Eduardo Daniel Cota de Oliveira Sanches Vicente
Processing and Consolidation of Digitized Customer Information
maryHeaven: An Intelligent Information Extraction System for CRM Reliability
UNIVERSIDADE Đ COIMBRA
1 2 9 0

1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Eduardo Daniel Cota de Oliveira Sanches Vicente

Processing and Consolidation of Digitized Customer Information

maryHeaven: An Intelligent Information Extraction System for CRM Reliability

Internship report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, in collaboration with Altice/MEO, advised by António Dourado Pereira Correia, Professor at the Department of Informatics and Luís Cortesão Engineer and Project Manager at AlticeLabs, presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Faculty of Sciences and Technology

Department of Informatics Engineering

Processing and Consolidation of Digitized Customer Information

maryHeaven: An Intelligent Information Extraction System for CRM Reliability

Eduardo Daniel Cota de Oliveira Sanches Vicente

Internship report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, in collaboration with Altice/MEO, advised by Luís Cortesão Engineer and Project Manager at AlticeLabs and António Dourado Pereira Correia, Professor at the Department of Informatics, presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

1 2 9 0

UNIVERSIDADE Ɖ COIMBRA

This page is intentionally left blank.

Abstract

Customer Relationship Management (CRM) helps a company manage and evaluate its relationships with its past, present, and potential customers. Virtually all processes with the customer are supported in CRM with multiple documents uploaded.

The *maryHeaven* project outlined in this report, in collaboration with Altice/MEO, aims to enrich the CRM database with a new tailored automatic document recognition system. This, to try to solve the problem of missing information in the CRM database in terms of personally identifiable data. This allowed us to extract relevant customer data from hundreds of thousands of scanned documents. In terms of use, for example, the date of birth can be used directly in marketing actions and also as input to improve artificial intelligence models that are in production, in which age is shown to be relevant to the problem at hand. On a monthly level, we expect to have about 3000 documents to be processed, of which we will count those that come in new and those that are already stored internally in the CRM.

In a first experiment made in the CRM environment, the data collected by *maryHeaven* was used to rectify about 25% of the matching entries in the Altice/MEO customer record from the supplied records that had valid Número de Identificação Fiscal (NIF) and Número de Identificação Pessoal (NIP), or simply the valid NIF field. We saw a 10% improvement in Names (filling in missing first or last names) in cases where only the NIF field was correct or verified, as we received fewer matches. In 90% of the cases, information came in to fill in the gaps (the overwhelming majority) or accurate information that was loaded into CRM for valid dates of birth (occasional cases).

The software is internally integrated via an File Transfer Protocol (FTP) server, which feeds the intelligent *maryHeaven* system with all kinds of FileNet files. With this integration it is expected that the software developed will be able to gain a new dimension and that, once implemented internally, it will be subject to future and improved changes, and that this solution can later be used for new purposes in new challenges.

Keywords

Computer vision, Image Processing, Object Detection, Optical Character Recognition,Image Quality

This page is intentionally left blank.

Resumo

CRM ajuda uma empresa a gerir e avaliar as suas relações com os seus clientes passados, presentes e potenciais. Praticamente todos os processos com o cliente são apoiados em *CRM* com múltiplos documentos carregados. O projecto *maryHeaven* delineado neste relatório, em colaboração com a Altice/MEO, visa enriquecer a base de dados *CRM* com um novo sistema de reconhecimento automático de documentos à medida. Isto, de forma a tentar resolver o problema da falta de informação na base de dados *CRM* em termos de dados de identificação pessoal. Este permite-nos extrair dados relevantes de clientes a partir de centenas de milhares de documentos digitalizados. Em termos de utilização, por exemplo, a data de nascimento pode ser utilizada diretamente em ações de marketing e também como input para melhorar os modelos de inteligência artificial que estão em produção, nos quais a idade é demonstrada como sendo relevante para o problema em questão. A um nível mensal, esperamos ter cerca de 3000 documentos a serem processados, dos quais serão contabilizados os que dão entrada nova e os que já estão armazenados internamente na *CRM*.

Numa primeira abordagem em ambiente *CRM*, os dados recolhidos por *maryHeaven*, foram utilizados para rectificar cerca de 25% das entradas correspondentes no registo de clientes Altice/MEO a partir dos registos fornecidos que tinham NIF e NIP válidos, ou simplesmente o campo NIF válido. Vimos uma melhoria de 10% nos Nomes (preenchimento de nomes ou apelidos em falta) nos casos em que apenas o campo NIF estava correto ou verificado, uma vez que recebemos menos correspondências. Em 90% dos casos, a informação chegou para preencher as lacunas (a esmagadora maioria) ou informação precisa que foi carregada em *CRM* para datas válidas de nascimento (casos ocasionais).

O software está integrado de forma interna por intermédio de um servidor FTP, que alimenta o sistema inteligente *maryHeaven* com todo o tipo de ficheiros FileNet. Com esta integração espera-se que o software desenvolvido seja capaz de ganhar nova dimensão e que, uma vez implementado internamente, seja alvo de futuras e melhoradas alterações, e que, esta solução possa, mais tarde, ser usada com novos propósitos em novos desafios.

Palavras-Chave

Visão Computacional, Processamento de Imagem, Detecção de Objectos, Reconhecimento Óptico de Carácteres, Qualidade de Imagem

This page is intentionally left blank.

Dedicated to my grandmother, Maria.

This page is intentionally left blank.

Acknowledgements

I would like to thank all those who had a more active or less present in this project, either internally as part of the CRM team, or the Altice/MEO team, or even the elements that make up the HR team of the organizing company of the GENIUS project, the INOVA-RIA company.

To Professor António Dourado for the positive addition of informative sources about the Machine Learning and Deep Learning areas, as well as his constant active presence in what was the initial exploratory dynamics of solutions to be integrated, for example in phase 0, for the training and evaluation of Object Detection models. To the supervisor and Engineer Luís Cortesão, who every week, in an active way, expressed his interest in the area raised and explored in this process, and also made sure that the processes inherent to the client were done in a practical, simple, and direct way. From the exploration of work dynamics to his active participation in all phases of the project's development, he showed to be an excellent mentor and project manager.

In personal and professional terms, with this collaborative project/initiative between the University of Coimbra and Altice Labs, not only was I allowed to grow and strengthen important technical aspects within the Computational and Deep Learning areas, but also to strengthen soft skills essential to the growth of a good professional on the rise. This academic internship allowed me to evolve my sense of commitment and responsibility, to strengthen the ideal of organization, to gymnastic and consolidate communication in an efficient way, development of a good and structured critical analysis, and above all, allowed me to shape and evolve my current emotional intelligence, of which emerges more self-controlled and self-managed.

Contents

This page is intentionally left blank.

Acronyms

AGGD Asymmetric Generalized Gaussian Distribution. xvii, 10, 11

AHE Adaptive Histogram Equalizatio. 73

API Application Programming Interface. 39, 64, 66, 105

AR Autorização de Residência. 35, 36, 59, 94

AUC Area Under the ROC Curve. 17

BB Bounding Box. xviii, 16, 17, 20, 21, 29, 46, 48–50, 53, 56–59, 67, 83, 86, 98, 101

BI Bilhete de Identidade. 35, 36, 59, 94

BRISQUE Blind/Referenceless Image Spatial Quality Evaluator. 8, 9, 11, 77, 106, 110, 112, 113, 118

CC Cartão de Cidadão. xviii, 35–38, 40, 46, 48, 53, 54, 56–61, 81, 83–88, 92, 94, 95, 98, 100, 101, 103, 104, 107, 111–114, 117, 118

CLAHE Contrast Restricted Adaptive Histogram Equalization. 73, 74

CNN Convolutional Neural Network. xvii, 15, 16, 18, 19, 29, 45, 47

CONV Convolutional Layer. 15

CPU Central Processing Unit. 30, 60–62, 82

CRAFT Character Region Awareness for Text Detection. xvii, 28, 29

CRM Customer Relationship Management. iii, v, xviii, 1–3, 12, 15, 33–38, 40, 46, 55, 84, 91, 107, 110–115, 117–119

CRNN Convolutional Recurrent Neural Network. 28

CSV Comma-separated values. 50, 94, 95, 107, 110, 113

CTC Connectionist Temporal Classification. 28, 30, 83

CUDA Compute Unified Device Architecture. 60, 61, 64, 105

cuDNN CUDA Deep Neural Network library. 61

CV Computer Vision. 3, 6–8, 15, 16, 22, 24, 28, 30, 31, 33, 42, 45, 50, 63, 72, 91, 117, 118

DL Deep Learning. 6, 16, 19, 21, 31, 45, 56, 57, 60, 91, 117–119

FC Fully Connected Layer. 15

FP False Positives. 58

FPS Frames Per Second. xviii, 22, 61

FTP File Transfer Protocol. iii, 115

GCP Google Cloud Platform. xix, 50, 53, 56, 80, 81

GDPR General Data Protection Regulation. 3, 50

GGD Generalized Gaussian Distribution. xvii, 9–11

GPU Graphics Processing Unit. 30, 60–62, 64, 82, 94

GUI Graphical User Interface. 91, 93, 94, 112

HMM Hidden Markov Model. 25, 30

HT Hough Transform. 15

ICDAR International Conference on Document Analysis and Recognition. 29

ID Identity Document. xviii, xix, xxi, 1, 2, 12, 13, 33–37, 39–43, 45–49, 53–55, 58–60, 62, 70, 71, 73–77, 79, 82–86, 91, 92, 94, 97, 98, 106, 107, 109–113, 117–119

IF Identificação Fiscal. 35, 36, 60, 94

IoU Intersection over Union. xvii, 17, 29, 67, 98

IQA Image Quality Assessment. xvii, xxi, 8, 11, 12, 42, 110

IR Information Retrieval. 41

JPEG Joint Photographic Experts Group. xvii, 11, 12, 62, 82

LSTM Long Short-Term Memory. 28

mAP Mean Average Precision. xviii, xix, 17, 18, 21, 61, 67, 68, 98–100, 103, 104

ML Machine Learning. 6, 8, 33, 39, 40, 42, 45, 47, 50

MRZ Machine Readable Zone. 38, 70, 84–86, 92, 95

MS COCO Microsoft Common Objects in COntext. xviii, 20–22, 61, 63, 64

MSCN Mean Subtracted Contrast Normalized. xvii, 9–12

NIF Número de Identificação Fiscal. iii, v, xxi, 37, 39, 84–88, 95, 108, 109, 111, 114, 118, 119

NIP Número de Identificação Pessoal. iii, v, xxi, 37, 39, 85, 87–89, 95, 108, 109, 111, 114, 119

NMS Non-maximum Suppression. 17, 20, 21, 29

NN Nearest Neighbour. 15

NSS Natural Scene Statistics. 8

OCR Optical Character Recognition. xvii, xix, 3, 4, 6, 12, 22–25, 28, 30, 31, 34, 38, 40–42, 70, 71, 75, 76, 79–84, 87, 92, 94, 106, 107, 109, 112, 118, 119

OpenCV Open Source Computer Vision Library. xviii, xix, 50, 61, 72–74, 83, 85, 86

OSD Orientation Script Detection. xix, 75, 76, 81

PASCAL Pattern Analysis Statistical Modeling and Computational Learning. xviii, 57, 64

PDF Portable Document Format. 38

PNG Portable Graphics Format. 62, 82

POOL Pooling Layer. 15

PP Project Profile. 15

R-CNN Region Based Convolutional Neural Network. xvii–xix, xxi, 16, 18–21, 45, 47, 53–56, 64, 92, 93, 97, 101, 103, 106, 130

RGB Red, Green and Blue. 41, 72

RNN Recurrent Neural Network. 29, 30

ROC Receiver Operating Characteristic. xviii, 17, 51

ROI Region of Interest. 19

ROI's Regions of Interest. 16–18, 20

RPN Region Proposal Network. 19, 20, 45, 101

SSD Single-Shot Detector. xvii, xix, 20, 21, 97, 101, 102, 106

SSD Summed Square Difference. 45, 64

SSIM Structural Similarity Index Measure. 8

SVM Support Vector Machine. 19

SVR Support Vector Regression. 11

TIE Textual Information Extraction. 6, 31

VOC Visual Object Challenge. xviii, 57, 64

XML eXtensible Markup Language. xviii, 57, 64

YOLO You Only Look Once. xvii–xix, 21, 22, 45, 57, 60–63, 67, 94, 97–100, 103–107, 111, 117, 118

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Nowadays, many businesses have already invested in a CRM system and are actively using marketing automation applications. It is becoming increasingly relevant to package, review, and enrich contacts from a variety of sources to maintain high quality within the CRM systems, according to Petrovi et al. [80].

Quality of data is paramount because, directly or indirectly, it will be responsible for how well the relationship between company and customer is. The author [80] identifies a list of likely and dangerous problems that could lead to a malfunction in a long-term relationship. Decentralized data, inconsistencies in input and storage, inadequate integration of different data sources, and their tendency in quality deterioration over time are some of the many identified.

Once we reach an equilibrium in the relationship between the CRM and the customer, it means that, for that customer in particular, its data are of high quality, which, consequently means that the information collected across multiple touch-points and channels accurately reflects the behavior and sentiments of the customer [78]. The more correct and accurate the data in the customer files, the better the relationship between the company and them, and the better the success of the CRM project [80].

In the course of this introductory chapter, a brief contextualization of the problem, as well as the exploration and argumentation of the highlighted problems.

1.1 Problem Formulation

In Altice/MEO, the problem in question is related to existing missing information fields on the customer's profiles. Sometimes a customer's data file has its name, ID number, and any other important information, but, even though they have a lot of up to date information, there is some other important information present in the customer's ID card that it is not registered in the CRM.

To avoid wasting resources or time, the proposed solution is to collect all the scanned documents about customers, who are already in CRM and, build an automated solution capable of viewing, detecting, recognizing and extracting the personal data in the detected IDs.

1.2 Business Question

Looking into the set of issues encountered and specified [80], we are interested in identifying defects and misinformation problems. The main subject to be analyzed and resolved in this project, from a CRM perspective, is the business question made by Altice/MEO: **How to collect document identification data from our CRM documents to complement our Customer Database Profile**?

As a company generates certified leads, a CRM tracks the activities of a potential client with his name, birth date, cell phone number, social networks, etc. Before aiding leads with a planned journey that benefits both parties, the system gathers as much information as possible about the target clients.

A record made between a customer and a department within MEO may result in a documentary record of the operation performed. For example, when a contract at MEO's sales department, or, the registration of a new service by a customer at an MEO store happens, a document that records the operation is created. Usually consists of a duplicate form and, at the end of the document, sometimes a scan of the identification document of the customer. In order not to lose the customer data information in the CRM database, these documented operations will be named with the ID in the database that relates the processes to the target customer, the FileNet.

In the CRM environment, these documents will be named with the 8 digits that make up the FileNetID they correspond to internally. The pages with the scanned personal documents are our extraction source. From these, containing IDs, we expect to collect important information such as name, date of birth, etc.

With this relevant information, we can later perform matching scripts on the CRM tables to obtain a data collection that could provide valuable information about a set or group of potential customers. These sorts of data may be used to do a potential and strategic statistical analysis, which is timely for a company to understand how they may enhance their customer experience as their marketing and sales success.

1.3 Business Value and Goals

Internally, there have always been about 302.000 scanned documents stored that have not yet been processed. Every month about 3.000 new documents are inserted into the database. Something that, in practical terms, can take years to process. The value of this system is in automating the manual insertion of the different personal fields present in the different types of identification documents, thus populating the missing information fields in the CRM and correcting outdated data.

The correct identification of the client is essential for its univocal representation in the company's client registry, representing the first dimension of its characterization. It is also relevant for tax or legal issues arising from the commercial and contractual relationship with customers. Obtaining this data, the identity base of MEO customers, is essential for a personalized commercial action based on the analysis of characteristics, behavior and needs, and using identity for communication and contact.

In terms of goals, the first objective is to identify among the various types of documents those that are personal identification documents. For each type of document identified, it is important to maximize data collection, measuring its quality in terms of performance and feasibility in a scalable implementation.

From a quantitative perspective, theoretically, it is expected that the software to be developed will be able to collect personal information to fill and correct in about 15% of the data already entered previously in the CRM tables and achieve an addition of new data in the biographical tables of about 30%.

These objectives should take into account an implementation of automatic learning and discovery mechanisms of types of identification documents that will allow extracting personal identification data to complement the customer's file. In addition to this, it is worth mentioning the importance of identifying and implementing image processing and OCR suitable for each type of document, always articulating the information processing with ontologies appropriate to the diversity of information collected.

The processes implemented should take into account the personal information processing constraints resulting from the General Data Protection Regulation (GDPR).

1.4 Report Outline

The following internship report, presented and contextualized in the previous subsections, is going to be divided in several chapters:

- In Chapter 2, a survey of all the theoretical components necessary to consolidate the goals outlined in the previous subsection will be carried out. Yet within this, the different technologies needed within the pipeline will be raised, as well the work related to the application of these techniques and technologies for the same purpose as this project. By comparing and passing through the several knowledge areas in all the chapter subsections, real applications, and theoretical foundations of the technologies used in Computer Vision (CV) solutions, we will get an average overview of the project scope, which will turn the reading much more interactively and easy to follow up.

- In Chapter 3, in addition to the proposed resolution of this project, all the activities and tasks that must be considered for the realization of the presented proposal are presented and schematized. Within the proposal, the phases inherent to the intelligent and automated pipeline built specifically for the need at hand are subdivided and analyzed individually, so that there is a logical thread throughout its construction.

- The techniques utilized to train object detection models to help in phase 0 of the proposed pipeline will be covered in the chapter 4. We begin by discussing the exploratory testing conducted throughout the project's early months. The approach for training the various types of methods, such as one-stage and two-stage methods, as well as the strategies for assessing the models generated, are then provided.

- The methods presented in chapter 5 are important for the pipeline, in terms of information quality. Here, the experiments in the area of image processing were all focused on the preprocessing component of the pipeline to be constructed, that is, the operations and algorithms employed in all of the experiments that will be discussed were designed to aid the preprocessing in the OCR task. Therefore, this chapter aims to get to know the types of Image Quality problems that we have at hand, as well as the technologies and strategies that can be applied to surpass those problems.

- It is important to comprehend what techniques may be utilized to enhance the confidence extraction of the OCR motors under investigation, as well as how these will be included into the pipeline. In light of this, a brief discussion of the methods utilized in this phase will be presented in Chapter 6, as well as some of the tactics used to enhance and format the data collected by the OCR motor in the post-processing step.
- In chapter 7 a chronological path has been followed behind the scenes of the final software installation, maryHeaven, as well as the strategy behind its maintenance and integration into the Altice/MEO processes.
- Chapter 8 is where the results of the many phases that make up the final system will be analyzed and debated. From then, concepts will be assessed one by one in order to put the final software's robustness and integrity to the test.
- Finally, in the last chapter, chapter 9, t is carried out an introspective analysis of the path taken throughout the project. Not only in terms of successes, but also in terms of challenges and points of risk that the program faced during its development throughout the project's months. Aside from that, a little critical reflection on the work methodologies used, as well as the competencies and soft skills that were improved or learned throughout the project investigation conducted in collaboration with Altice/MEO, is carried out.

This page is intentionally left blank.

Chapter 2

Background

The task of classifying a certain digitized document and extract its information has a huge and deeply complex background of study, not only in the field of Optical Character Recognition (OCR) and extraction but in other important areas and technologies that are very important for the simplicity of performing OCR extractions in a Big Data collection of documents. Dealing with the complex task of teaching and commanding a machine to look at and analyze a matrix of pixel data in a digital document and expect to get great and accurate results without all the background support from image enhancement techniques and image preprocessing algorithms, it is hard-boiled.

Textual Information Extraction (TIE) is related to extracting text from images, whether it is informational or descriptive. TIE systems typically receive as input images or videos, which may be with different numbers of color channels, compressed or uncompressed, and with moving or static text [37].

There are immense applications of TIE systems, from the simple task of extracting the digits of a license plate to supporting the medical industry, automobiles, tourism tools, and many other applications [37].

The text to be extracted can be present in a different possible number of scenarios, like in historical books, newspapers, handwritten or digitized documents, television programs, road signs, or websites. In this project, we are interested in focusing on textual extraction in the scenario of digitized documents, i.e., document text.

When it comes to this type of documents, many challenges in terms of image processing arise. When we talk about digitized documents we have to take into consideration the environment in which it was involved at the time of its digitization, and the positive or negative consequences of that operation. So the ultimate challenge for any TIE system is when variations occur in images such as resolutions, complex or noisy background, text illumination, reflections, or other types of distortions that might happen in the digitization process of the target document [22, 37]. These, and many other variations, make it hard to get a good and reliable automated TIE system.

Image Processing is the "life-saving boat" of all the classifiers, including the TIE systems, that are, in practical terms, Computer Vision (CV) applications.

Encountered our baseline approach for the resolution of the previously introduced problem, we are going to explore in the next sections the background work and approaches in the area of CV, the inner subareas of research and the evolution in the past years on Deep Learning (DL) and Machine Learning (ML) areas, applied to CV systems.

2.1 Computer Vision

As the name implies, CV is a research discipline concerned with the progressive learning of computers and systems by one of nature's greatest gifts, vision. It aims to turn computer system software and hardware into automated and intelligent solutions that can perform as well as, if not better than, the human visual system [23, 26, 82, 101]. This exciting field is concerned with the theoretical basis underlying the information retrieved from digital material, pictures frames, or other forms of digital data.

