights for personal egocentric images, to stimulate the memory of the patient.

To do this, we have created an application for Android tablets where the patient has to observe a series of images. All of them belong to their daily life that have been taken by a wearable camera and then processed with our algorithm in order to select the rich images.

The activity has been structured in a total of 9 levels with increased difficulty and 3 main screens. The first one indicates the instructions of the game and when the patient has read them, he must click on the CONTINUE button. Then it moves to the next screen, where a grid appears with nine white photographs. Subsequently, with a rate of 1 per second between each of them, a number of images (the higher level, the more images) will rise. See figure 6.1

When the series is finished, the images will be placed backwards again and the patient can start answering pressing them in the same order in which they appeared previously. If the patient did not start answering in 5 seconds, the answer in considered wrong. In case the patient responds well, he will be given the feedback of "RIGHT!" and can move to the next level; otherwise, he can try again (see Fig. 6.2) and if it fails again, he will be given the correct answer and the same process will be repeated, but with new images.

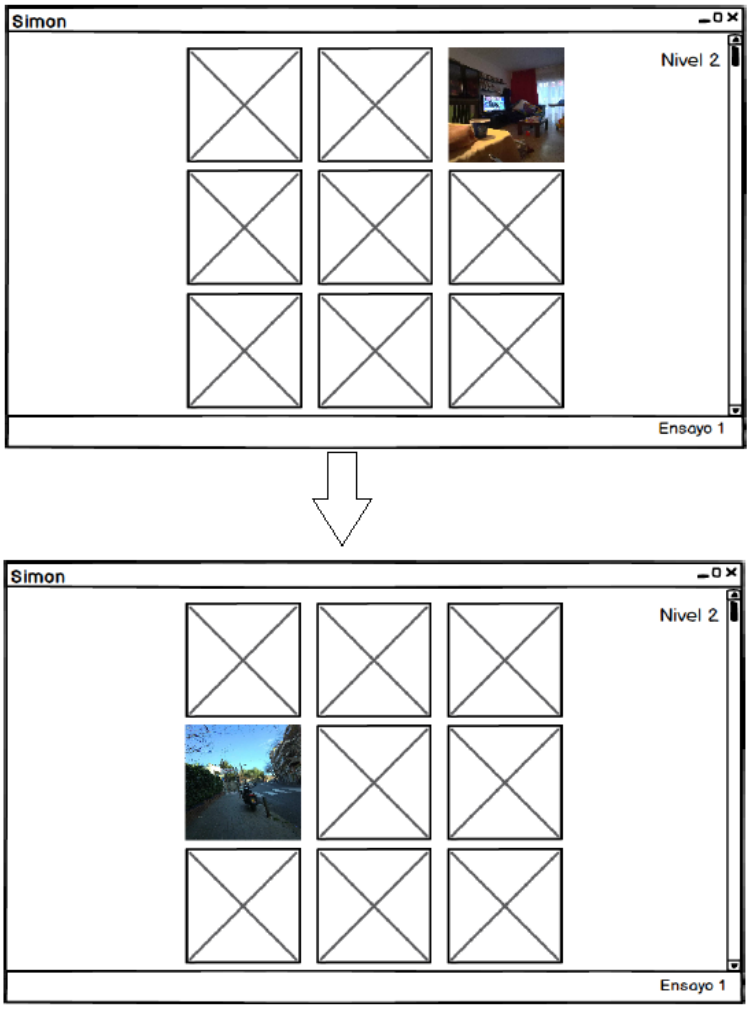

Figure 6.1: Example of the screens that shows the series for level 2.

If the patient answers wrong after 4 attempts, it goes back one level (if he is not in the first one, case where the game would terminate) and the game remains the same. If during the remaining game he fails again any level, the game will be finished (he can only go back one level).

The game ends when the player fails two levels or if he pass the last level, the 9th. At the end of the game, it shows a table with the percentage of hits in each level, whose overall result is translated into neurons won. See figure 6.3.

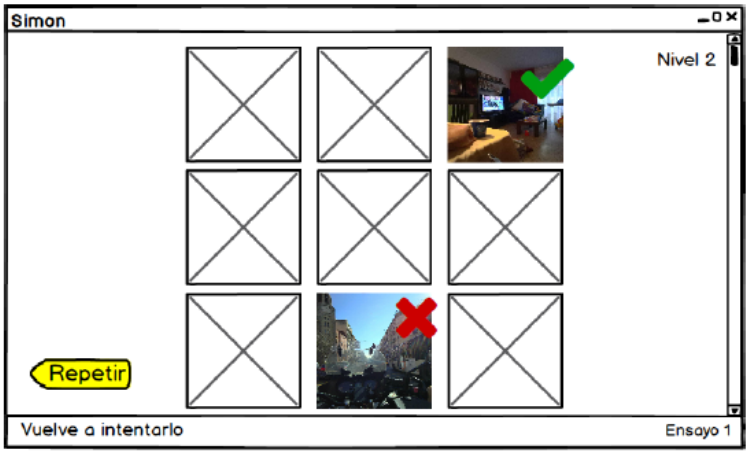

Figure 6.2: Example of the screens where the patient gets wrong.