It is important to point out that we, human beings, have problems and imperfections. One example of those imperfections in the visual illusions is that sometimes, depending on the scenario and the background complexity of what we are watching, we can be conducted to mistakes and fake or incorrectly contextual inferences [23].

Why are we focusing on the problems and challenges that particular visual situations of the human visual system present? Because, like people, robots and computers are designed to fail, even if they haven't yet reached their final state; no system in the universe is perfect. Furthermore, the vision inference task is extremely difficult and complex [23, 26, 82, 101], and it becomes even more difficult and complex when we add scenarios with highly complex visual disturbance and noisy elements, which can give the wrong idea of existing elements or other descriptive critical points/vectors, distorting the final classification.

The difficulty of vision is explained in part by the fact that vision is an inverse problem, in which we search for more extra information (sometimes redundant) with the sole purpose of attempting to solve some of the given unknown or insufficient information that was left to answer in the middle of the brain work, according to [101] (image processing). We say inverse because we are attempting to recreate the characteristics of what we're seeing outside in one or more complicated pictures (shape, light, color distribution, depth, etc.).

Both CV and machine vision have consolidated their research to the point that they are now referred to as Machine Vision [13, 99]. Machine vision is a mix of CV techniques and processes with other methods supporting industrial applications, and an intelligent automated system, while CV works in the technological field of automated applications in image analysis. In this study field, industrial applications are pre-programmed to tackle single and specific operations [13, 99], however learning-based techniques are gaining traction at a rapid rate.

Even though we are presented with a very complex scenario with a lot of different graphical elements, we have proof that vision exists and is possible because of our existence and because of our great visual system that can retrieve a lot of relevant information from the light distribution [82]. As a result, we may conclude that while CV is not impossible, it is one of the most difficult novelties of the twentieth and twenty-first century.

2.2 Image Processing

Once we apply a set of operations and functions in the input data, we are ensuring that a wide range of problems are reduced and avoided, such as noise or image distortion [9, 19, 35, 101]. The use and development of image processing operations are affected and, consequently shaped by the development of computers, the development and creation of discrete mathematics theory, and the need for a wider range of applications in the main industrial fields (agriculture, medicine, military, and overall industrial companies) [19, 23, 35].

More directly, the goal of image processing is to convert the digital input, in our case, into a more suitable form, for further better analysis [101]. Some specialists consider that this stage is outside the CV application [101], but most CV applications, such as recognition and detection require this stage in order to achieve good and accurate results [23, 26, 101].

Input Quality is one of the most important factors in a CV application. With no cleaning stage in a process that uses quality output as the first metric of evaluation, the pipeline is not completed or ready. The question arises if we are talking about a pipeline that is tuned to perform image classification, which is such a delicate task. Taking image classification as an example, if we do not perform image preprocessing, which has the main goal of input standardization, the complexity of the system, and consequently, the accuracy of the classifier will be very low.

Image processing involves simple tasks like resizing, geometric and color transformation, the color converting [26], and many more that are going to be seen in the next subsections of this section. In the first chapter of this dissertation, we have seen that the input that we are expecting has no fixed layout or template, so every image of the input dataset can have multiple features that are basically with different values.

In the next subsections we will cover which operations and functions we have at our hands to perform a good and reliable preprocessing stage, not only by trying to standardize the input images that will be send to the final model, but also to increase the ML solution accuracy, but, at the same time, by applying data augmentation techniques for overfitting reduction [26].

2.2.1 Image Quality Assessment

Quality of images is a concept that relies heavily on observers. In general, it is associated with the conditions in which it is seen. It is thus a very subjective subject. The objective of the image quality evaluation is to express the human sense of quality quantitatively. These measurements are usually used to assess algorithm performance in many areas of CV, such as image compression, image transfer and image processing [64].

The Image Quality Assessment (IQA) is organized into two primary research sectors: (1) reference-based assessment and (2) non-reference assessment. IQA is based largely on research. The key distinction is that reference-based techniques depend on a high-quality picture as a source for assessing the image difference. The Structural Similarity Index Measure (SSIM) [110] is an example of reference-based assessments. Non-reference image quality evaluation does not require a reference picture to evaluate image quality. Instead, the algorithm is given simply a distorted image whose quality is being evaluated [68].

The majority of blind techniques have two phases. The first phase calculates features that characterize the structure of the picture, while the second step looks for patterns in the features that relate to human opinion. TID2008 is a well-known database based on a mechanism for calculating human opinion ratings from related pictures [81]. It's a popular way to compare how well IQA algorithms work.

Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [68] is a model that calculates features only from picture pixels (other methods are based on image transformation to other spaces like wavelet). It is shown to be very efficient since it calculates its characteristics without the need of any transformations.

The spatial Natural Scene Statistics (NSS) model of locally normalized luminance coefficients in the spatial domain, as well as the model for pairwise products of these coefficients, are used in BRISQUE [68].

Given an image, we need to compute the locally normalized luminescence via local mean subtraction and divide it by the local deviation (equation 2.1). A constant C is added to avoid zero divisions.

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C} \tag{2.1}$$

Taking the previous equation (equation 2.1), if the domain of $I(i,j)$ is [0, 255] then $C = 1$ if the domain is [0, 1] then $C = \frac{1}{255}$.

We must first compute the local mean (equation 2.2) before we can calculate the locally normalized luminescence (equation 2.1), commonly known as Mean Subtracted Contrast Normalized (MSCN) coefficients. A Gaussian kernel of size w is used here (K, L).

$$\mu(i,j) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} I_{k,l}(i,j) \tag{2.2}$$

The local mean may be complex, but it is simply computed by applying a Gaussian filter to the image. Before the calculation of the MSCN coefficients in equation 2.1, we must calculate the local deviation, as described in equation 2.3.

$$\sigma(i,j) = \sqrt{\sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} \left(I_{k,l}(i,j) - \mu(i,j)\right)} \tag{2.3}$$

The MSCN coefficients are distributed as a Generalized Gaussian Distribution (GGD) for a broader spectrum of the distorted image. The GGD density function is given by the following equation (2.4):

$$f(x; \alpha, \sigma) = \frac{\alpha}{2\beta\Gamma(1/\alpha)} e^{-\left(\frac{|x|}{\beta}\right)^{\alpha}} \tag{2.4}$$

where,

$$\beta = \sigma \sqrt{\frac{\Gamma\left(\frac{1}{\alpha}\right)}{\Gamma\left(\frac{3}{\alpha}\right)}}$$

and Γ is the gamma function. The parameter α controls the shape and σ^2 is the variance.

The signs of neighboring coefficients have a regular structure as well, which is disrupted when distortion occurs. The model of pairwise products of nearby MSCN coefficients in four directions is proposed in the following equations (2.5): (1) horizontal H, (2) vertical V, (3) main-diagonal D1 and (4) secondary-diagonal D2.

$$\begin{aligned}
&(1) H(i,j) = \hat{I}(i,j)\hat{I}(i,j+1) \\
&(2) V(i,j) = \hat{I}(i,j)\hat{I}(i+1,j) \\
&(3) D1(i,j) = \hat{I}(i,j)\hat{I}(i+1,j+1) \\
&(4) D2(i,j) = \hat{I}(i,j)\hat{I}(i+1,j-1)
\end{aligned} \tag{2.5}$$

Lasmar et al. [57] points out that the GGD does not suit the empirical histograms of coefficient products well. They suggest fitting an Asymmetric Generalized Gaussian Distribution (AGGD) model [57] instead of fitting these coefficients to GGD. The density function for AGGD is given by:

$$f\left(x;\nu,\sigma_l,\sigma_r\right)=\begin{cases}\frac{\nu}{(\beta_l+\beta_r)\Gamma\left(\frac{1}{\nu}\right)}e^{\left(-\left(\frac{-x}{\beta_l}\right)^{\nu}\right)} & x<0\\ \frac{\nu}{(\beta_l+\beta_r)\Gamma\left(\frac{1}{\nu}\right)}e^{\left(-\left(\frac{x}{\beta_r}\right)^{\nu}\right)} & x>=0\end{cases} \tag{2.6}$$

where, $\beta_{\text{side}}=\sigma_{\text{side}}\sqrt{\frac{\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)}}$ and the side can be either r or l. Another parameter that is not reflected in the previous formula (2.6) is the mean, which is given by $\eta=\left(\beta_r-\beta_l\right)\frac{\Gamma\left(\frac{2}{\nu}\right)}{\Gamma\left(\frac{1}{\nu}\right)}$.

An AGGD described in [57] starts, as described in equation 2.7, by calculating γ where N_l is the number of negative samples and N_r is the number of positive samples. Also, at the same time, calculates the $\hat{r}$ estimation described in the 2.8, which later will be used aside with the γ to calculate the $\hat{R}$ (equation 2.9).

Finally, the α estimation (equation 2.10) is made using the approximation of the inverse generalized Gaussian ratio. In equation 2.11, the left and right scale parameters estimation is also made.

$$\hat{\gamma}=\frac{\sqrt{\frac{1}{N_l-1}\sum_{k=1,x_k<0}^{N_l}x_k^2}}{\sqrt{\frac{1}{N_r-1}\sum_{k=1,x_k>=0}^{N_r}x_k^2}} \tag{2.7}$$

$$\hat{r}=\frac{\left(\frac{\sum|x_k|}{N_l+N_r}\right)^2}{\frac{\sum x_k^2}{N_l+N_r}} \tag{2.8}$$

$$\hat{R}=\hat{r}\frac{\left(\hat{\gamma}^3+1\right)\left(\hat{\gamma}+1\right)}{\left(\hat{\gamma}^2+1\right)^2} \tag{2.9}$$

$$\begin{aligned}\hat{\alpha}&=\hat{\rho}^{-1}(\hat{R})\\ \rho(\alpha)&=\frac{\Gamma(2/\alpha)^2}{\Gamma(1/\alpha)\Gamma(3/\alpha)}\end{aligned} \tag{2.10}$$

$$\begin{aligned}\sigma_l&=\sqrt{\frac{1}{N_l-1}\sum_{k=1,x_k<0}^{N_l}x_k^2}\\ \sigma_r&=\sqrt{\frac{1}{N_r-1}\sum_{k=1,x_k>=0}^{N_r}x_k^2}\end{aligned} \tag{2.11}$$

Fitting the MSCN coefficients and shifted products to the GGD yields the features required to calculate picture quality. The MSCN coefficients must first be fitted to the GGD, followed by the pairwise products to the AGGD (figure 2.1) [68]. In table 2.1 we can see the list of the features generated to support the quality assessment.

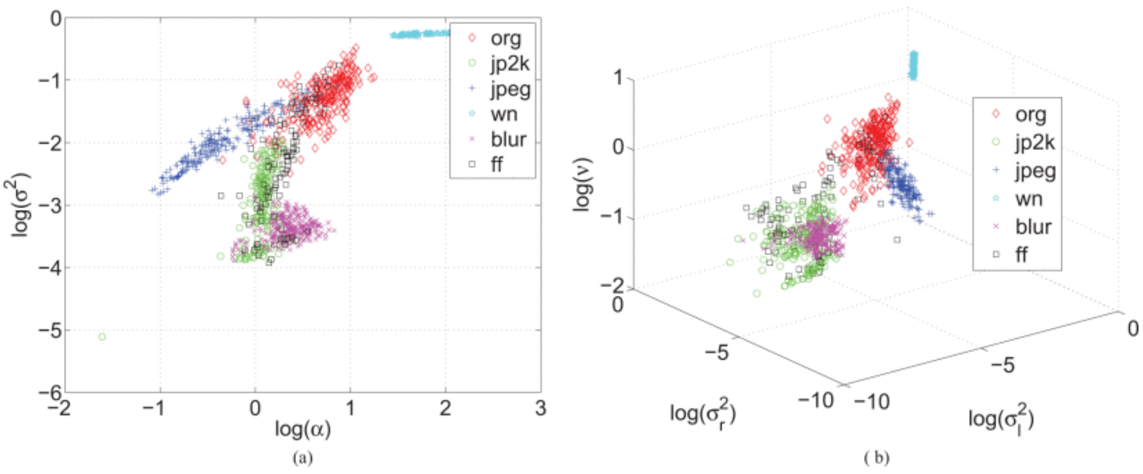


Figure 2.1: (a) Two-dimensional plot of shape and scale parameters obtained by fitting GGD to the empirical distributions of MSCN coefficients of pristine images from the Berkeley image segmentation database [66] and simulated distorted images (Joint Photographic Experts Group (JPEG) 2000, JPEG, White Noise, Gaussian Blur, and Fast fading channel errors) from the LIVE IQA database [91]. (b) A 3D scatter plot of form, left scale, and right scale was created by fitting AGGD to horizontal paired products using the same collection of pictures as before (a) [68].

Feature ID	Feature Description	Computation Procedure
$f_1 - f_2$	Shape and variance	Fit GGD [90] to MSCN coefficients
$f_3 - f_6$	Shape, mean, left variance, right variance	Fit AGGD [57] to H pairwise products
$f_7 - f_{10}$	Shape, mean, left variance, right variance	Fit AGGD [57] to V pairwise products
$f_{11} - f_{14}$	Shape, mean, left variance, right variance	Fit AGGD [57] to $D1$ pairwise products
$f_{15} - f_{18}$	Shape, mean, left variance, right variance	Fit AGGD [57] to $D2$ pairwise products

Table 2.1: Summary of features extracted in order to Classify and Quantify distortions [68].

After calculating the MSCN coefficients and the pairwise products, we can verify that the distributions are in fact different (figure 2.2) [57, 68].

The quality evaluation is calculated using a pre-trained Support Vector Regression (SVR) model. However, we must scale the features to [-1, 1] in order to get excellent results. The visual quality is measured on a scale of 0 to 100. A picture quality of 100 indicates that the image is of poor quality. This method was tested with the TID2008 [81] database and performs well, even compared with referenced IQA methods.

BRISQUE [68] outperforms the full-reference peak signal-to-noise ratio and the structural similarity index statistically, and it outperforms all current distortion-generic Non-reference IQA methods. BRISQUEs computational complexity is minimal, making it ideal for real-time applications.

BRISQUE characteristics can also be utilized to identify distortions. On a future work perspective, a nonblind image denoising algorithm may be supplemented with BRISQUE to perform blind picture denoising as an example of a novel practical use. BRISQUE augmentation improves performance above state-of-the-art techniques, according to the findings [68].

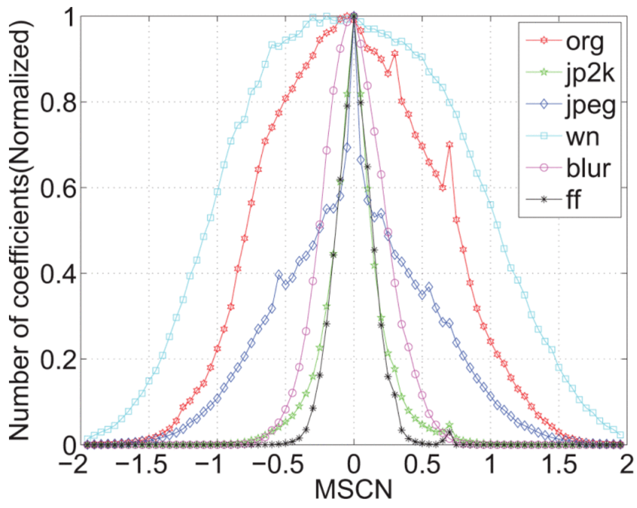


Figure 2.2: MSCN coefficients histogram for an undistorted natural picture and its different warped variants:"org" being the original image, “wn” is related with the additive white Gaussian noise from the LIVE IQA database [91], “jp2k” related with the JPEG2000, “jpeg” is an JPEG compression, ”blur” refers to the Gaussian blur, which is a type of blur, and, for last, Rayleigh fast-fading channel simulation, abbreviated as “ff” [68].

2.2.2 Image Enhancement Techniques

The process of modifying digital images so that the effects are more acceptable for display or further image analysis is image enhancement (figure 2.3). We may, for example, remove noise, sharpen, or brighten an image, making it easier for key features to be identified [9, 23, 63].

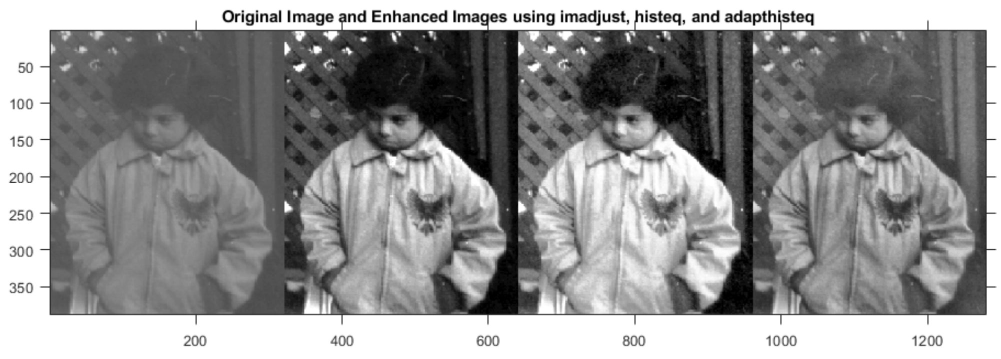

Original Image and Enhanced Images using imadjust, histeq, and adapthisteq

Figure 2.3: Image Enhancement Processing in an old Photography [2]

Since we are expecting noisy images with, sometimes, a complex/confusing background, we need several and different enhancement techniques to get through these types of issues. Not only, in the OCR phase of our pipeline system, but also the Object Detection initial phase. If we perform various preprocessing steps through all the pipeline, without getting to lose important information, we are contributing to the increase of reliability in our Customer Relationship Management (CRM) solution. Furthermore, to get higher Identity Document (ID) recognition rates, we must have good and adaptive preprocessing pipelines.

In further subsections, we are going to take a look at the different techniques that are useful to improve the quality of input images used on our dataset and in the input data.

Thresholding

Since we are dealing with IDs, there are several parts (text and graphical) that are useful and others that are not that much. Being able to divide out regions of interest into objects which we are, of course, interested in, from the ones that represent, for example, regions of background that does not bring anything new to the recognition and detection phase it is possible by using an enhancement technique called thresholding (figure 2.4)[9, 29]. It works as a segmentation tool, but it works at the pixel level. Thresholding operations usually are made on grayscale or color images, where the output is a binary image representing the segmentation [29].

Global thresholding methods operate with one pre-chosen threshold value for the entire document image, which is often estimated by looking at the intensity histogram of the target image [9]. In practical terms when applying global thresholding on a grey-level image the output is a binary image, where this image is composed by only two types of pixels. The ones that represent the foreground area (text/graphical components) and the ones that represent the background area of the document image [9].

There are several types of global thresholding techniques and they are constantly appearing because the thresholding operation is only determined by the way that we choose the thresholding value (T).

$$g(x,y) = \begin{cases} 1, & \text{if } (x,y) > \mathrm{T} \\ 0, & \text{if } (x,y) \le \mathrm{T} \end{cases} \tag{2.12}$$

Being g(x,y) the thresholded image.

There are many ways to select a global threshold (T) [76]. From all the possible methods to find and set the value of T, the Otsu method [9, 76], that seeks for the best value of threshold capable of minimizing the intra-class variance is described in equation 2.13.

$$\sigma_\omega^2 = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \tag{2.13}$$

where ω_1 and ω_2 are the weights of the two classes separated by T and σ_i^2 is the variance of these classes.

He also proved that minimizing the intra-class variance is the same as maximizing inter-class variance, as is shown in equation 2.14 [9]:

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \tag{2.14}$$

In terms of algorithm flow of the previous Otsu method (equation 2.14), it firstly computes the histogram and the probabilities associated with each intensity level and initializes the weights $\omega_\mathrm{i}(0)$ and $\mu_\mathrm{i}(0)$. Then, while the values of T don't reach the maximum intensity, it updates the values of $\omega_\mathrm{i}(0)$ and $\mu_\mathrm{i}(0)$, while compiling and computing the maximum value of ${\sigma_\mathrm{b}}^2(\mathrm{T})$, which, at the end corresponds to the desired threshold [9].

2.2.3 Noise Removal

Digital data is vulnerable to different kinds of noise. Noise is the result of errors in the process of capturing images that result in pixel values that do not represent the real

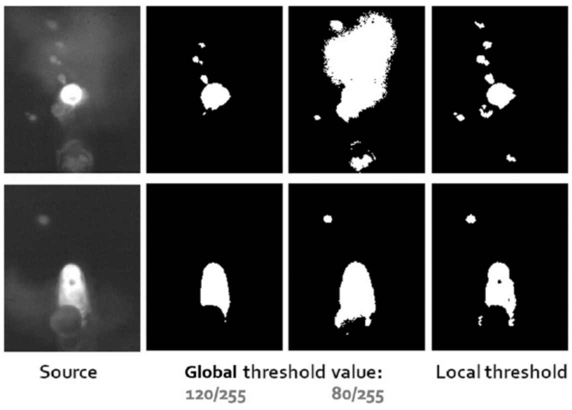


Figure 2.4: Practical Application of Different values of Threshold in comparison with Local Thresholding results in the same target Image [73]

strength of the scene [29]. In the perspective of this project, i.e, in image classification systems, the reduction of noise, in the preprocessing stage, can add substantial reliability and robustness to the feature extraction and recognition stages [9, 23, 29, 101].

Analyzing the inputs that we have in our datasets, the most often type of noise that we can find a lot in binary images is speckle noise, which is a salt-and-pepper noise effect [9, 29].

One way and the most accurate and efficient way to solve this type of noise problem is by operating a filling process, where each isolated pixel that is labeled as "salt-and-pepper" is filled by the remaining surrounding pixel values (figure 2.5).

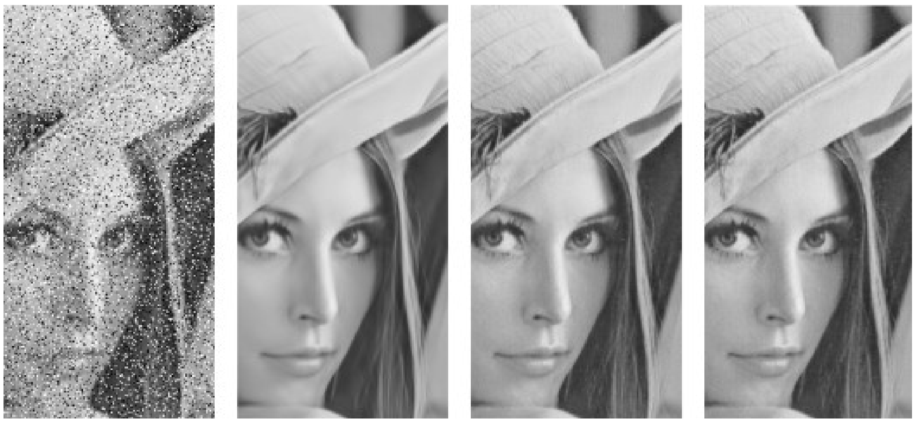


Figure 2.5: Salt-and-pepper noise removal based on shearlet transform and noise detection [100]

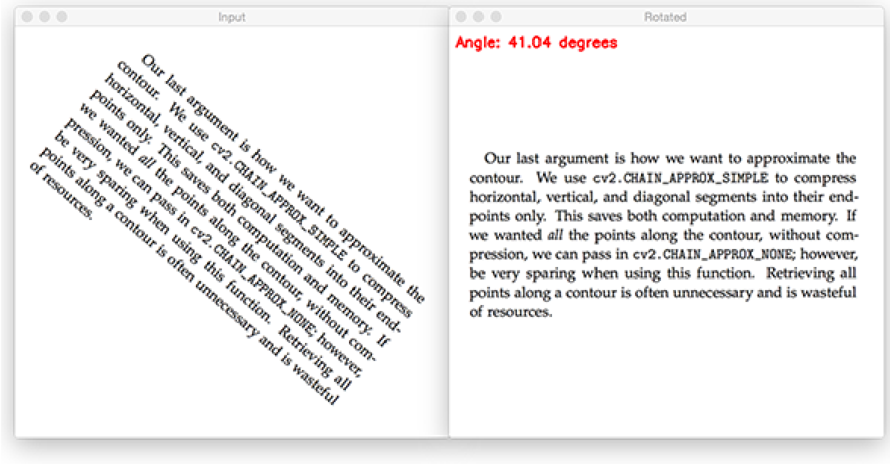
2.2.4 Skew Detection and Correction

Sometimes our input images have a wrong orientation, i.e, seeing a document (A4 pages) in a normal way, we see that the words and the graphical components take a general orientation angle (page orientation), which can be portrait orientation (vertically oriented) or landscape orientation (horizontally oriented).

Most paper documents use portrait orientation [23], where the visual components are represented from the top to the bottom of the document, which consequently means that the characters that we are looking for, are at the same orientation [23]. Although, often we can expect some different orientation angles, due to many possible logistic problems, such as, in the moment of the scanning the document is wrongly oriented inside the scanner, or due to CRM data persistence problems or even due to possible deficient posterior compression of the document file.

In CV terms, the name of the method which allows us to correct the orientation angle of the document is documented skew correction (figure 2.6) [9]. There are many ways to detect the skew of an image. By analyzing the projection profile, by doing Hough Transform (HT) analysis, clustering or connect components or correlation between lines technique [9]. Since 1998 [44] that we are struggling in having a good and reliable technique. By then, the accuracy in degrees of most skew techniques was higher than 0.1º [9].

Nowadays, most of the famous used skew techniques are based in Project Profile (PP), HT and Nearest Neighbour (NN) [7].



Angle: 41.04 degrees

Our last argument is how we want to approximate the contour. We use `cv2.CHAIN_APPROX_SIMPLE` to compress horizontal, vertical, and diagonal segments into their end-points only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in `cv2.CHAIN_APPROX_NONE`; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

Figure 2.6: De-skew Process of 41.04 degrees [5]

2.3 Object Detection

Convolutional Neural Networks (CNNs) are an application of a deep neural network applied to the context of the image classification problem. Its architecture is created based on the inherent needs of image classification, so the internal layers of the network are called convolution layers instead of the fully connected layers. However, the fully connected layers still play a very important role in the classification part of CNNs [8, 23, 26, 58, 115].

Internally, the input to the network is the target image in its pure state.

From layer to layer inside the hidden layers, the features that were saved in the features map are learned gradually, that is, in the most initial layer, the network learns the most basic features, such as lines and edges, then passing to the next ones it learns to detect and recognize corners, shapes and circles, until in the deeper layers the network learns more complex features and shapes, such as objects in their global aspect [26].

More specifically, a CNN normally consists of a Convolutional Layer (CONV), a Pooling Layer (POOL), and a Fully Connected Layer (FC) [8, 23, 26].

As we can see in figure 2.7, the CNN architecture is composed of the input layer, followed by two consecutive pairs of a convolution layer followed by a POOL (forming the feature learning stage of the CNN) and the FC followed by the application of a softmax [8, 26, 58].

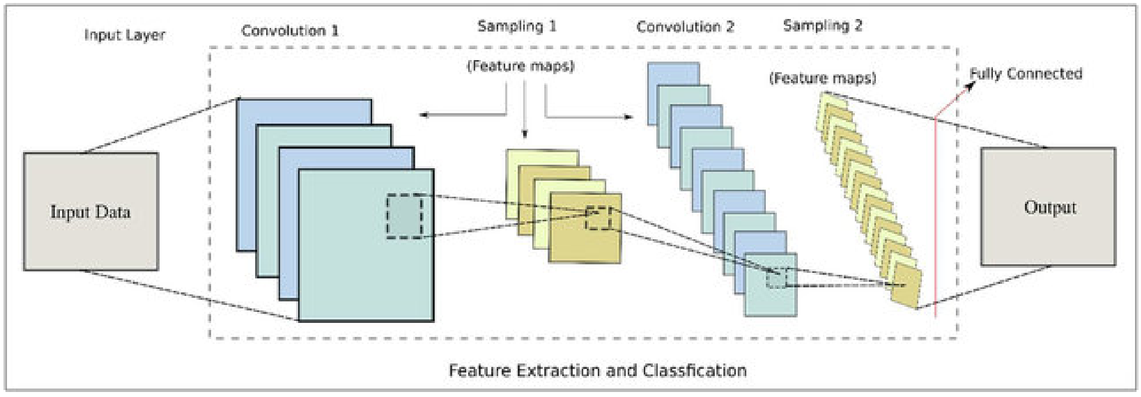


Figure 2.7: General architecture for a CNN [48].

In addition to the classic layers commonly introduced in a CNN network, it is also possible to add layers like the dropout layer that helps with the overfitting problem. The dropout layer causes a certain percentage of nodes in a certain network layer to be deactivated, thus reducing the super learning effect that can produce tragic results in the image classification process [8, 26].

The architecture of CNN can join several tasks in a single technological flow, in this case, the possibility to join the extraction of features with the classification in a single medium, and also, as in a CNN application, the fast Region Based Convolutional Neural Network (R-CNN) [34], combine classification with bounding box regression in multifaceted learning. Benefiting from the wide learning power of CNNs, many of the CV problems are rescaled into high dimension problems and solved from this point of view.

CNNs are a practical application usage of deep neural networks applied to the image area, i.e., they are architected according to the needs and specificity of the features extracted from the images. With the help of their power of self-learning imposed by the layers of seizure and the freedom of choice and adjustment of hyperparameter values [26], it is possible to formulate new specialized networks that have their origin in the architecture of the CNNs.

The deep neural networks (e.g., CNNs) are used for the image classification task [26]. When we classify an image and assign it a class, we are only considering one object in its entirety, the image itself. However, on many occasions we are asked to classify several objects in the same input image, so we have to consider that we may have the possibility of having in any part or region of the image one or more objects of the same or different classes. In CV, this task is called object detection [23, 26, 101, 115].

When we apply object detection to a given image we are not only interested in classifying the multiple objects in the image taking into account the features extracted from them and their properties, but we are also interested in knowing where it is in the global image. This last task is performed by drawing a Bounding Box (BB) around the previously classified and recognized object [26]. In other words, right from the start, we are imposing the creation of a technology that will have a level of complexity much higher than what we have seen in the deep neuronal networks applied only to the classification of unique images.

The generic frameworks oriented to object detection have a structure that, although it varies in some fields and hyperparameters, doesn't leave much of the norm in a generalized way in all the existing frameworks [26, 115]. Typically a generic framework for object detection has a phase where the most favorable regions for the existence of an object in the image (region proposal) are proposed. The DL model used initially generates Regions of Interest (ROI's) that will then be processed in the course of the process. These regions, where the model believes there are objects, are characterized by a set of coordinates in the image (BBs) and these have an associated objectness score [26]. The higher the value of this score the more likely it is that the target BBs represent the specific location of an object in the image.

Normally, a detector is composed by a backbone which is pre-trained on ImageNet and a head which is used to predict classes and BBs of objects.

In the extraction and detection of visual features, these are analyzed taking into account the proposed ROI's. The features are analyzed and evaluated independently of each other and extracted based on the region under study [26, 115]. Besides, it is predictable that at some point we may have the case where BBs are generated that are over other BBs already found, that is, the case where an object is classified and found multiple times. For this not to happen it is necessary to apply Non-maximum Suppression (NMS). The NMS technique ensures that an object is only detected and located once, thus suppressing other BBs that may be related to the same object with a lower objectness score, thus ensuring that only the BB with the higher objectness score is maintained and used for the final classification of the object [26].

The process behind the NMS technique takes into account a fixed confidence threshold value and a box with a rating probability below the stipulated threshold value is discarded [26, 115]. If there are still boxes over others, the overlap is calculated. Here the remaining boxes are summed up and the calculation of the Intersection over Union (IoU) value is carried out (equation 2.15). Any box with the IoU value below the NMS threshold (usually 0.5) is deleted (figure 2.8).

$$\text{IoU} = \frac{B_{\text{ground truth}} \cap B_{\text{predicted}}}{B_{\text{ground truth}} \cup B_{\text{predicted}}} \tag{2.15}$$

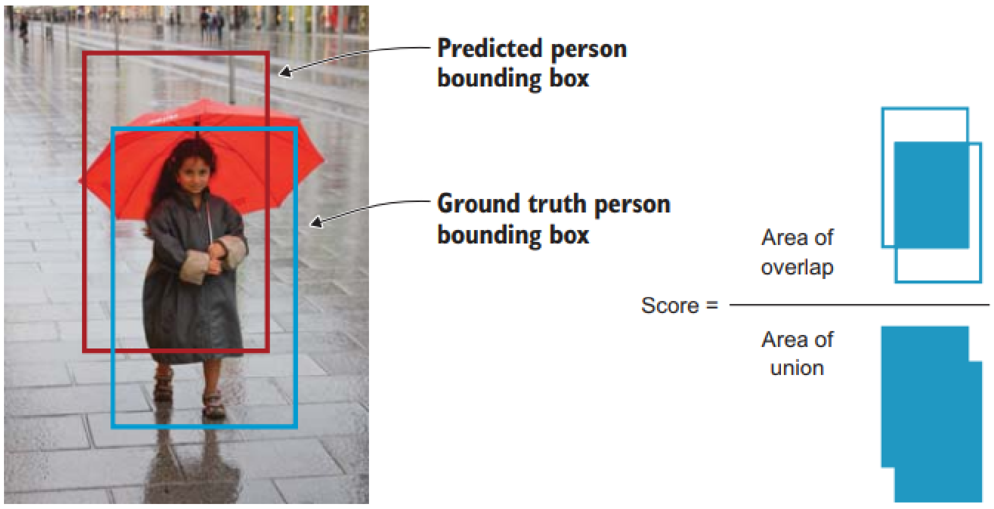

Predicted person bounding box
Ground truth person bounding box
Area of overlap
Score =
Area of union

Figure 2.8: IoU process explained [26].

In addition to this evaluation metric, a metric also very common in these frameworks is the Mean Average Precision (mAP), which is used to calculate the network accuracy. The mAP value is normalized in the range 0 to 100, and the closer the values are to 100 the better [26]. The mAP of an object is the result of the average Area Under the ROC Curve (AUC) of the Receiver Operating Characteristic (ROC) curve [26, 115]. The calculation behind the mAP value is more complex than the calculation, for example, of the recall value.

Within the multiple applications formulated over the years in the object detection field, we can say that there are two major groups of object detection frameworks: The Region Proposal-Based Frameworks and the Regression-Based Frameworks [26, 115].

The first framework mentioned generates proposed regions first and only then applies the classification to each of the proposed regions into different types of objects [115], it is also called two-stage method. The second one, also called one-stage method, addresses the problem of object detection as a regression or classification problem, thus adopting a unified framework structure, returning directly at the end the classification of the various objects detected in the image and their corresponding classification [26, 115].

2.3.1 Region Proposal-Based Frameworks

The proposed region-based frameworks impose two procedural steps as seen above. They are the frameworks that are closest to the human visual system when it comes to detection and recognition of objects in the real world [26]. Of the various algorithms created over the years, the networks that stood out the most are those belonging to the R-CNN family [33]. Since the creation of the R-CNN network in 2014 by Ross Girshick [33], the R-CNN family has been expanding, giving space to the emergence of Fast R-CNN [34] and Faster R-CNN [87]. At the time, the first network of the R-CNN family (R-CNN) was created, its mAP value would be around 53.3% [33] in the PASCAL VOC-2012 benchmark dataset, which meant a 30% increase over the best solution at the time, thus marking forever in the history of CNNs a major step towards optimization in the task of object detection [26, 33, 115].

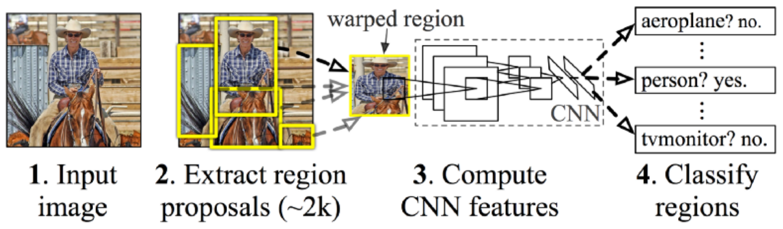


Figure 2.9: R-CNN global architecture [33].

The R-CNN model (figure 2.9) consists of four different phases. In the first phase, it is like what was seen in the generalized structure of these networks, an extraction phase of ROI's [26, 33]. This search and extraction are done with the help of the selective search algorithm that scans the input image to find regions that contain blobs, and with this proposes the ROI's, which are then passed to the next modules of the network pipeline so that these are processed and treated [107]. Never forgetting that these regions are normalized taking into account the fixed dimension of the input images since CNNs only work with a fixed dimensional value of the images (input layer dimensions) [26].

Despite the promise left by R-CNN and the novelty brought to the complex task of object detection, there are many less positive aspects that, in comparison with the progress witnessed, not only in its family but among other methods of different algorithmic origin, leave it a little behind. Among these, the fact that R-CNN is quite slow should be highlighted. Since each input image it calculates about 2000 ROI's to be examined during the entire model pipeline, it leads not only to a time-consuming process but also to a very high computational complexity, which consequently leads to a lack of internal memory in a short time [26]. This factor can be decisive for real-time applications, such as self-driving cars [28].

Taking into account the disadvantages of R-CNN it is necessary to find a solution that is an end-to-end DL model, i.e., that does not involve the need to compute deep CNNs, that does not consist only of standalone algorithms together and that, consequently, has an automated learning way that involves the use of the properties of deep neural networks [26]. In response to these and other disadvantages presented by R-CNN, Fast-R-CNN [34] emerged.

Fast-R-CNN instead of starting right away with the module of region proposals and then moving on to the features extractor, like R-CNN, proposes an approach where it applies the CNN features extractor to the whole input image and only then makes the regions proposals [34]. This way, instead of running 2000 CNNs over 2000 overlapping regions, only one CNN is run for the whole image [26, 34]. Furthermore, it reuses the use of CNN to play the role of a classifier, thus replacing the classic use of Support Vector Machines (SVMs) for this network pipeline module with a softmax layer [26]. In this way, we achieved the end-to-end DL model we were looking for.

The training of a Fast-R-CNN becomes much faster because all the components are in a single CNN. Despite this, there is still a small problem associated with this model, which is the fact that the selective search algorithm is too slow and is out of context with the proposed general CNN architecture. In order to achieve a complete end-to-end DL model [26], it is still necessary to find a viable solution that can include all the modules and procedures inherent to object classification and localization, that is, to find a way to include and combine the region proposal algorithm in the end-to-end DL network [87].

Faster-R-CNN [87], the third and last of the R-CNN family lineage, has once again come to the benefit of its family to try to adjust the disadvantages found in the last studied network, Fast-R-CNN. The proposed architecture for Faster-R-CNN is practically similar to the architecture of Fast-R-CNN, with the only difference that instead of using a selective search algorithm in the feature map to identify the region proposals, a Region Proposal Network (RPN) is used [87].

The input image is present to the network and its features are extracted with the help of a pretrained CNN. These features are sent in parallel to the RPN, where it determines which sites or regions in the image are potential for the existence of objects, and simultaneously, sent to the Region of Interest (ROI) pooling layer where the features are extracted with the filter of a fixed size window [26, 87, 115]. The outputs of the two modules are then passed to two fully connected layers, one responsible for classifying the object itself and the other for predicting the coordinates of the location of the detected object.

Although the goal of the R-CNN family is to achieve real-time object detection [26, 33, 34, 87], they have not yet achieved this feat. First, because the training time associated with these is too high taking away the robustness of the final model, besides, the fact that the training of these takes place in a scenario of multiple phases leads to the following modules being congested because they are directly independent of the results of the previous ones, which consequently results in an accumulation of inference time [26, 87, 115].

2.3.2 Regression-Based Frameworks

All previously presented models belonging to the R-CNN family are region-based, that is, the detection task occurs in two distinct phases, one where the model proposes a set of ROI's using the selective search or an RPN, and another where the classifier only processes the regions that became candidates of the last phase.

What happens when we manage to join these two distinct processes into one, thus creating a single-stage detector?

Regression-based frameworks are single-stage detectors [26, 115]. These types of frameworks skip the region proposal phase and run the detection task directly on a dense sample with possible locations for the objects in the image. This type of approach, besides being simpler, is faster, which makes these frameworks much more of the goal of the R-CNN family, which is the detection of objects in real-time [26].

Overall, frameworks of this type tend to be faster but at the same time they end up sinning when it comes to accuracy, where one notices a general decline concerning the two-phase frameworks, such as the R-CNN family [26, 115].

An example of a regression-based framework is the Single-Shot Detector (SSD) network [60]. When this model was proposed in 2016, the network had achieved for the first time in long time records in terms of performance and accuracy for the task of object detection, thus achieving a result of 74% mAP for the standard dataset PASCAL VOC and Microsoft Common Objects in COntext (MS COCO) [26, 60, 115].

In single-stage detectors like the SSD, the convolution layers make the prediction, once subdivided into two phases [60], in only one, hence the name assigned to them. The image passes through the network once and the objectness score for each BB is predicted using a linear regression, which indicates the level of overlap with the ground truth. If the BB has an overlap value of 100% with the ground truth, it means that the objectness score is 1 [26, 60].

The approach made by the SSD network is based on the feed-forward process of the convolutional network, which produces a set of BBs of fixed size and, depending on what is found in these, applies a score that reflects the presence of an object. This process is then followed by a phase where NMS is applied [26, 60], which then leads to the return of the final detections made by the model.

In architectural terms (figure 2.10), the SSD model is composed of a base network for the extraction of feature maps [26, 60, 115], which normally corresponds to a pretrained network (for example a VGG16 network or a ResNet network) [26] used to make the classification of high-resolution images. After this, a series of convolution filters are added to the base network to progressively decrease the size of the target image, so that features of various dimensions are generated. Finally, the NMS application is used to eliminate overlapping boxes and keep only one box per object detected [60].

In figure 2.10, it is possible to see that the network makes a total of 8732 detections per class, which are then sent to the NMS layer so that they are reduced to one detection per object [26].