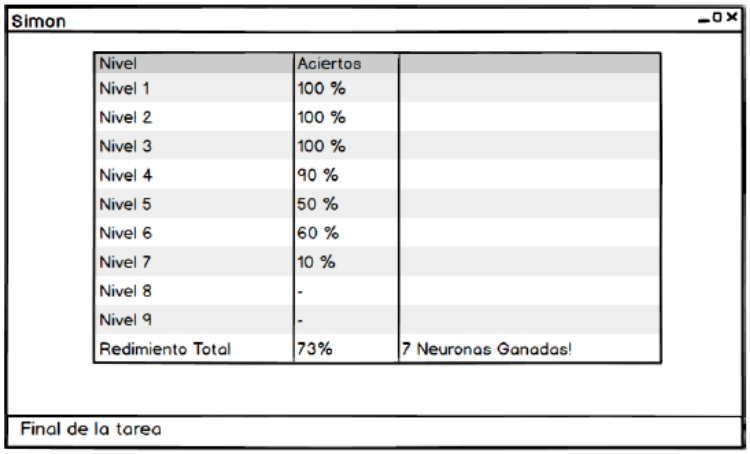

Figure 6.3: Example of the final screen.

6.2 Rememory-Back

Rememory-Back is also a memory game. It is like Simon but here the patient should not remember any sequence of images; instead, he should only observe an image. The importance of using rich images is that these can stimulate the patient and he may also feel more motivated to play and therefore will exercise more his memory. The mechanics is very simple:

The first screen tells him the instructions of the game and in the second the patient is informed that before starting the game there are some practice examples that will serve

to understand the logic of the game. To start, the patient must select his preferred level of difficulty (Level 1 or Level 2).

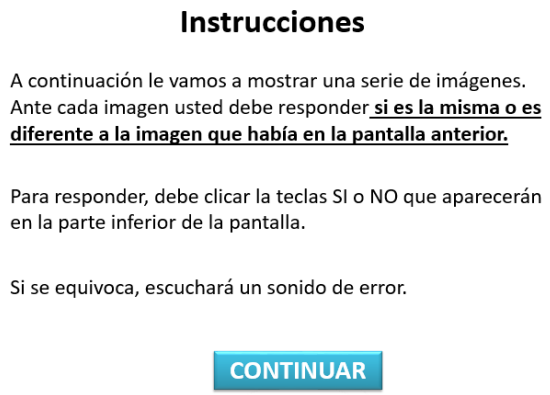

Figure 6.4: Example of a Rememory-Back screen.

Next, an image will be shown to the patient which he should observe and try to memorize the maximum number of characteristics (which people appear, where it is, some of the objects that appear...) and after some little time, a second image will be shown and the patient must respond by pressing YES or NO if it is the same image or a different one.

Figure 6.5: Example of a Rememory-Back screen asking the patient's answer.

The same process is performed with the third image (in which the patient must indicate if it is the same as the second) and later.

When the practice is finished, the patient is told that from now on the game begins. Each time the patient responds incorrectly, a sound is played to indicate that he or she has been mistaken. At the end of the game, the patient will see how many points she won with this activity.

Currently, the project Rememory is acquiring data from patients wearing the camera for 2 weeks. Then the team of the University of Barcelona will process their data and prepare images for different cognitive exercises. There is when Simon will be implemented to be applied on the patients in order to re-live their experience.

Chapter 7

Conclusions

Thanks to the wearable cameras, it is possible to capture the day-to-day activities of patients with MCI. This implies that after a considerable period of time, we face a large number of images. These photographs provide an important cognitive stimulus to the patient, since the image is taken from an egocentric point of view and allows the patient to relive an event from the past. Unfortunately, these images have some disadvantages such as their limited field of view or free motion of the camera, which causes to take bad images.

In this project, we have helped doctors in their work to diagnose and treat a disease as complicated as Alzheimer's by automating the process of filtering these photographs by discarding those that have no meaning (such as those taken to a wall or to the sky) and selecting those images that may favor the exercise of patient's memory.

This classification is achieved through an object detection process (performed using YOLO9000[19]) and a subsequent analysis of this data with Machine Learning technologies (feature extraction and Random Forest Classifier).

This classifier system has many applications. In chapter 6 we mentioned two alternatives but the possible applications are very diverse and can help automate many processes and be a part of many projects that require human interaction with images as they are meaningful photographies which ease the attention of the user and help exercising some capabilities like memory in our case.

Future Work

The results obtained in the two tests performed gave worse than the original one. They were supposed to give better results since they have replaced features with others that give more meaning or that favor a more similar feature value when the concept they represent is related. Therefore, we believe that this part of the project is a matter of study. Some possibilities are:

- Look for another database different to WORD2VEC which may give better results in our particular project.
- Explore with a wide battery of classifiers.
- Find weights to YOLO detections to give some importance to some classes.