A very important aspect to be said about SSDs is that they are robust even when in the image there are features and objects of varying dimensions [26, 115]. To deal with this factor, the SSD uses a varied and tunable set of different convolution layers that vary in size, which leads to the extraction of features that vary in all sizes, thus allowing the SSD

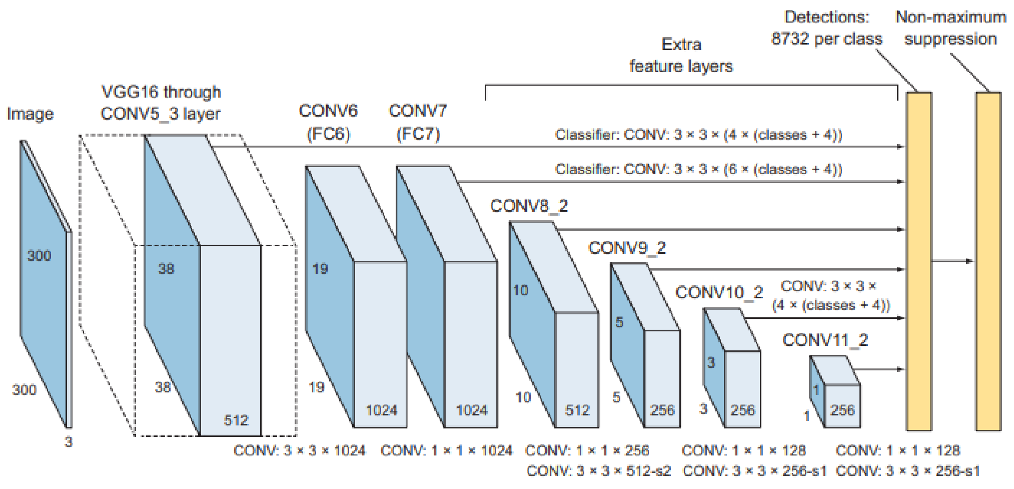


Figure 2.10: SSD architecture [26].

to become generalized in terms of object detection, and invariant to the changes that occur within the image.

Like the R-CNN family, You Only Look Once (YOLO) [84] is a family of networks specialized in object detection, developed by Joseph Redmon, and which has grown over the years since his first lineage, created in 2016 (YOLOv1).

The YOLO family is a series of end-to-end DL models created to obtain the results of object detection in the minimum time possible [26, 84], a little like what happened with the R-CNN family. YOLOv1 [84], published in 2016, is a single-direction network that unifies the detector and classifier components. In the same year, YOLOv2 [85] (or YOLO9000) was created, and it appears with the capacity to detect more than 9000 objects. This network was trained by two large datasets, ImageNet and MS COCO, so it obtained a result of 16% mAP, which is not good compared to the models that were mentioned earlier. Two years later, the third iteration of this family, YOLOv3 [86], appears. It appears with a larger dimension concerning its "ancestors" models, which guarantees it to obtain a 57.9% mAP in the same benchmarks datasets run in YOLOv2 [26, 85]. This makes it the best of its family and the best networks in general for object detection.

Compared to the "rival" family, the R-CNN family, they have relatively lower accuracy, although they are much faster and approach the objective of detecting objects in real-time [26].

YOLO networks predict a limited number of BB that are generated by subdividing the original image into a cell-matrix [26, 84, 115]. Each cell directly predicts the BB and the classification to be assigned. The result is a large set of candidate BBs which are later consolidated into the final forecast bypassing the NMS layer.

Taking into account that a progressive evolution within the YOLO family has emerged, it is wiser to just look at the architecture of YOLOv3 [86], the most recent network and the one with the best results of all three.

In functional terms, YOLOv3 [86] initially divides the initial image in a matrix of S x S cells. If the ground-truth center falls into one of the cells, that cell is responsible for detecting the existence of an object. Each cell provides a B number of BBs and their respective objectness score values as well as the class to which these objects belong [26, 86, 115]. Contrary to what we have seen, in YOLOv3 a sigmoid function is used instead of a softmax function, because a cell can have more than one class. The use of this function not only makes the model more robust but also gives us the freedom to implement multilabel classification more securely [26].

Overall, the architecture of YOLOv3 [86] (figure 2.11) contemplates several phases similar to what has been seen within networks of the same kind. In 2020, the last "official" member of the YOLO family, YOLOv4 [14], is presented by improving YOLOv3's AP and FPS by 10% and 12%, respectively. By combining a new set of features and techniques, like FropBlock regularization, Mosaic Data Augmentation, Mish activation or CIoU loss, that combined achieved state-of-the-art results: 43.5% AP (65.7% AP50) for the MS COCO dataset [114] at a real-time speed of 65 Frames Per Second (FPS) on Tesla V100.

In addition to this latest version of the family, YOLOv4, several other improved versions of this last version of the family have appeared, such as Scaled YOLOv4 [109], the most accurate (55.8% AP MS COCO test-dev) among neural network published.

Scaled YOLOv4 [109] lies on the Pareto optimally curve — no matter what other neural network you take, there is always such a YOLOv4 network, which is either more accurate at the same speed, or faster with the same accuracy, i.e. YOLOv4 is the best in terms of speed and accuracy.

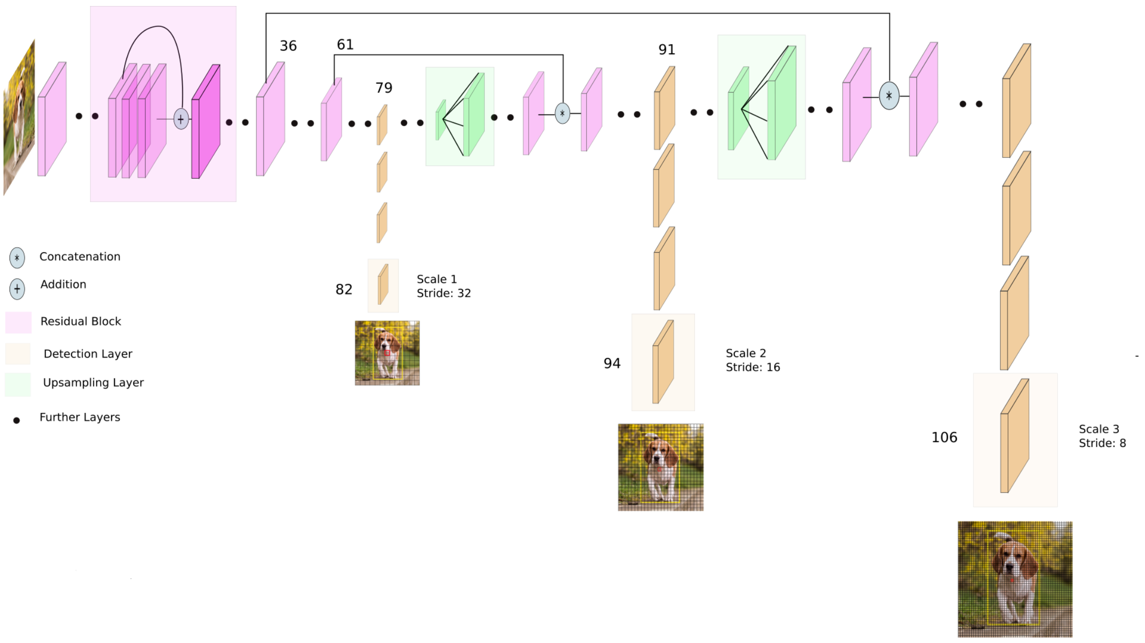


Figure 2.11: YOLOv3 architecture [26, 86].

2.4 Optical Character Recognition

Within the CV area, over time there have always been innovations aimed at facilitating work that for many years was considered manual and laborious. One of these tasks involves, for example, reading data from a financial statement to be entered into a database automatically, that is, detecting the various characters in the document and then classifying and recognizing them. The use of this type of technology, where it is possible to distinguish digital textual characters contained in images or physical documents, such as scanned documents, is called OCR.

The first idea of OCR technology was introduced through the invention of the retina scanner, a digital image transmission system that made possible the digitization of many documents [12, 65]. The first implementation of OCR software was made in 1929 and patented in 1935 by US Patent [39, 103], and it was a reading machine used for reading, at the date, military messages. The available commercial OCR software can be classified into four distinct generations based on their efficiency, robustness and accuracy. The first generation of OCRs was only capable of recognizing characters of a given font and shapes. The first generation of OCRs to be commercialized on a large scale was the IBM 1418 [12, 69]. They used template matching logic to extract the characters. The second-generation of OCRs are much more capable of recognizing scanned characters, and in addition to this, they were also capable of knowing handwritten characters. From the second to the third generation, the significant differences remained for the improvements in accuracy and robustness. For the fourth and final generation of OCRs, the power of scanning and recognizing characters has increased significantly. Due to the ability to distinguish text from complex background fonts, such as advanced text assemblies, tables and layouts with multiple horizontal and vertical lines, detection and recognition of mathematical symbols and special characters, and in addition, the ability to extract from low quality and noisy documents has emerged.

After the beginning of OCR technology within the fourth generation, many improvements and advances were made. Not only to improve the performance of the OCR engines but also to improve the reliability and quality of the information being extracted. In a way, we can divide the effort made over the years into two distinct phases: the detection phase and the recognition phase [12].

In terms of evolution in character recognition, Bensal et al. [11], in 2000, proposed a system that used a word dictionary tailored at the end of the system pipeline. From this addition, it is possible, at the time of segmentation, to perform an analysis of whether further tokenization of the detection buffer is needed or not, based on what is present in the engine's dictionary. In 2004, Kang et al. [50] proposed a system with the same focus as the previous one, but, in this case, the author took the output characters, and, from these, he stochastically applied pairing algorithms to produce words accurately. This method became more efficient for handwriting recognition problems.

In character detection, it is relevant to highlight the study carried out in 2006 by Liu et al., where they proposed a different methodology for detecting low-resolution characters. It is a technique that addresses the quality of the input image and the recognition of the detection performed. The analysis of the input image quality is emphasized in the first instance, followed by an evaluation by the returned color scales, causing the recognition method to select the outputs with higher input quality, which in classification terms results in having a higher likelihood of being the correct decision. In 2012, Yutao et al. [113] developed a new idea for the extraction system, proposing a hybrid system based on regression, which implies the application of a wavelet transform. The results indicate that using wavelet transform for feature extraction from input characters made the feature vector smaller. Also, using a clustering algorithm, more features were selected and sorted, thus achieving optimal performance in the classification task. In 2013, Giri et al. [32] used an algorithm that is very close to the human character detection and recognition system. The algorithm analyzes the minimum amount of data and makes the whole system independent of the font or size of the input characters.

Today, the approaches of OCR engines can be divided into 2 categories: offline and online approaches [12].

The offline approach is the detection and recognition of scanned characters in documents or static physical input images. The online one consists of the stream recognition of the written characters. It is in our best interest to explore the offline approach and the concepts it uses, given the scope. The engines available today are designed to be as multilingual as to cover a varied batch of languages and thus generalize the OCR solution to either landscape character scenarios or document or static character scenarios [12].

Offline multilingual OCR engines are often designed for a particular script or language. The ideal approach for a multilingual OCR software built from scratch would be to optimize a network already trained for the task of detection in different languages, and through fine-tuning, specialize it for the target languages. Most OCR engines can only process with a limited set of languages because, in terms of research, there is a tendency to use language-specific features and models to simplify the inherent problems. Furthermore, the task of detection and extraction becomes even more complex when more than one language type occurs in the target document or scenario [79].

A multilingual OCR engine consists of a pipeline system (figure 2.12), where processes like preprocessing, feature extraction and classification or recognition. That means that if we face a multi-language scenario, an initial script is needed to classify the languages, which consequently ends up recursively using the pipeline processes, making the task more rigorous, but at the same time, slower [79].

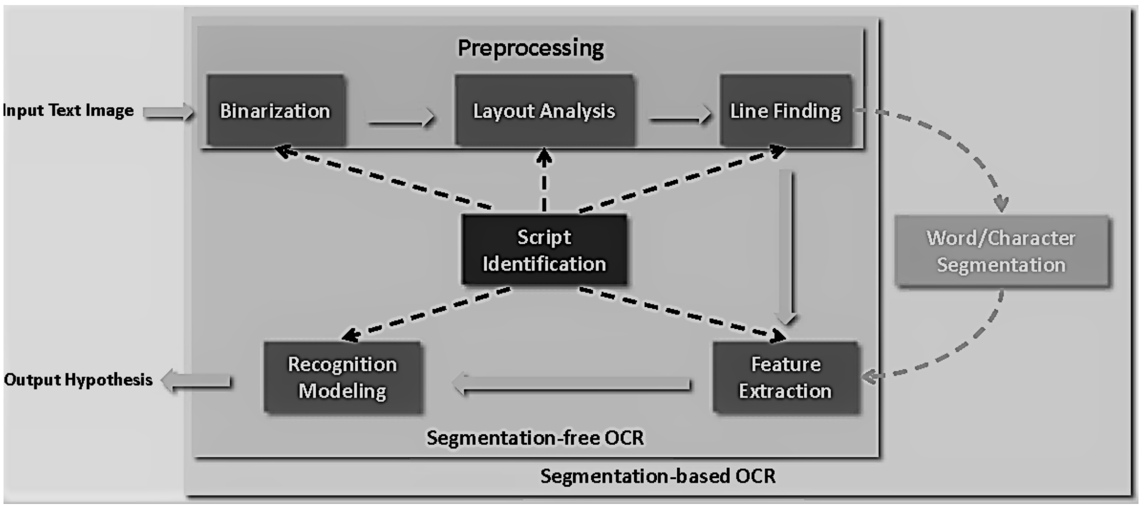


Figure 2.12: Typical components of a multilingual OCR system. [79].

As seen earlier in the section about image processing, a good application of algorithms and preprocessing operations can make some trivial tasks in the CV much easier. The input to an OCR engine is usually a color or gray-scale image. If an image contain several distinct elements, such as tables, figures, and other types of factors that may disturb the consequent stages of the engine and information extraction, the input must be processed. In this case, the application would perform an initial binarization to the image to normalize the input image in the OCR engine [31, 51, 79]. The next step is identifying the text regions present in the document, in order to isolate the main graphic textual components from the rest of the document. This process is called page segmentation [16, 18, 38, 79]. After the regions are found, a line search algorithm is applied to extract textual lines from the scanned. The methods associated with this process assume that the text is horizontal. As already seen in the Image Processing section, the projection profile-based method can assist in accomplishing this step [62, 79]. Depending on the pipeline architecture, there may also be a final preprocessing step where the tokenization of the textual lines detected in the previous target is done. A disadvantage in this model of word or character segmentation [79, 111] is that, in the context of multilingual OCR engines, there must be a focus in terms of language set, otherwise there may be a large discrepancy in comparison with the ground truth.

Following the architecture of a classic multilingual OCR engine, there is then the so-called optional script identifying the target language [20, 49, 79], after the detection and clipping of the textual elements that were once isolated. Optional because most OCR engines have a priori the set of languages selected.

The recognition step is the process of identifying the text by classifying each character or word [79, 111]. This stage is the most challenging within the world of OCR engines. Overall, approaches to text recognition in conventional scanned documents can be divided into segmentation-based and segmentation-free methods. The difference between the two is in the output returned at the segmentation stage, if this is done in small units as described above it is a segmentation-based method.

Segmentation-free approaches analyze individually sequentially the entire segmented textual lines. Hidden Markov Models (HMMs) can be integrated into multilingual OCR engines using a sliding window approach, whereby features present in the input are extracted and filtered, as has been done before in speech recognition [70, 71, 79]. Segmentation-free approaches that use this approach are fine-tuned to operate and extract in the handwritten text as input.

Segmentation-based OCR systems, such as the open-source engine Tesseract [52, 79, 96], performs the character segmentation step and refines this segmentation using a recognition-driven framework. Tesseract performs a bipartite classification for character recognition, in first, by making a list of potential candidates are identified for each of the as-yet-unclassified characters based on a look-up table matching strategy [52, 96]. In a second step, to safeguard the accuracy of the engine, a further measurement is made in terms of similarity between the ground truth and the unlabeled characters. The hypothesis for the final word recognition, that is, the correct regrouping of the characters, is done by simple linguistic analysis. These methods require much more time to reconfigure the segmentation algorithm for other target languages, and in addition, become weaker as the set of languages increases [79, 97].

Apart from this original way of recognizing and identifying characters, the internal structure of Tesseract (figure 2.13) is quite similar within segmentation-based approaches [52, 79, 96].

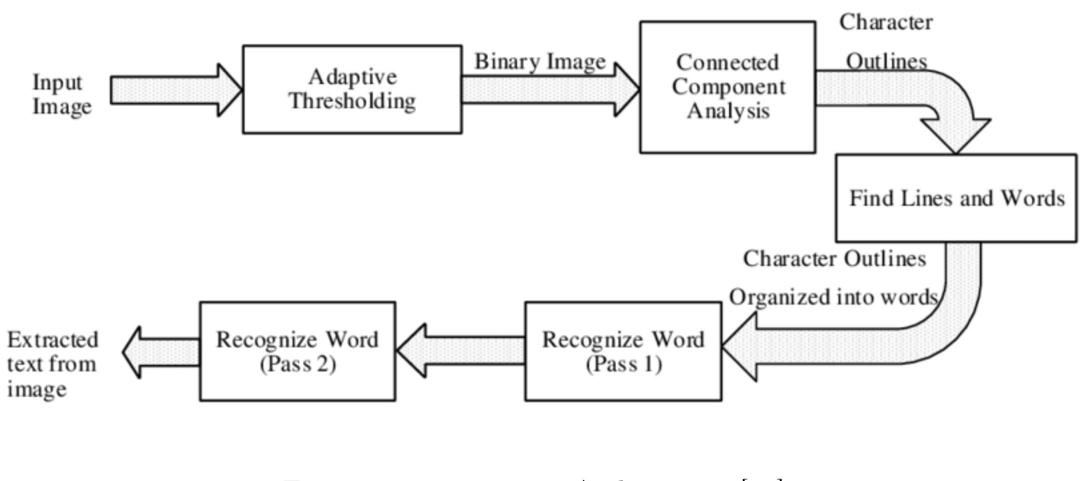


Figure 2.13: Tesseract Architecture [52].

The heart of Tesseract is the Character Segmentation, Language Models, and Beam Search models that compose the recognition stage (figure 2.14) [52].

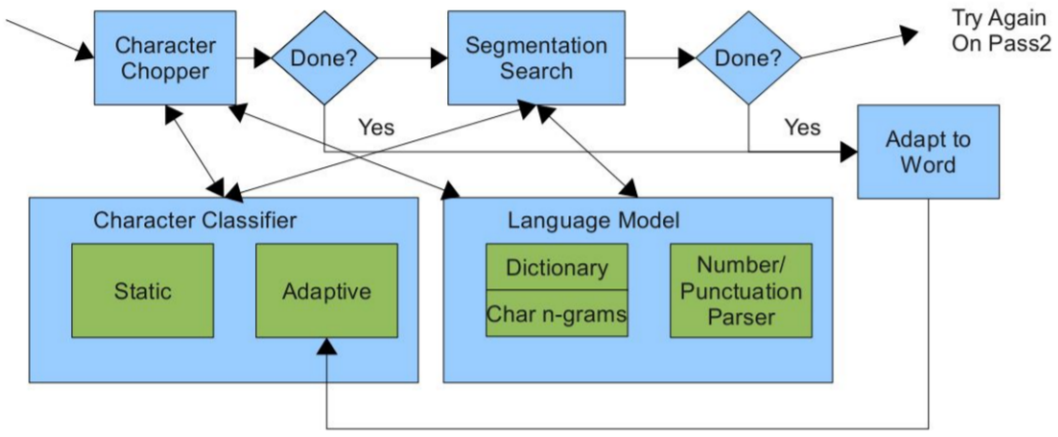


Figure 2.14: Tesseract Word Recognizer [105].

In character segmentation, Tesseract relies on segment-first using only geometry to isolate the different detected characters. Then is applied a maximal chop technique, which combines beam search. The usage of a second process recurring to beam search could cause over-segmentation. The sliding window technology avoids the segmentation of a set of unrelated characters. The principle of Tesseract to avoid over-segmentation and to perform well in character segmentation is to chop as needed, then combine as needed [52].

The over-segmentation aim is to maximize recall of chops with the compromise of reduction in precision. The segmentation set of possibilities is the computation of a directed graph, which contains all the possible interconnections between the different characters (figure 2.15).

In the example of figure 2.15 we are searching for the correct word, which is "higher". It is possible to see that by cropping or adding to much we lead to a solution, that is very distance from the a expected one. Also it is relevant to pay attention at the path given by the graph on the search for the right word.

It is relevant to say that the integration of language models as finite state machines in parallel execution to the segmentation graph search. The processes then combine “probabilities” to upgrade the sensibility of the recognition stage.

In the recognition and classification phase of the detected word or character, a ranking is made based on the word rating value that is given by the equation 2.16, where factors such as, number of inconsistencies, word size, the average blob rating, among others compose the weight for word factor. Then with the calculation of some distances, it is possible to obtain a set of possible results where those with the smallest distance between them and the ground-truth values are the candidates for the final output, as depicted in the montage of figure 2.16.

$$Word_{Rating} = word_{factor} \sum outline_{length} \tag{2.16}$$

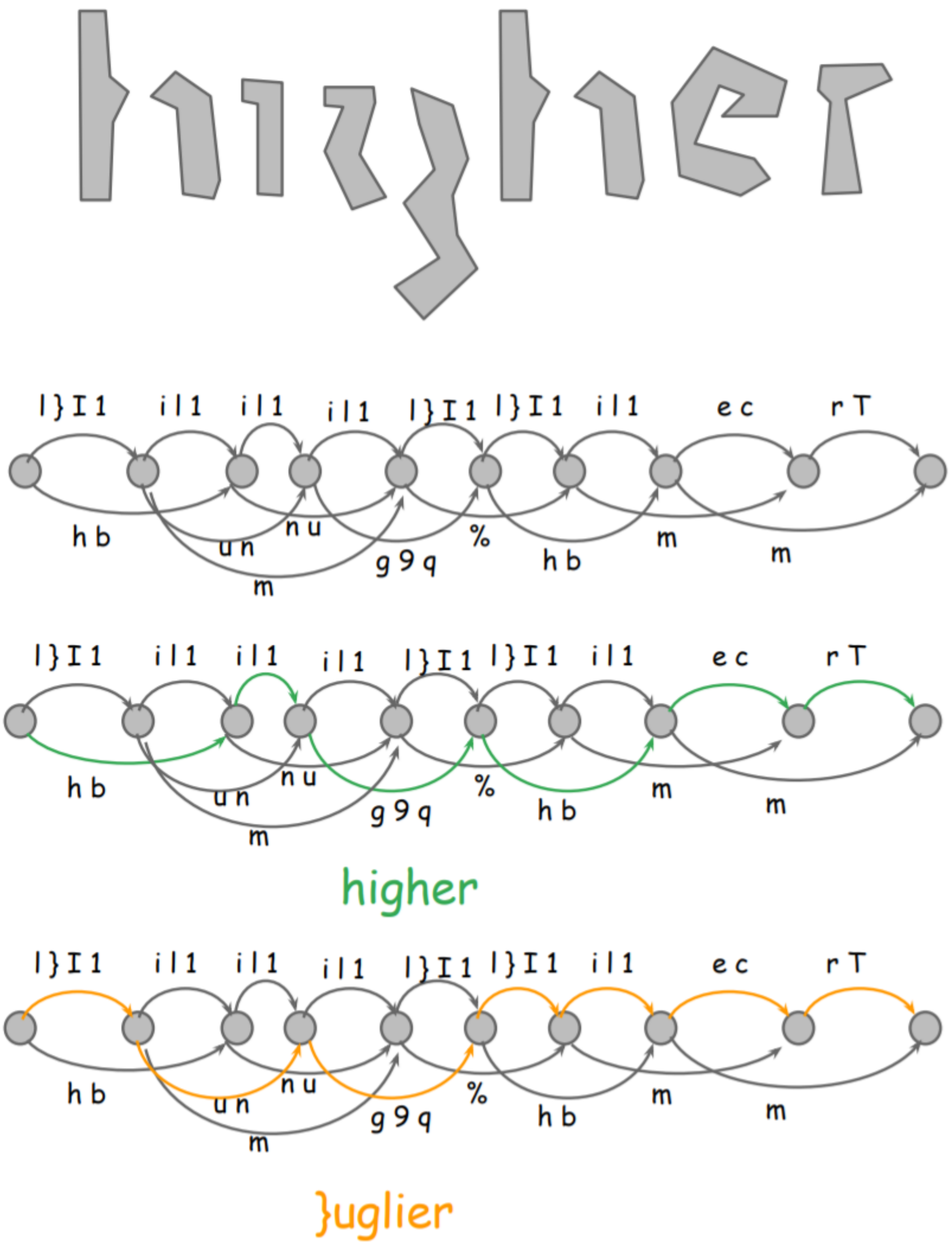


Figure 2.15: Segmentation Graph Search Overview. [105].

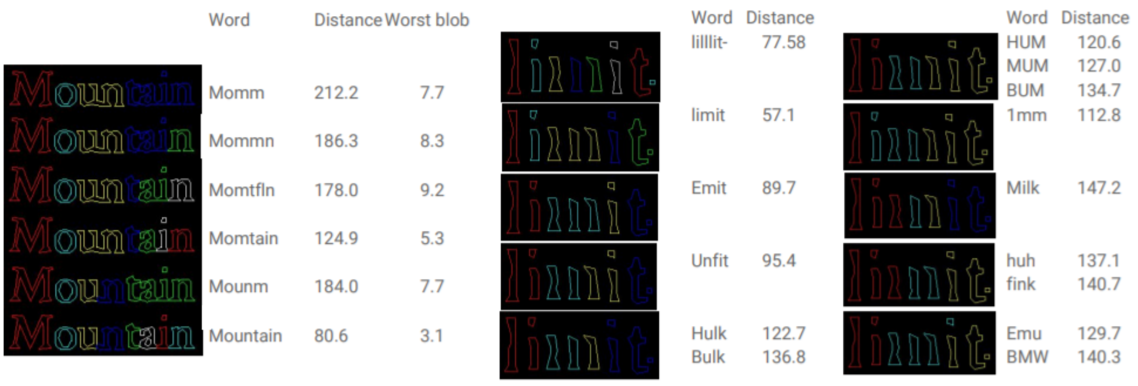
where,

$$word_{factor} = \sum_{f=1}^{K} \dot{W}_f, \tag{2.17}$$

being $\dot{W}_f$ the vector of weights of the following word features:

- Mean Blob Rating
- Number of inconsistent spaces
- Number of inconsistent char types

- Number of x-height's inconsistent
- Word length



Word	Distance	Worst blob
Momm	212.2	7.7
Mommn	186.3	8.3
Momtfln	178.0	9.2
Momtain	124.9	5.3
Mounm	184.0	7.7
Mountain	80.6	3.1

Word	Distance
lilllit-	77.58
limit	57.1
Emit	89.7
Unfit	95.4
Hulk	122.7
Bulk	136.8

Word	Distance
HUM	120.6
MUM	127.0
BUM	134.7
1mm	112.8
Milk	147.2
huh	137.1
fink	140.7
Emu	129.7
BMW	140.3

Figure 2.16: Chopping versus Combining. [105].

In the case on the left of figure 2.16, we can observe the segmentation process of the word "Mountain" where it is possible to verify that the output that is closer to the desired one is the one that contemplates a process that preserves more the segmentation (over-segmentation) than a more cohesive process. From this chop versus combining dilemma, we can observe that, in comparative terms, in the case of the word "Mountain" in which it was necessary to chop more than combining, there was, overall, a worse performance than in the process to its right for the word "limit". Of course we cannot generalize this idea because each word has its challenging aspects, and whether it is longer or shorter, what matters, in the end, is codependency and background learning of the relationship between characters. Furthermore, over-segmentation, as is observable in the results in figure 2.16, leads to an increase in the distance metric, which consequently means a lowering of recognition confidence (less distance, the less the classification error).

In addition to Tesseract, many other open-source projects have been emerging to address needs within the CV area. One such project, which remains in active participation to this day, is EasyOCR [47].

EasyOCR is an open-source ready-to-use multilingual OCR with more than 80 languages available. EasyOCR consists of a normal preprocessing stage [47] on a first stage. In a second stage, the detection process, using the Character Region Awareness for Text Detection (CRAFT) algorithm [10] is done, followed by the recognition stage, which uses a Convolutional Recurrent Neural Network (CRNN) recognition model [93]. CRNN is composed of 3 prerequisite components, a feature extraction component, which uses a Resnet50 [41], an Long Short-Term Memory (LSTM) [42] for sequence labeling, and a decoding stage, using Connectionist Temporal Classification (CTC) [36].

To directly challenge Tesseract, from an open-source perspective, EasyOCR is a novelty when it comes to the detection part because it uses in its structure (figure 2.17) the CRAFT algorithm [10].

The CRAFT algorithm [10] is a new proposal for text detection in a natural or synthetic environment, made possible by exploring the affinity between characters. Due to the lack of annotations at the individual character level, the proposed framework explores the solutions in the two scenarios, natural and synthetic. The network needs to be trained with the most recent affinity representation. The craft was evaluated on six different benchmarks, including TotalText [21] and CTW-1500 [112], which contain text with several levels of

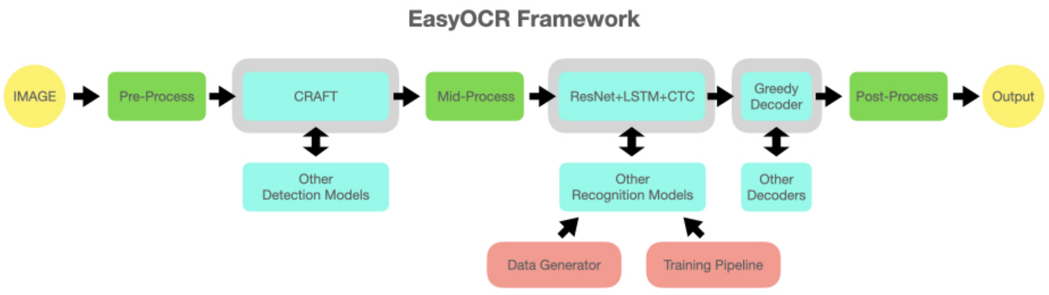


Figure 2.17: EasyOCR Framework Structure Overview [47].

curvature, rotation, orientation, and different scenarios to simulate both the natural and synthetic environments (figure 2.18).

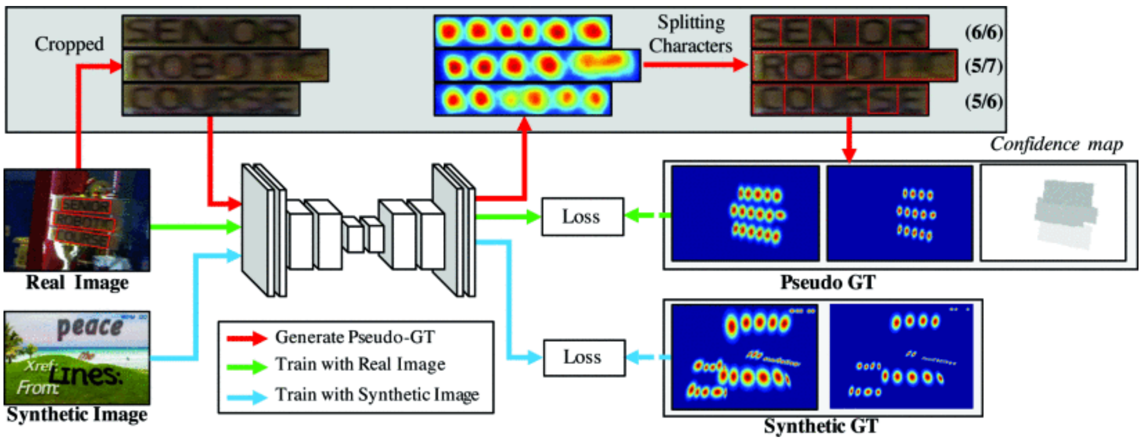


Figure 2.18: Overall training stream for CRAFT, where the training is carried out using both real and synthetic images in a weakly-supervised fashion [10].

In practical terms, the CRAFT algorithm [10] is designed with a CNN that produces the region score, used to locate the different and isolated characters in the image, and affinity score, which is used to group the different characters into a single instance.

The final output can be given in a variety of forms, including word boxes, character boxes, and polygons. Because the assessment methodology for datasets like International Conference on Document Analysis and Recognition (ICDAR) [45] is word-level IoU, it is relevant to study and investigate how to create word-level BB, QuadBox, from the predicted data using a simple but effective post-processing step (figure 2.19). An advantage of CRAFT is that it does not need any further post-processing methods, like NMS. Since we have image blobs of word regions separated, the BB for a word is simply defined by the single enclosing rectangle. On a different note, the character linking process is conducted at a pixel-level, which differs from other linking-based methods relying on searching relations between text components explicitly [10, 43, 92].

In summary, the CRAFT algorithm is, according to the published results [10], extremely flexible when it comes to detecting text in complex or noisy scenarios and is also sensitive to conditions such as text warping, pixel quality, curvature, or not, of the text, and arbitrary orientation.

Many efforts to build classifiers endowed with low error for recognizing and labeling characters for word-formation have been made over time. The Recurrent Neural Networks

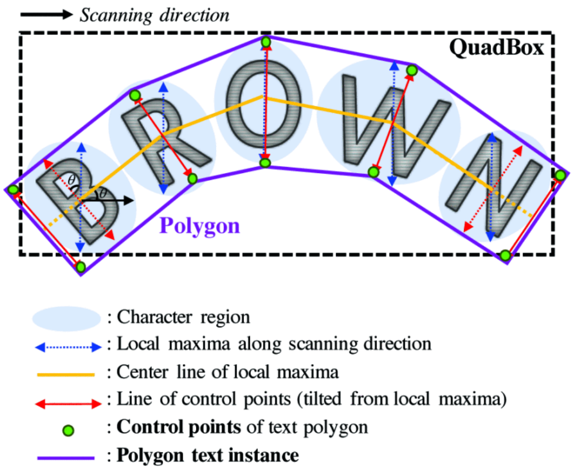

Scanning direction
QuadBox
Polygon
θ
: Character region
: Local maxima along scanning direction
: Center line of local maxima
: Line of control points (tilted from local maxima)
: Control points of text polygon
: Polygon text instance

Figure 2.19: Polygon with the character regions of intereset [10].

(RNNs) are strong learners for this type of task, but because they require pre-segmented data and post-processing transformations to turn the output characters into sequence labels, their use has been limited. In [36], a strategy for training RNNs is presented.

The difference between CTC and other classifiers is that CTC does not explicitly segment its input sequences, which means that, there is no need now to locate ambiguous label boundaries and opens the possibility to grouped label predictions which, consequently will minimize the loss on the word and character classification. It's relevant to remain that an approach "standalone CTC", i.e., without the segmentation mandatory phase, would seem problematic.

Another interesting feature of this decoding approach is that it does not explicitly support inter-label dependencies, in contrast, to what happens in the Markov chain's approaches [36, 47]. Hierarchy within structured data is not yet a feature of the CTC decoder, but it is already being considered for future work. Also, the TIMIT speech corpus [30] demonstrates its advantages over both a baseline HMM and a hybrid HMM-RNN.

In comparative terms, the Tesseract engine [52, 96] is optimized to run on Central Processing Unit (CPU), while EasyOCR is optimized to run on Graphics Processing Unit (GPU), with state-of-the-art values in terms of recognition time. Much is due to the previously mentioned structure of EasyOCR, and because it is composed of a model that internally uses multithreading computation in the GPU environment, thus optimizing the character recognition and decoding pipeline tasks. In a test performed with over 2000 sample images with letters and numbers [10, 47, 105], Tesseract showed an error rate below 1% when detecting alphabetic characters, while for numeric error rate around 6%. EasyOCR, on the other hand, showed an error rate below 5% for alphabetic character detection and an error rate below 2% for digit detection. This indicates that, in general, EasyOCR is currently more specialized in correctly detecting and recognizing characters of numeric origin, while Tesseract is more specialized in detecting and recognizing characters of alphabetic origin. Of course, despite this, it should be noted that Tesseract generally performs better than EasyOCR in terms of the time of prediction and overall classification. [10, 47, 105]

In addition to the optimizations made by the team so that the model supports more different languages (it now has more than 80 different idioms), several improvements have been made in the recognition phase, such as the use of external and donated knowledge sources to enrich the network.

The classical approach through the implementation of CV algorithms involves the application of filters that make the characters stand out from the background, the application of an outline detection algorithm to recognize the characters one by one, and in the last step of the application of an image classification process to recognize the characters. We can then say that the classic OCR system is an integrated pipeline of several sub-processes. Still, about the structure of the classical approach, it is relevant to highlight that the detection of contours is a challenge due to a lot of manual fine-tuning, making it impractical in most cases.

OCR yields great results only on specific use cases, but in general, it is still considered challenging [47, 52, 96]. There are indeed good solutions for certain OCR tasks that do not require DL, using only some morphological operations or libraries from CV software to achieve the same result. However, to step forward towards better and more general solutions, DL will be a tool to upgrade and help the TIE systems to evolve. Not only because of the needs relying on upon in the multilingual OCR complex, but also in the reliability of the software to deal with scenarios with very hard background, with too much noise or blur, or with an overall quality unfeasible for extraction.

This page is intentionally left blank.

Chapter 3

Processing and Consolidation of Digitized Customer Information

In Altice/MEO's sales, identification documents are scanned and associated with the customer profile, but the inherent identification data is not collected automatically for the Customer Relationship Management (CRM) systems. These data are very relevant to guarantee the best experience of the customer, by supporting future commercial actions.

Most digitized documents are a single or a collection of A4 pages in black and white, with information about the contract between the customer and the service or product. Sometimes, in these documents, we can find some personal information, mostly located in the citizen ID cards or other types of Identity Documents (IDs)

In a simplified way, and taking into account the objectives already specified at the beginning of this document, the procedure of extraction of scanned personal information comprises two primary tasks:

- Detection and Recognition of the ID type
- Extraction of the identifying fields present in the detected ID

Based on the knowledge gathered in the previous chapter, it is now necessary to understand what kind of methodologies and strategies need to be adopted to reach a robust final solution capable of computing the two mentioned tasks to no loss of performance or information.

Objectively, the first task would be to isolate the extraction area, that is, to detect and isolate the ID or part of it. Based on this knowledge, it will be much simpler to extract the information since we would then be performing extraction of the fields on a much smaller area of interest, which would, consequently, imply a significant increase in extraction confidence. It is possible, by adopting several strategies from the Computer Vision (CV) area, to isolate the IDs. Through morphological operations, transformations at the pixel level, and producing contours in the detection area, it is possible to find and isolate the desired coordinates of interest, that is, the coordinates that delimit the ID or part of it.

Despite the mentioned approach, we have previously studied that we can have reliable object detection approaches with Machine Learning (ML) technology. Therefore, it is relevant to apply the necessary methodologies so that this type of approach can be tested in the extraction and detection of IDs. Since this is where we delimit our "work" area, it is relevant to highlight that, logically, this phase compromises relevant importance in the rest of the logical pipeline of digitized information extraction, In practical terms, this phase of the process, where we search for contours and infer textual character extraction from the scanned documents, becomes complicated to manage in terms of implementation, since we are expecting to receive scenarios with random ID location and orientation.

Hence it is relevant to establish that a good Optical Character Recognition (OCR) engine to adopt will have to take into account the characteristics of a robust engine capable of behaving adequately to extract characters of Portuguese and English origin simultaneously, i.e., a multilingual OCR engine.

However, the challenge behind this problem is related to the quality and integrity of some digitized documents stored in CRM. When we refer to "quality", we are talking about the definition of the final image in the document, because, what we usually see in the dataset and general input is that, most of the documents have poor conditions and a lot of noise or blur, which can lead to a bad functionality of the ID detection or a bad or inexistent extraction of the major personal information inside the document. So, we can say that image processing has a major contribution for the success of the final automated system, which we can consider as a great motivation and also as an important evaluation metric of the final output. In the first part of this chapter, we will analyze the type of documents expected and their components. Further, the limitations inherent to the project and the architecture for the extraction system will be discussed.

3.1 Input Analysis

A record made between a customer and a department within MEO may result in a documentary record of the operation performed. For example, when a contract at MEO's sales department, or, the registration of a new service by a customer at an MEO store happens, a document that records the operation is created. Usually consists of a duplicate form and, at the end of the document, sometimes a scan of the identification document of the customer or customers is found.

This document is later processed with all the information about the transaction. If it is for example the sale of a service to a specific customer, the customer's file will be registered in a sales form together with his scanned personal documents. In order not to lose the customer data information in the CRM database, these documented operations will be named with the ID in the database that relates the processes to the target customer, the FileNet. The documents that we will find in the CRM environment will be named with the eight digits that make up the FileNet they correspond to internally, called the FileNetID.

The usual scenario is composed of a document with several images, where most of those are contractual paperwork where the customer is informed about its service or product, followed by the digitized documents of the target customer (figure 3.1). The pages with the scanned personal documents are our extraction source.

In figure 3.2 we can observe some sample target pages regarding the primary goal of this project, extracting information from scanned ID documents.