An important future step is to integrate in the final application of Rememory and apply it on the patients in order to check the hypothesis for the intervention of mild cognitive impairment, to be done in September 2017.

Bibliography

[1] ALCALÁ MANGAS, M.E. y VALENZUELA SŃCHEZ, E. *El aprendizaje de los mayores ante los retos del nuevo milenio*. Dkinson, Madrid, 2000. P. 25-60.

[2] BOLANOS, Marc; DIMICCOLI, Mariella; RADEVA, Petia. *Toward storytelling from visual lifelogging: An overview.* IEEE Transactions on Human-Machine Systems, 2017, vol. 47, no 1, p. 77-90.

[3] CIRESAN, Dan Claudiu, et al. *Flexible, high performance convolutional neural networks for image classification*. En Twenty-Second International Joint Conference on Artificial Intelligence. 2011.

[4] CRIMINISI, A.; SHOTTON, J.; KONUKOGLU, E. *Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning.* Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114, 2011, vol. 5, no 6, p. 12.

[5] DOHERTY, Aiden R., et al. *Wearable cameras in health.* American journal of preventive medicine, 2013, vol. 44, no 3, p. 320-323.

[6] DOHERTY, Aiden R.; MOULIN, Chris JA; SMEATON, Alan F. *Automatically assisting human memory: A SenseCam browser.* Memory, 2011, vol. 19, no 7, p. 785-795.

[7] FELZENSZWALB, Pedro F., et al. *Object detection with discriminatively trained part-based models.* IEEE transactions on pattern analysis and machine intelligence, 2010, vol. 32, no 9, p. 1627-1645.

[8] GIRSHICK, Ross. *Fast r-cnn.* In Proceedings of the IEEE International Conference on Computer Vision. 2015. p. 1440-1448.

[9] GIRSHICK, r., DONAHUE, J., DARRELL, T. and MALIK, J. *Rich feature hierarchies for accurate object detection and semantic segmentation*. In CVPR, 2014.

[10] GRÉGORY ROGEZ, James S. and SUPANCIC, Deva Ramanan. *Understanding Everyday Hands in Action from RGB-D Images*. IEEE International Conference on on Computer Vision (ICCV) 2015.

[11] JINDA-APIRAKSA, Amornched; MACHAJDIK, Jana; SABLATNIG, Robert. *A Keyframe Selection of Lifelog Image Sequences.* En MVA. 2013. p. 33-36.

[12] GAUTHIER, Serge, et al. *Mild cognitive impairment.* The Lancet, 2006, vol. 367, no 9518, p. 1262-1270.

[13] HO, Tin Kam. *The random subspace method for constructing decision forests*. IEEE transactions on pattern analysis and machine intelligence, 1998, vol. 20, no 8, p. 832-844.

[14] HO, Tin Kam. *Random decision forests*. In Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. IEEE, 1995. p. 278-282.

[15] NEBEL, Jean-Christophe. *Recognition of Activities of Daily Living with Egocentric Vision: A Review*. Kingston University, 2016.

[16] PEDREGOSA, Fabian, et al. *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 2011, vol. 12, no Oct, p. 2825-2830.

[17] PIRSIAVASH, Hamed; RAMANAN, Deva. *Detecting activities of daily living in first-person camera views*. En Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012. p. 2847-2854.

[18] REDMON, Joseph, et al. *You only look once: Unified, real-time object detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. p. 779-788.

[19] REDMON, Joseph; FARHADI, Ali. *YOLO9000: Better, Faster, Stronger.* arXiv preprint arXiv:1612.08242, 2016.

[20] REN, Shaoqing, et al. *Faster r-cnn: Towards real-time object detection with region proposal networks.* En Advances in neural information processing systems. 2015. p. 91-99.

[21] SASAKI, Yutaka, et al. The truth of the F-measure. Teach Tutor mater, 2007, vol. 1, no 5.

[22] SÁNCHEZ CASTELLANAS, Jaume. *ReMemory: sistema d'avaluació per al tractament de persones amb deteriorament cognitiu lleu.* Departament de Matemática Aplicada i Anàlisi, UB. 2016.

[23] STEHMAN, Stephen V. *Selecting and interpreting measures of thematic classification accuracy.* Remote sensing of Environment, 1997, vol. 62, no 1, p. 77-89.

[24] SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. *Sequence to sequence learning with neural networks.* In Advances in neural information processing systems. 2014. p. 3104-3112.

[25] TING, Kai Ming. *Precision and recall*. In Encyclopedia of machine learning. Springer US, 2011. p. 781-781.

[26] UIJLINGS, Jasper RR, et al. *Selective search for object recognition.* International journal of computer vision, 2013, vol. 104, no 2, p. 154-171.

[27] WORLD HEALTH ORGANIZATION, et al. *Dementia: a public health priority.* World Health Organization, 2012.