Keeping in mind the scenarios in 3.2, we are not only interested in identifying each of the IDs found at a FileNet page, but extract the information from each one and then analyze how to use it for CRM data completion. Also, there will be cases where, for example, one of the IDs parts found is upside down, which will consequently increase the complexity of the extraction task, or even cases where the ID itself is the only element composing the

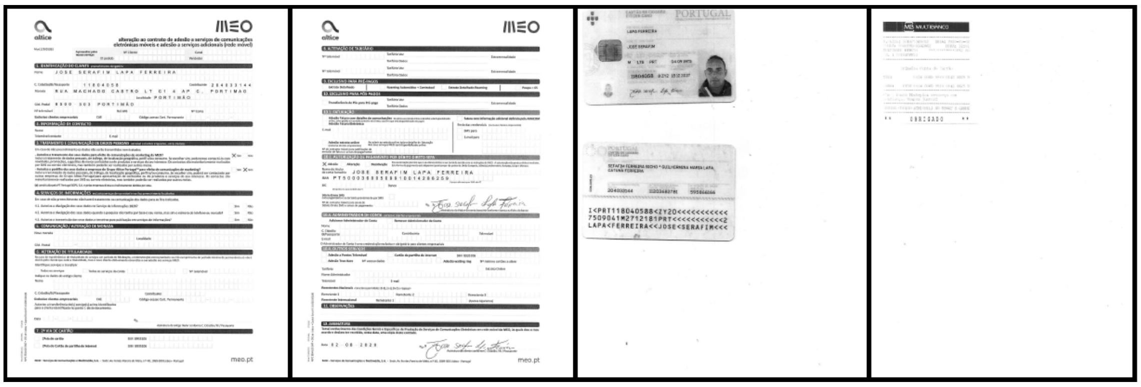


Figure 3.1: Usual FileNet Scenario

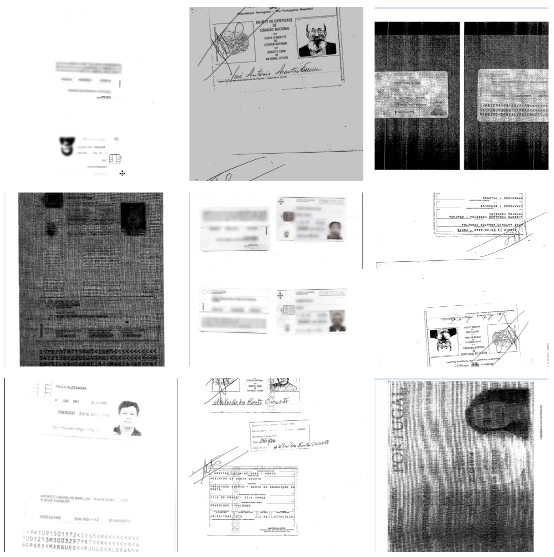
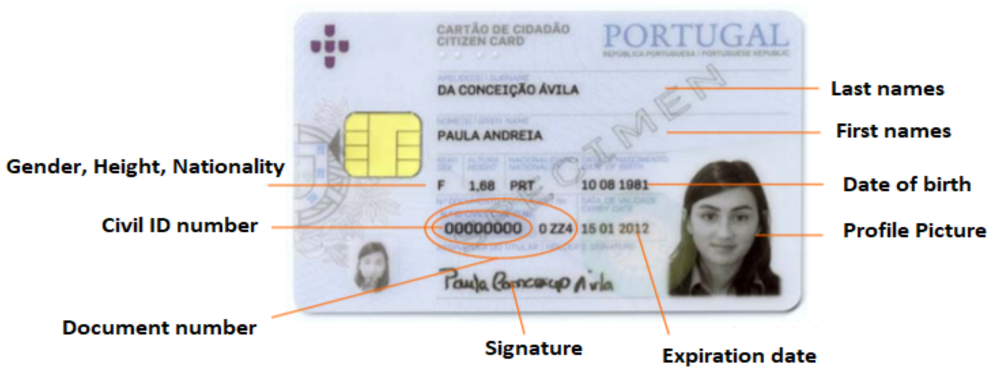


Figure 3.2: Different Scenarios of the IDs in the Digitized Documents of CRM, showing the complexity of the problem in scope

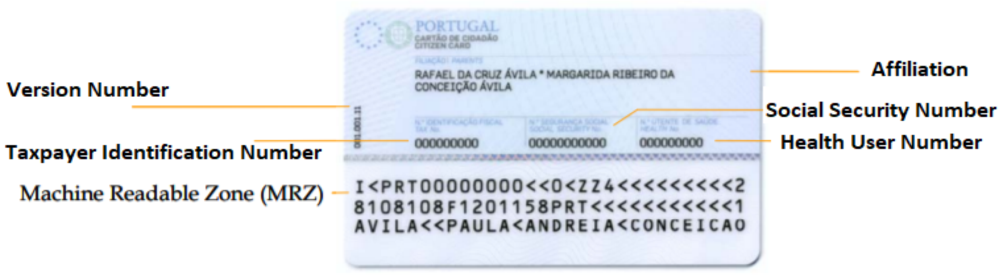
digitized document, i.e., the document itself is the ID. Given that there can be multiple detections of parts of the same or different IDs in a FileNet document, it is also necessary to assume the best strategies in terms of data structure so that later on, in terms of post-processing, there can be a coherent and final assessment of what information should be processed for each ID detected in the total document.

Among the IDs that are expected to be found in the different FileNet documents, the ones that stand out internally for Altice/MEO, from a CRM perspective, are the Citizen Cards (in Portuguese, Cartão de Cidadão (CC)) (figure 3.3), the Identity Cards (in Portuguese, Bilhete de Identidade (BI)), the Residence Permits (in Portuguese, Autorização de Residência (AR)), and the Taxpayer Cards (in Portuguese, Identificação Fiscal (IF)) (figure 3.4).

Since we have as input a set of labeled customer files, that can have or not IDs in it (figure 3.2), we must acknowledge that we need to reduce the computational complexity of the task of extracting information in every input file, because, we can have the case where there is no related or valid ID in the input. Besides, another problem in the dataset formed by these documents is that there is an unbalanced distribution of the found IDs classes along with the many input documents. For example, we are going to find more occurrences

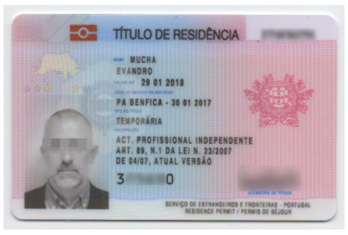


(a) Front side of the Portuguese Citizen Card

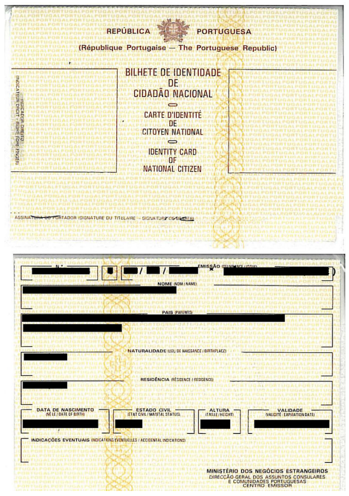

PORTUGAL
CARTÃO DE CIDADÃO
CITIZEN CARD
RAFAEL DA CRUZ ÁVILA * MARGARIDA RIBEIRO DA CONCEIÇÃO ÁVILA
000000000 00000000000 000000000
I<PRT00000000<<0<ZZ4<<<<<<<<<<2
8108108F1201158PRT<<<<<<<<<<<1
AVILA<<PAULA<ANDREIA<CONCEICAO
Version Number
Affiliation
Social Security Number
Taxpayer Identification Number
Health User Number
Machine Readable Zone (MRZ)

(b) Back side of the Portuguese Citizen Card

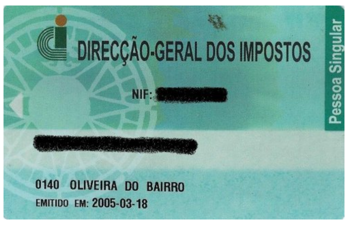
Figure 3.3: Portuguese Citizen ID Card (in Portuguese, CC)



(a) Example of the front part of an AR [3]


REPÚBLICA PORTUGUESA
BILHETE DE IDENTIDADE DE CIDADÃO NACIONAL
CARTE D'IDENTITÉ DE CITOYEN NATIONAL
IDENTITY CARD OF NATIONAL CITIZEN

(b) Sample of an BI [4]



(c) The Portuguese tax payer card (IF) [1]

Figure 3.4: Other types of IDs to be found.

for CC class than for BI class, because in Portugal the last one it is not anymore in usage, but there is still some old documents and information in CRM that have this type of document.

Regarding the quality of the dataset documents, most of the input files are in bad conditions, i.e, some of them can have a lot of noise derived from the bad scanning, or even expect some samples to have a highly complex background, various orientations, occlusions, flares [95], within many others defects associated with the digitized process inside

ID CLASS	TYPE OF ID	INFORMATION TO GATHER	CRM FIELDS DESCRIPTION
CC	**Citizien Card**	NAME	Complete Name
		NIP	8 digits long
			Type of Personal Identification = 1
		NIF	9 digits long
			Type of Fiscal Identification = 1
		Birth Date	Complete Date of Birth
			Consider only valid dates, later than today minus 110 years, and less than today minus 18 years.
BI	**Identity Card**	NAME	Complete Name
		NIP	8 digits long
			Type of Personal Identification = 2
		Birth Date	Complete Date of Birth
			Consider only valid dates, later than today minus 110 years, and less than today minus 18 years.
IF	**Portuguese Tax Payer Card**	NAME	Complete Name
		NIF	9 digits long
			Type of Fiscal Identification = 2
AR	**Visa**	NAME	Complete Name
		NATIONALITY	Country of origin
		VISA NUMBER (NIP)	8 digits long
			Type of Personal Identification = 9
		Birth Date	Complete Date of Birth
			Consider only valid dates, later than today minus 110 years, and less than today minus 18 years.

Table 3.1: Table with the data to be obtained from the scanned IDs identified in the FileNet Documents

CRM processes.

From this data, we expect to collect important information such as name, date of birth, tax identification number and the personal or civil identification number, in Portuguese, Número de Identificação Fiscal (NIF) and Número de Identificação Pessoal (NIP) respectively. After these fields are collected and processed, they are sent into the CRM environment, where later they are allocated to the corresponding biographical sheets (table 3.1).

Observing the figure 3.3, we can acknowledge all the textual and graphical elements present in the citizen card. These features can be used, for example, by an Object Detection model to recognize them in an initial phase. In general terms, we can subdivide the extraction for the CC into two parts, respectively, the two parts, the front and the back part of the ID. In figure 3.3, the front part comprises most of the elements to be extracted, specified earlier in table 3.1. From the front, it is possible to get three of the four fields required. It means that, even if the front part has a high success rate in terms of extraction, the forth element, the NIF, will always be missing. The NIF field, which, as specified in 3.3, only occurs on the back of the document. In other words, this means that to have a full complete extraction, both parts that make up the CC are necessary.

Still looking at the components of the CC, the back end comprises an alphanumeric code at the end of the card, the Machine Readable Zone (MRZ).

The MRZ code is usually at the bottom of the identity page at the beginning of a passport and, in the case of the CC, at the bottom of the back-side. The MRZ code includes the documents owner's data and a "Checksum" that prevents information fraud.

From the 91 digits that make up the MRZ, it is possible to achieve (if there is no error in the extraction of the characters) a total extraction of the elements present in the front part, which means that, in practical terms, an MRZ code, replaces the front. This analysis has a relevant impact on how the final solution will be formulated and organized.

As it is possible to see from the different scenarios presented in the figure 3.2 referenced above, there are several distinct challenges when it comes to image quality and image processing. These two perspectives are of utmost importance when we talk about extracting information from files with poor or no quality at all. It is these and other aspects that we need to take into account before any analysis or formulation of a solution. What strategies need to be taken to get around some critical scenarios that are presented, in terms of image quality? Is there a way to reduce the extraction region to minimize the possible loss of information? How to automate this detection?

Besides this obstacle, others exist. In terms of extraction, there can always be errors associated, either in the detection of the fields to be extracted or in the recognition of the characters, which will result in the creation of incorrect entries into the database. How can we minimize the error in information extraction? Which is more beneficial to exploit, character detection or character recognition? What is the state-of-the-art of open-source OCR models?

3.2 Previous Work

Internally, in line with the emergence of this project, an initial test took place using Kofax OmniPage Capture software [54]. This software incorporates an OCR engine trained to collect information in Portable Document Formats (PDFs), usually with a template. It was designed to collect data from financial statements and other types of documents that have a specific template.

The software besides supporting the OCR functionality is also able to process the scanned information from documents and all the classification components integrated with the CRM.

In an attempt to collect as many dates of birth as possible, a group from Altice/MEO investigating the usefulness of using dates of birth in a commercial marketing perspective used the software mentioned above. The software allows the use of regular expressions that can be programmed to identify the fields belonging to the citizen card. In the case of the field "date of birth", where the format is "dd/mm/yyyy", the regular expression introduced in the software would take into account the numerical separators as well as the expected length for the sequence of characters. By applying rules and regular expressions it is possible to filter the results of the OCR engine to satisfy the needs inherent to the use of the software.

The experiment was performed on about 250 scanned documents and only included the detection of characters. No validation on the extracted data was done. The software evaluation used only documents containing citizen cards. In terms of accuracy, the rate was around 60% for the extraction of birth dates only. This accuracy value represents only the achievement in terms of the number of fields extracted but not validated.

Since this is an informational field of relevance to sales and marketing, not only this field but all other identifiable fields must be validated. This was one of the limitations of the enterprise extraction in the Kofax software experiment [54]. Not only are we not sure that the fields being extracted are correct, but there is no room to bypass the search rules for other fields that are more complex and do not have detection anchors.

The main problem detected by the team, and also mentioned here, was indeed the quality of some documents and their layout, i.e. the layout of the scanned ID documents on the pages of the target documents. Most of the problems stem from the overall image quality, the noise present in many of the documents, the orientation of the personal document scan, the image deconstruction that exists in some areas of the documents, as well as contrast, brightness, among other image processing challenges.

Although the initial trial had some success, this was a free trial of the software. Looking from a company perspective, in which thousands of new documents get registered every month, the cost associated with the purchase of licenses of products like Kofax is not worth it when there is a way to automate the process permanently and keep the process integrated internally for company use only. This way, there is no need to opt for Application Programming Interfaces (APIs) that are external and, which can consequently compromise the security of personal data stored internally in the company. It is therefore useful to find a solution that can integrate the automated part of the detection of the types of documents and their automatic extraction and classification, in an internal environment integrated with the company with exclusive use only of the processes intrinsic to the company.

Although the experiment focused exclusively on the "date of birth" field, it has to emphasize that when collecting fields from identification documents, the highest priority fields are those that represent the customer's identity and are necessary for the customer relationship. Starting with the name field, and then the unique identification fields, i.e. the NIF and NIP. The name and NIF are the most relevant fields, but it is important to note that either one is only useful if collected together with the others.

3.3 Limitations and Scope

In general, the main component to be drawn initially will be the localization and capture of the different types of personal identification listed above. Taking into account that we are working on an ML component, we have to keep in mind that the models that are going to be in the final software are likely to fail, and so, right from the start, the supervised learning inherent to a good part of the project is a limitation. Furthermore, still speaking in supervised learning, the quality and quantity of data in the input dataset is also an added concern to be taken into account. The more varied and quality the input set for the ML models, the better the results achieved.

Bridging with what was mentioned earlier, the quantity and quality of data that will be available for the different classes, that is, for the different types of identification documents, are metrics to take into account, i.e., one of the scopes of the project, and so, said that, it will be necessary to find in the process solutions to solve the problems that might arise.

Keeping in mind that we are dealing with sensitive data, it is difficult to get a good number of input samples for what is necessary to build a robust and high accuracy ML model. Furthermore, **the primary goal of extracting information from the different types of identification, and consequently complementing the data inside CRM, will not be achievable.** Because, in quantifiable terms, there is a discrepancy between the number of samples available for the different classes. There are many more scanned documents with CC than with the other types, which in percentage terms, represent about 80% of the total sample are samples of scanned documents with citizen cards. So said that, **the focus on extracting and collecting useful information from IDs will only be study and implemented for the CC class.**

Can immediately raise the problem of class imbalance. The construction of an ML model could result in metrics such as precision and recall getting relatively low values. Which, in practical terms, would mean an increase in the error of detection and recognition of the different types of identification, which would imply incorrect extractions and, consequently, a wrong complement of information.

Despite the limitation, it is still possible, through various techniques discussed below, to circumvent this limitation and, therefore, it will be possible to understand, in a perspective of future work, if it is indeed fruitful to expand the scope of information extraction to other types of personal identification. From a business perspective, extracting information from expired documents is useful whenever there is no information from the corresponding document in question.

Another challenge in this project is the inherent quality of these documents stored in MEO's CRM database. Many of the contracts, letters, or other types of typification inherent in these FileNet files are, from an image quality perspective, very noisy. In addition, numerous input files are corrupted, which adds to the difficulty in automating the information extraction system.

Finally, putting the focus on the last piece of the puzzle, the extraction and recognition of the different characters will be possible with the help of an OCR engine. Taking into account that in theoretical terms there is no optimal OCR engine, there are several possible solutions to be tested for the mentioned task. The use of an open-source OCR engine adds some limitations considering the goals of this project.

It is difficult to control the output of an OCR engine that is intended to be multifaceted, that is, to deal with different scenarios, and not specifically in the scenarios described in figure 3.2. That is, even if you can reconcile a good detection network, the second part, recognition, is always going to be prone to reading errors. A way around this obstacle would be to create a new model through fine-tuning with a detection backbone already trained, and a new recognition network with the dataset formed by a considered number of samples from the various documents made available for the development of this project.

In summary, we can subdivide this project into three phases, each with its particularities and obstacles. As mentioned before, the first phase is the detection and recognition of citizen cards and other types of identification documents. A second phase, where a pipeline of morphological transformations and operations on the input image would be dedicated to trying to improve the quality of the document identified in the previous phase. Finally, the last phase is dedicated exclusively to the detection and recognition of the different biographical fields present in the input documents.

3.4 Proposed Pipeline Architecture

As we could see and analyze in the current chapter, the IDs, in general, has not only textual information, but also graphical information, within a specific layout [95]. In addition to this, some difficulties arise at the time of performing the classification of the documents:

- The ID can be in different positions within all document, which consequently means that we will have a much larger area to cover when performing OCR processing for information extraction;
- The orientation of the ID in the document is not known a priori, and this situation could be catastrophic for the OCR results and the quality of the information retrieved;
- Also, the OCR task can be challenging due to the noisy background in the document and other distracting elements, that could confuse the OCR reading;
- Another problem is the color channels present in the digitized document. Sometimes we will have some documents that use the 3 channels-Red, Green and Blue (RGB)- and ,sometimes, we will have digitized documents using only 1 channel, which means that the document is in the monochromatic form (grayscale);
- And, we have to consider that, it is helpful for the last task in the pipeline, where we need to retrieve the information extracted in the OCR engine, to know which type of ID we are working with, because it will improve the way we treat the detected information and, most importantly, to know which information field we are dealing with.

Taking those points into account, and having in mind that traditional OCR is based in template matching [88], and of course, mostly suitable for scenarios with low complexity, we urgently need something that will support the OCR task ahead, before any Information Retrieval (IR). Similarly to what has been seen in most solutions in the document classification world, the best approach, and taking also into account the fact that it is preferable to rely on graphical elements then textual [72, 95, 104, 108, 116], is to consider and treat the first task of this project as an Image Classification Problem.

Therefore, what is proposed within this project and based on what was presented so far, is an automated pipeline adapted to the needs of the image classification problem that we have in hand.

A pipeline in software and computational terms is a collection of computational processes or programs (elements) that are connected in series, and the output of one element is the input of the next one [101].

By establishing the previous theoretical foundations as the main points to solve this problem, we can consider that all of the needed functions and operations over the system are going to be used in different stages of the final system. We can assume that this system is a pipeline of multiple stages and operations of detection, recognition, and extraction, that will implement different technologies over the pipeline for different and specific goals.

Once the development engine has been found, the aims and technology for this pipeline must now be understood.

To achieve the fixed goals, the final system must be able to identify among the various types of documents and different scenarios, those that are personal IDs, using for that, CV tasks, such as image annotation, data augmentation, classification/regression models, which of course are strongly connected with machine learning-based approaches.

These technologies and procedures are going to be covered and analyzed in the rest of this document as being Phase 0 (figure 3.5). Inside this Phase we will cover other sub-phases that are also important for auxiliary matter:

- Phase 0.1: Automatic detection of the format of the document(.PDF, .TIFF, .PNG, .JPG, .ZIP, etc.).
- Phase 0.2: ID (Object) with machine learning-based approaches or deep learning-based approaches [115].

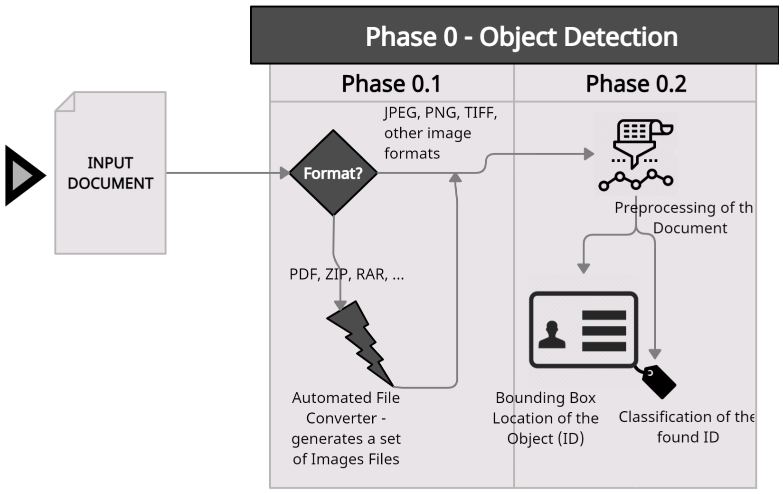

Phase 0 - Object Detection
Phase 0.1
Phase 0.2
INPUT DOCUMENT
Format?
JPEG, PNG, TIFF, other image formats
PDF, ZIP, RAR, ...
Automated File Converter - generates a set of Images Files
Preprocessing of th Document
Bounding Box Location of the Object (ID)
Classification of the found ID

Figure 3.5: Phase 0 Architecture Proposal

If, after the initial ML stage, we identify the existence of an ID, we can advance in the pipeline system and try to extract the many features and information possible from that ID, but always taking into account the need for image processing of the area labeled as ID area in the total area of the initial document (also in Phase 0.2).

The Phase 1 (figure 3.6) is considered the most important because it is where most of the OCR extraction errors occur. With a good application of good functions and operations in the preprocessing of the ID, the results can improve significantly taking into account the fact that most of the input have blur and noise associated, and a good image processing phase could remove some of the major issues related to the recognition and detection of characters in the ID. Because of the image quality variance that can be found in the input files, we call this phase, Phase 1, as Adaptive Image Processing Phase. Adaptive phase because it implies the application of Image Quality Assessment (IQA) algorithms to classify each ID or part of it taking into account its quality, on the desired scale. In this way, knowing in advance the classification in terms of the quality of the input ID, it is possible to adapt and apply the necessary morphological functions or operations, depending on the case at hand. Consequently, this will facilitate the way documents are processed and increase, theoretically speaking, the performance of the final software.

For each type of document identified and processed, the last goal is to maximize data collection, measuring its quality in terms of performance and feasibility of a scalable and parallel implementation. The implemented processes should be accomplished in this stage using OCR Engine (figure 3.7) as its main technology (Phase 2).

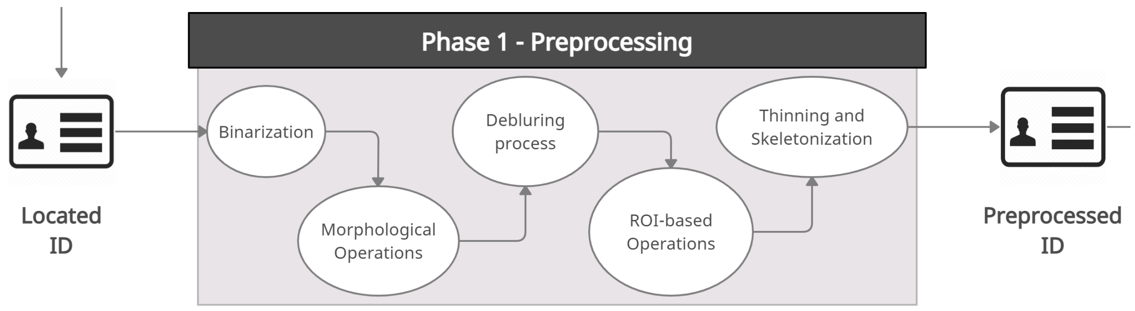


Figure 3.6: Phase 1 Overview

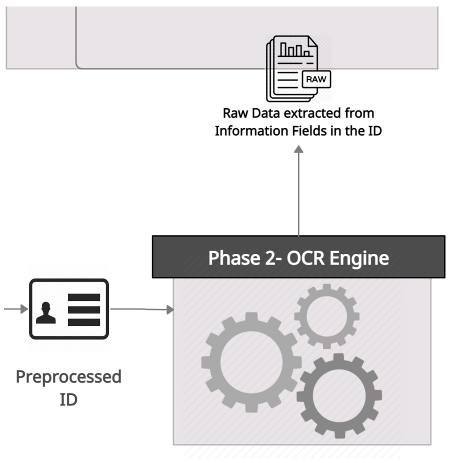


Figure 3.7: Phase 2 Overview

Covering all the pipeline, the last task is the most important, Phase 3 (figure 3.8), where we take the information extracted from the previous Phase and with the help of logical functions, match the pairs of fields found on the ID parts, promoting a strong information post processing. Here, we have yet to implement some measurement tools, so that looking at confidence levels or other types of metrics can solve some conflicts that could occur, due to some bad information extraction.

Eventually, exiting the pipeline, we have one more Phase (Phase 4) where we make the right qualitative and quantitative evaluation of the system and take into account a set of metrics, statistical evaluation, and performance analysis to, in future work, improve and update the quality of the system output and selection criteria. In figure 3.9, it is possible to check the proposed pipeline model for this project.

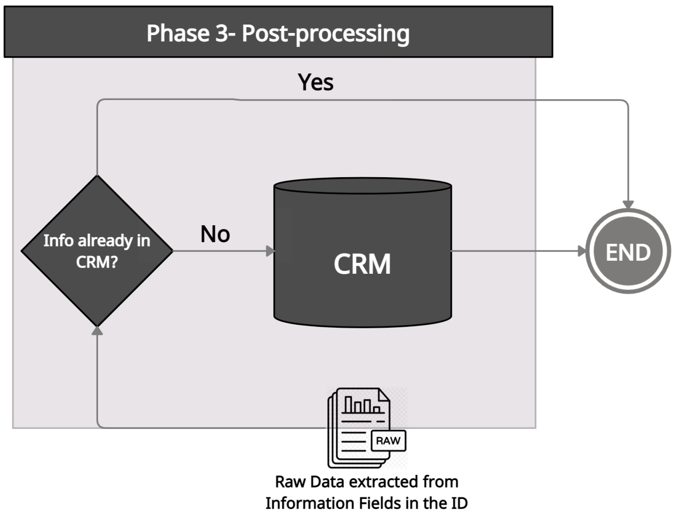


Figure 3.8: Phase 3 Overview

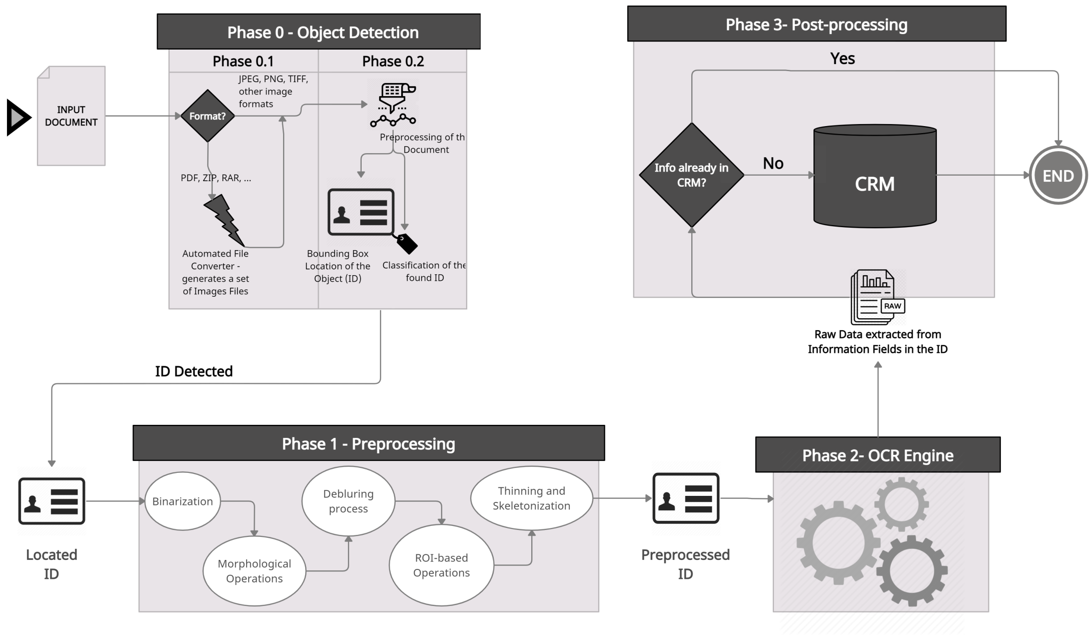

INPUT DOCUMENT
Phase 0 - Object Detection
Phase 0.1
Phase 0.2
Format?
JPEG, PNG, TIFF, other image formats
PDF, ZIP, RAR, ...
Automated File Converter - generates a set of Images Files
Preprocessing of th Document
Bounding Box Location of the Object (ID)
Classification of the found ID
ID Detected
Located ID
Phase 1 - Preprocessing
Binarization
Morphological Operations
Debluring process
ROI-based Operations
Thinning and Skeletonization
Preprocessed ID
Phase 2- OCR Engine
RAW
Raw Data extracted from Information Fields in the ID
Phase 3- Post-processing
Yes
Info already in CRM?
No
CRM
END

Figure 3.9: Complete Proposed Pipeline for the Project Architecture

This page is intentionally left blank.

Chapter 4

Phase 0 - Object Detection

Object detection is a Computer Vision (CV) approach for identifying and locating objects in images and videos. Used to count items in a scene, determine and monitor their precise positions, and properly label them. Also, in parallel, draw bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene.

Object detection is commonly confused with image recognition. Image recognition assigns a label to an image. Meanwhile, object detection draws a box around each element in a scene and labels it automatically. The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition.

As seen in chapter 2, object detection is broken down into Machine Learning (ML)-based approaches and Deep Learning (DL)-based approaches. CV algorithms are employed in ML-based systems to detect groupings of pixels that may belong to an object by looking at various characteristics of a picture, such as the color histogram or edges. These characteristics are then put into a regression model that predicts the object's position as well as its label [26].

DL-based methods, on the other hand, use Convolutional Neural Networks (CNNs) to conduct end-to-end, unsupervised object detection, which eliminates the need to specify and extract features individually. For the sake of this research, we will be focusing on DL methods because they have become state-of-the-art approaches to object detection [26, 115]

State-of-the-art object detectors are divided into two categories. On the one hand, two-stage detectors like Faster Region Based Convolutional Neural Network (R-CNN) [87] or Mask R-CNN [40] use a Region Proposal Network (RPN) to generate regions of interest in the first stage and then send the region proposals down the pipeline for object classification and bounding-box regression [98].

Single-stage detectors, on the other hand, such as You Only Look Once (YOLO) [85, 86] and Summed Square Difference (SSD) [60], handle object identification as a straightforward regression issue by taking an input image and learning the class probabilities [98].

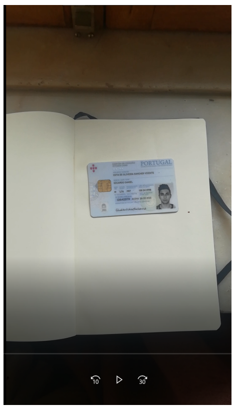
One-stage detectors have high inference speeds and two-stage detectors have high localization and recognition accuracy, but they are also the slowest [61].

In the context of this project, this phase will be the unique tool for extracting the different Identity Documents (IDs) that can be found in FileNets. Through their location, isolation in the context of the inserted scenario and their subsequent classification, i.e. label assignment.

This chapter will survey the methodologies used to train object detection models to aid in phase 0 of the proposed pipeline. Initially, we discuss the exploratory tests made in the first months of the project. The methodology behind the training of the different types of methods, one-stage methods and two-stage methods, is then presented, as well as the techniques for evaluating the models created. It is relevant to emphasize that the choice of the most fittable model is of great importance since this phase, being initial and primary, will influence all the output values of the remaining pipeline phases, hence it is relevant to perform all kinds of parametric evaluation to the training performed with the chosen dataset and the metrics used to evaluate their performance and accuracy in a digitized information processing environment.

4.1 Exploratory Work

In one of the first experiences carried out in this Phase - which was run with the help of MatLab language and its knowledge libraries Computer Vision ToolBox and Deep Learning ToolBox [67] - interesting results were obtained and with these, some guide points were formulated for the future implementation of the technology inherent to this phase. In this experiment, two videos of one minute each were filmed, with the Cartão de Cidadão (CC) of figure 4.1, to be the object focused on the experiment. In one of the videos, the front part of the card is filmed from various perspectives and angles (figure 4.1a), and in the other video, the back part of the same card (figure 4.1b). Note that both videos were filmed in the same light exposure conditions, and at various angles and distances, so that it is possible to simulate the conditions that exist within Customer Relationship Management (CRM) documents.


(a) Footage of the Front Side of the Test CC

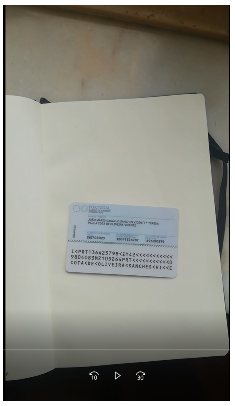
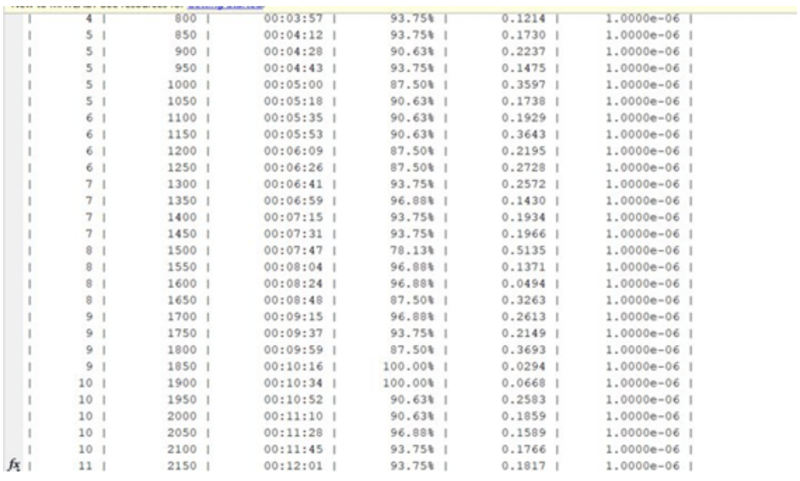

(b) Footage of the Back Side of the Test CC

Figure 4.1: Two videos of 1 minute each were filmed to create the Training Dataset

After the filming and compilation of the videos, 75 frames of both videos were pre-selected, which in short, resulted in the elaboration of a dataset of 150 images. As already studied, in the task of object detection and recognition it is necessary to provide the neuronal network with the Bounding Boxs (BBs) from where it will be expected to have an ID. For this, not only the training image directory was registered in the dataset, but also the location of the ID within the image. Since this is footage, and consecutive frames have similar positions, the delimitation of the BBs was forced, that is, a position where a set of 25 frames have the same positions were taken, to facilitate the process of creating the BBs.

Once the dataset assembly phase was concluded, the 150 images were trained to create what would be a model for the detection and recognition of IDs, using for that the construction of a Faster R-CNN. Similar to what happens with most of the technologies for object detection, the knowledge transfer from the AlexNet [1] network was used in this experiment [56].

When we say knowledge transfer, we are talking about a ML technique that focuses on the storage and application of information acquired while resolving one particular issue, which is consolidated by mirroring the extraction power of pre-trained CNNs and their optimal hidden layers (convolutional layers). Applying the "knowledge" and technology used by AlexNet in its hidden layers, we can obtain very interesting results close to the optimum. Regarding the choice of hyperparameters, these were chosen according to the default values recommended by the documentation provided by MatLab for Computer Vision ToolBox, except for the number of epochs that were defined as 15.



```
|      4 |      800 |     00:03:57 |      93.75% |      0.1214 |      1.0000e-06 |
|      5 |      850 |     00:04:12 |      93.75% |      0.1730 |      1.0000e-06 |
|      5 |      900 |     00:04:28 |      90.63% |      0.2237 |      1.0000e-06 |
|      5 |      950 |     00:04:43 |      93.75% |      0.1475 |      1.0000e-06 |
|      5 |     1000 |     00:05:00 |      87.50% |      0.3597 |      1.0000e-06 |
|      5 |     1050 |     00:05:18 |      90.63% |      0.1738 |      1.0000e-06 |
|      6 |     1100 |     00:05:35 |      90.63% |      0.1929 |      1.0000e-06 |
|      6 |     1150 |     00:05:53 |      90.63% |      0.3643 |      1.0000e-06 |
|      6 |     1200 |     00:06:09 |      87.50% |      0.2195 |      1.0000e-06 |
|      6 |     1250 |     00:06:26 |      87.50% |      0.2728 |      1.0000e-06 |
|      7 |     1300 |     00:06:41 |      93.75% |      0.2572 |      1.0000e-06 |
|      7 |     1350 |     00:06:59 |      96.88% |      0.1430 |      1.0000e-06 |
|      7 |     1400 |     00:07:15 |      93.75% |      0.1934 |      1.0000e-06 |
|      7 |     1450 |     00:07:31 |      93.75% |      0.1966 |      1.0000e-06 |
|      8 |     1500 |     00:07:47 |      78.13% |      0.5135 |      1.0000e-06 |
|      8 |     1550 |     00:08:04 |      96.88% |      0.1371 |      1.0000e-06 |
|      8 |     1600 |     00:08:24 |      96.88% |      0.0494 |      1.0000e-06 |
|      8 |     1650 |     00:08:48 |      87.50% |      0.3263 |      1.0000e-06 |
|      9 |     1700 |     00:09:15 |      96.88% |      0.2613 |      1.0000e-06 |
|      9 |     1750 |     00:09:37 |      93.75% |      0.2149 |      1.0000e-06 |
|      9 |     1800 |     00:09:59 |      87.50% |      0.3693 |      1.0000e-06 |
|      9 |     1850 |     00:10:16 |     100.00% |      0.0294 |      1.0000e-06 |
|     10 |     1900 |     00:10:34 |     100.00% |      0.0668 |      1.0000e-06 |
|     10 |     1950 |     00:10:52 |      90.63% |      0.2583 |      1.0000e-06 |
|     10 |     2000 |     00:11:10 |      90.63% |      0.1859 |      1.0000e-06 |
|     10 |     2050 |     00:11:28 |      96.88% |      0.1589 |      1.0000e-06 |
|     10 |     2100 |     00:11:45 |      93.75% |      0.1766 |      1.0000e-06 |
fx |     11 |     2150 |     00:12:01 |      93.75% |      0.1817 |      1.0000e-06 |
```

Figure 4.2: R-CNN training process, where it is possible to follow the Accuracy epoch after epoch

In figure 4.2, it is possible to follow the training that took place after this initial dataset, where we can see that the accuracy percentage (column 4) reaches 100% in many epochs, although this learning is not constant and stable until the end of the training. Which may indicate overfitting. Something that was expected considering the static nature and lack of invariance of the images used for the formulation of the used dataset. In addition, it should be noted that the choice of hyperparameters also has a decisive impact on the performance and occurrence of overfitting of the model created.

In figures 4.3 and 4.4, it is possible to see some tests performed with the previously created network.

In figure 4.3, we have a dataset document provided by Altice/MEO, which, in terms of the model created, represents a novelty for the features and type of scenario it had to process during the training phase. Despite the novelty and complexity imposed in the document, the model was able to mirror the knowledge taken from the static training dataset, managing to classify and locate with minimum error, the cards in the document.

In figure 4.4, the model finds as input a well-know scenario, but of the ID it dealt with during the training phase, which in theoretical terms should have resulted in a perfect

[1]AlexNet [56] is a CNN created in 2012 by Alex Krizhevsky, which obtained a top-5 error of 15.3% in the annual ImageNet Large Scale Visual Recognition Challenge, and today is used for the basic training of CNN design in various industrial applications.

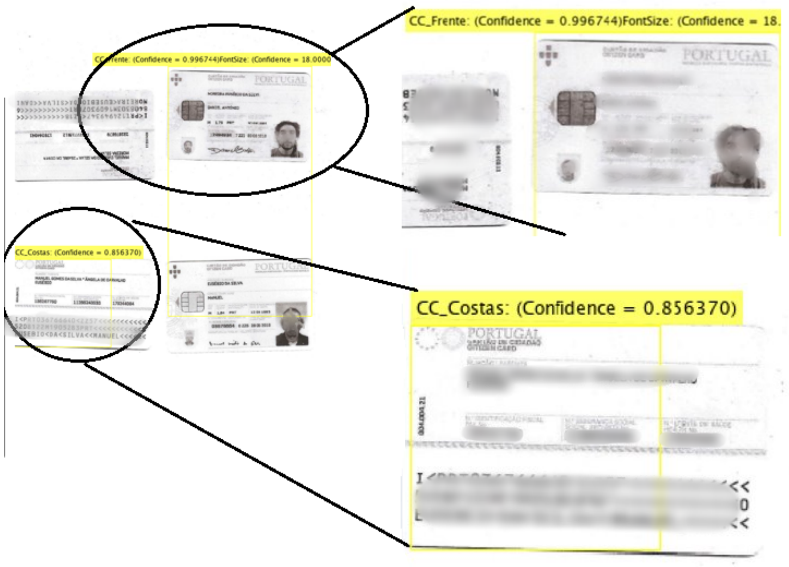

CC_Frente: (Confidence = 0.996744)FontSize: (Confidence = 18.
CC_Costas: (Confidence = 0.856370)

Figure 4.3: Test Document from CRM [citar]

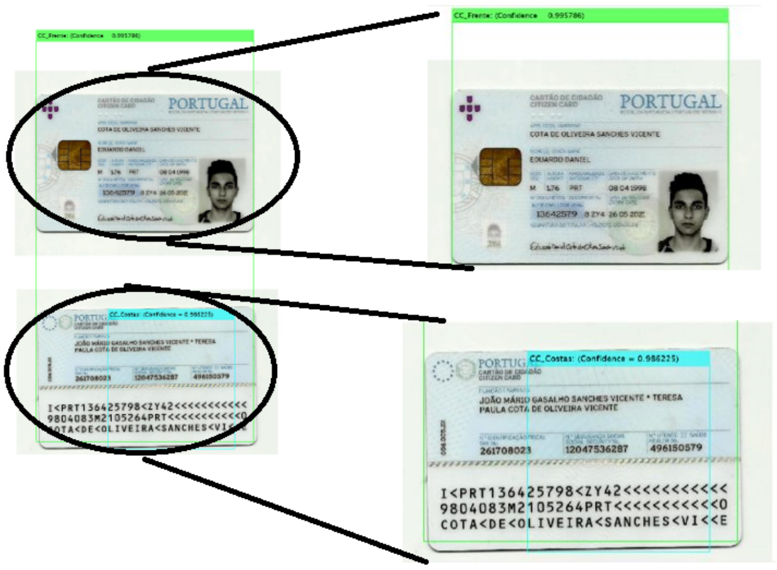

CC_Frente: (Confidence = 0.995786)
CC_Costas: (Confidence = 0.986225)

Figure 4.4: Digitized CC, the same used in the videos that originated the dataset of training

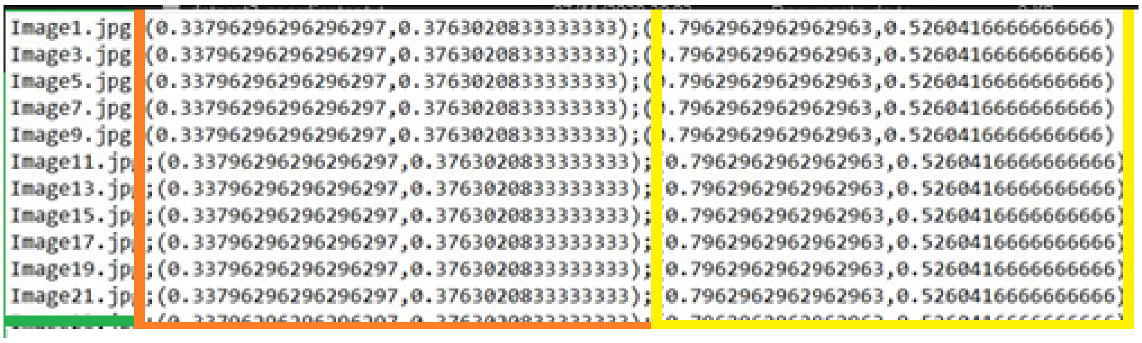
combination, not only of its location but also of its classification.

It is noticeable, in both results, that the model is not yet sensitive to the point of creating with precision the BBs for the location of the detected object. Although, in both results it has correctly identified the occurrence of the front of the card and the back, it has not yet been able to explicitly make a good detection and recognize which are the defined limits for both detected objects.

Besides this, figure 4.3 represents a level of complexity a little above the one imposed in the training phase, because the model did not learn how to classify more than one ID per document, and there is no case in which both parts of the card appear.

To improve the last model, some changes were applied on the dataset. The dataset initially composed of 150 images, 75 for each side of the card, was enriched with 350 new images taken from the same videos previously described. This resulted in the creation of a training dataset of 500 images, 250 from each class.

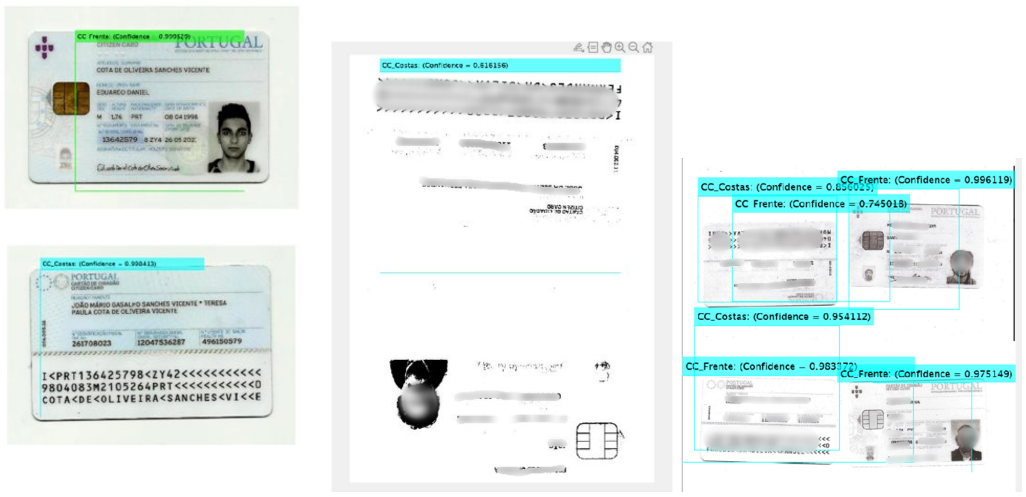
Putting everything together, the final dataset looks as shown in figure 4.5. In figure 4.5 we can see that each line represents a training image, where the first parameter is the name of the image in the directory where the dataset is (green), followed by the coordinates of the center of the BB (orange) and its size (length, height) in yellow.



```
Image1.jpg (0.33796296296296297,0.3763020833333333);(.7962962962962963,0.5260416666666666)
Image3.jpg (0.33796296296296297,0.3763020833333333);(.7962962962962963,0.5260416666666666)
Image5.jpg (0.33796296296296297,0.3763020833333333);(.7962962962962963,0.5260416666666666)
Image7.jpg (0.33796296296296297,0.3763020833333333);(.7962962962962963,0.5260416666666666)
Image9.jpg (0.33796296296296297,0.3763020833333333);(.7962962962962963,0.5260416666666666)
Image11.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
Image13.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
Image15.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
Image17.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
Image19.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
Image21.jp;(0.33796296296296297,0.3763020833333333); 0.7962962962962963,0.5260416666666666)
```

Figure 4.5: Dataset with 500 images from the recorded videos

Keeping the same training conditions as before, some results were much more positive compared to the previous results (figure 4.6).


CC Frente: (Confidence = 0.999529)
CC_Costas: (Confidence = 0.998413)
CC_Costas: (Confidence = 0.816156)
CC_Costas: (Confidence = 0.8...)
CC_Frente: (Confidence = 0.996119)
CC Frente: (Confidence = 0.745018)
CC_Costas: (Confidence = 0.954112)
CC_Frente: (Confidence = 0.983372)
CC_Frente: (Confidence = 0.975149)

(a) (b) (c)

Figure 4.6: Tests on the R-CNN Object Detector Trained with a 500 images Dataset

From the analysis made to the results, it is possible to verify that, for example, in the figure where the ID used for the training appears (figure 4.6a) the BBs generated from the parts of the ID in the scanned document are much closer to the correct coordinates than in previous experiments.

Furthermore, in tests performed with documents provided by Altice/MEO, a slight improvement is verified regarding the detection and classification of the IDs in the input documents (figures 4.6b and 4.6c).

It is also important to highlight the progress made in what is visible in the figure 4.6b. In which, although the front of the card was not detected and classified, the model was able to interpret the orientation of the document and, despite this difficulty, classify and detect the back of the card in the document.

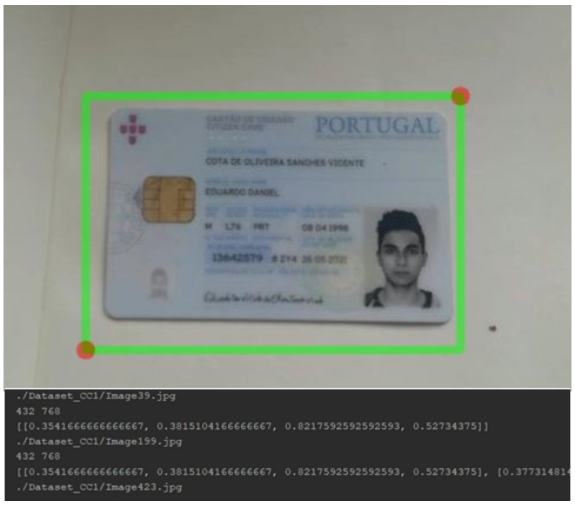
After the experiences and results achieved with MatLab, this knowledge was transported to other technologies already existing in the CV world. As is the case of the APIs made available by Google Cloud Platform (GCP) [46].

The GCP [46], is a cloud computing service offered by Google. Besides offering a wide range of solutions and services for data storage and Big Data analysis, GCP also offers ML solutions. Within these, it is important to highlight Google's progressive bet on CV related solutions. Taking into account the fact that GCP operates its services in the cloud, i.e., keeps the data for a shared service among the various users of the same service, in experimental terms, we are limited to using a dataset with local information.

That said, the experiences that were made with GCP's solutions did not take into account the input documents provided by Altice/MEO due to what is stipulated and protected by General Data Protection Regulation (GDPR).

Considering the mentioned limitations, an industrial survey of the applicability of the solutions made available by GCP was only done, to find out to what extent it is useful for the customer to use this tool in this project. In other words, in addition to experiments made in the field of object detection, a market survey was made (prices and performance of the services provided) to analyze the pros and cons of the solution. The GCP platform offers a ML API Rest solution, called GCP's Cloud AutoML API [46].

A script was created to initially automate the creation and delimitation of the BBs of the objects in the images. This is because the GCP AutoML tool needs specific formatting in the Comma-separated values (CSV) format of the coordinates and directories of the images that are used in the dataset. The script created (figure 4.7) in Python, with the help of the Open Source Computer Vision Library (OpenCV) library, not only allowed the design and delimitation of the BBs of the objects in all images but also the automatic writing to a table of the coordinates associated with the image, so that within GCP it was only necessary to upload the images in a bucket of google cloud storage and a CSV file to link with their respectively BBs.



```
./Dataset_CC1/Image39.jpg
432 768
[[0.3541666666666667, 0.3815104166666667, 0.8217592592592593, 0.52734375]]
./Dataset_CC1/Image199.jpg
432 768
[[0.3541666666666667, 0.3815104166666667, 0.8217592592592593, 0.52734375], [0.37731481
./Dataset_CC1/Image423.jpg
```

Figure 4.7: OpenCV script to draw BBs automatically

Internally, the AutoML tool uses a series of cloud-trained networks, which as they make classifications and feature extraction in other models, will increase their learning capacity for choosing the best hyperparameters and architecture taking into account the nature of the dataset. That is, they end up being able to create, for each specific classification case, the best approach and the best architectural choice taking into account the problem and nature of the dataset at hand, thus replacing the need for a preparatory analysis of the best approach for the assembly and construction of the ML solution. In practical terms, it ends up facilitating all the preparation work that is inherent to the construction of an ML model for the problem, because this service does it for us.

As it was done in the above-mentioned experience in the MatLab environment, 500 images were loaded, 250 of each part of the card. From the 250 images of each part, 200 were separated for training, 25 for validation, and another 25 for testing.

The training in this service is done in the Cloud, the expected time for the service running on a US server is at least 9 hours. Here the negative aspects of the solution to be tested begin to emerge. Logistically it becomes impractical to maintain such a solution with so much sensitive data in the cloud, that moreover, it ends up not being as fast as the customer needs it to be unless Altice/MEO consider paying according to their logistical needs.

After the training, we are presented with the model behavior in a Receiver Operating Characteristic (ROC) curve (figure 4.8) in terms of accuracy and recall. As expected, and from what had already been confirmed in the model implemented in the MatLab environment, there was clear overfitting of the model. Despite this, a recall value of about 85% and 100% accuracy was obtained.

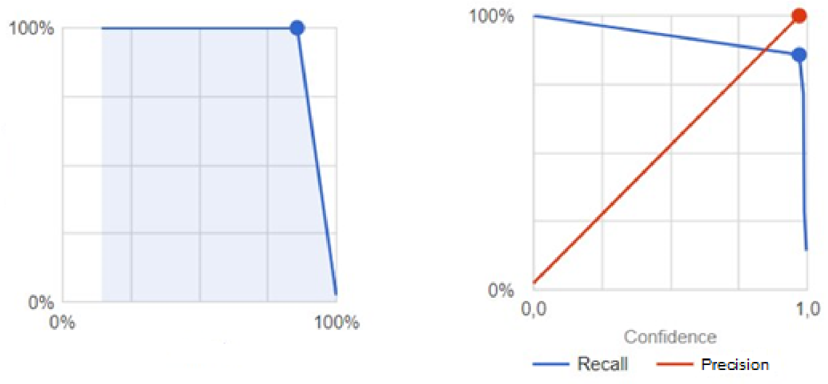


Figure 4.8: ROC Curve for the Model Created with the AutoML Object Detector

Similar to what was made in the model previously created in Matlab, several tests to this object detector were made. In the figures 4.9, 4.10, 4.11 and 4.12 it is possible to verify some of the results obtained in tests performed with AutoML in the cloud.

In general, in the documents where the front of the card appeared, they were correctly detected and delimited. Although the same can not be said for the occurrences to the back of the card, wherein none of the cases has accomplished any detection or consequent classification. Considering that the model was trained to deal only with situations in which only an occurrence of one of the parts of the card occurs, hence the justification that in figures 4.9 and 4.12 the detection and classification. Regarding what happened in figure 4.11, where only the first part of the card is detected and correctly classified, it may be related to the fact that the model is not trained and used to classify two objects at once in the same image, because in the training dataset this never occurred. Finally, figure 4.10, is the case where no classification or detection was carried out. Despite the similarity with the scenario in figure 4.11, in this case, we are dealing with a scenario where besides the

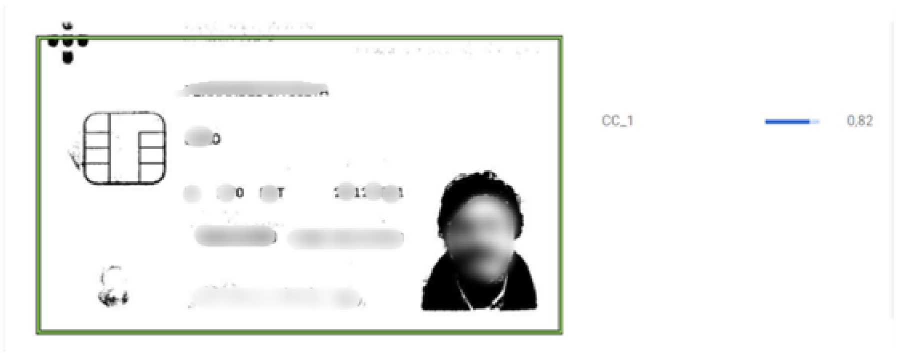


Figure 4.9: Test 1 made with the Object Detector created with AutoML

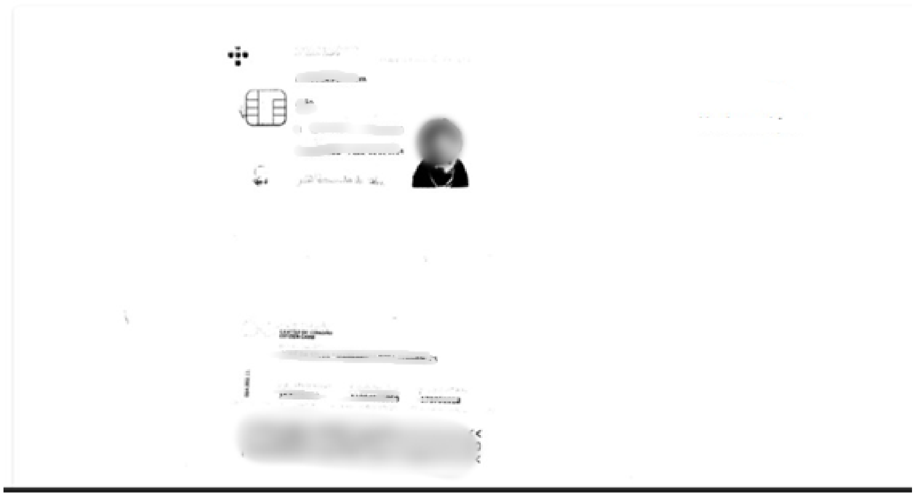


Figure 4.10: Test 2 made with the Object Detector created with AutoML

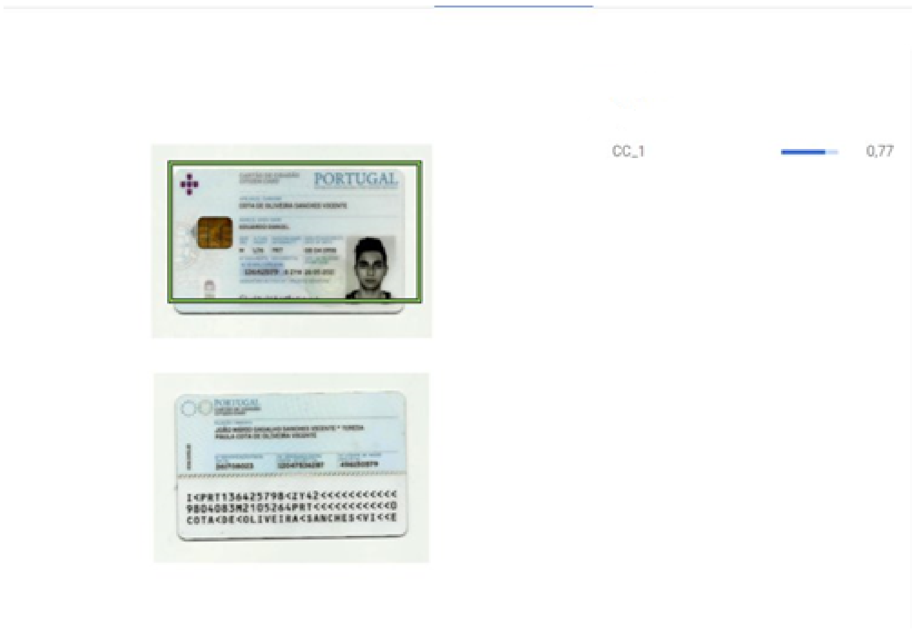


Figure 4.11: Test 3 made with the Object Detector created with AutoML

color palette in the document is different from the one used in the entire dataset, it is evident that we are dealing with a case that presents greater complexity since the features

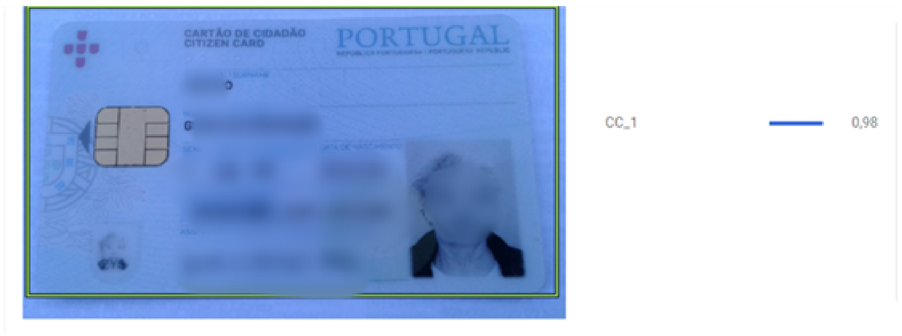


Figure 4.12: Test 4 made with the Object Detector created with AutoML

that will be detected for this case will be much more scarce and of another nature than those that were used to train the model.

Overall, the experience followed with the GCP AutoML service was a success, and with this, it was possible to draw several conclusions about the future of the models to be implemented, as well as the logistical survey of costs and operationality of this solution from an industrial perspective for Altice/MEO.

All the experiments mentioned so far have been done based on the dataset originated from the two films made to a single CC, to assess how well the models using knowledge transfer as a basis behave in classifying terms. Also, these experiments were intended to verify whether just training a model with various images of just one card as a "template" would be effective for learning the features necessary for the detection of objects (IDs) in the input documents provided by Altice/MEO. Based on the conclusions drawn from the experiments performed so far, we can conclude that no, it is not effective.

Returning to the MatLab environment, the last test was done this time to try to prove one of the conclusions drawn from the experiences already mentioned, which is that the limitation of scenarios within the dataset is a critical factor for good classification and detection of objects in the model created.

For this, it was finally added to the training dataset another 300 scanned documents provided by Altice/MEO. These entered documents consisted only of the pages of the documents in JPEG format where it was possible to locate and classify an ID in the scanned document. As with the other images already present in the dataset, the design and capture of the coordinates of the BBs in the documents introduced in the new dataset were done the same way, keeping all the training conditions previously used. This includes not only the values of the hyperparameters but also the architectural conditions of Faster R-CNN.

In the following figures 4.13, 4.14, 4.15, 4.16 and 4.17, it is possible to check some of the results obtained.

The figures from 4.13 to 4.17, are some of the optimal results achieved with this model (green color refers to the front side of the CC and the blue color to the back side of the CC).

As it is possible to verify, no matter how complex the scenario, how many IDs are present, and what is the orientation presented in the scanned document, it is important to highlight how well the model behaved in such testing stage. This result leads us to conclude that the diversification of scenarios in the dataset is of extreme importance for the classification and localization performance of the object detection model being trained, regardless of its

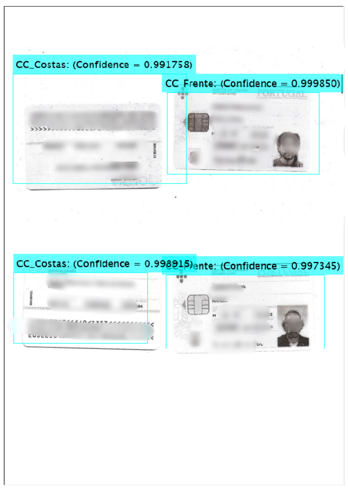

CC_Costas: (Confidence = 0.991758)
CC_Frente: (Confidence = 0.999850)
CC_Costas: (Confidence = 0.998915)
ente: (Confidence = 0.997345)

Figure 4.13: Test 1 made with the Faster R-CNN trained with an 800 images dataset

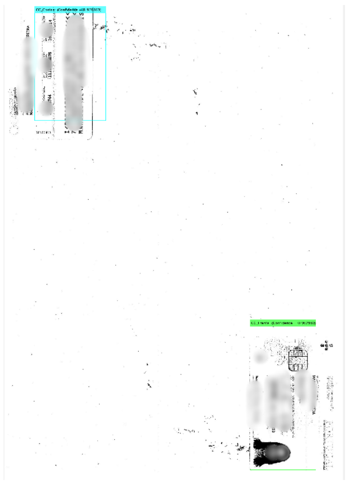


Figure 4.14: Test 2 made with the Faster R-CNN trained with an 800 images dataset

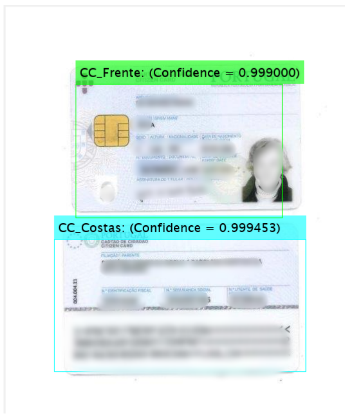

CC_Frente: (Confidence = 0.999000)
CC_Costas: (Confidence = 0.999453)

Figure 4.15: Test 3 made with the Faster R-CNN trained with an 800 images dataset

origin or architecture. After the introduction of more examples and scenarios provided by Altice/MEO, we were able to obtain very interesting and promising results that give hope to evolve in the construction of more models and increasingly specialized for the detection, not only of CCs but of other types of IDs.

Despite the good results, it is important to emphasize that some more negative aspects

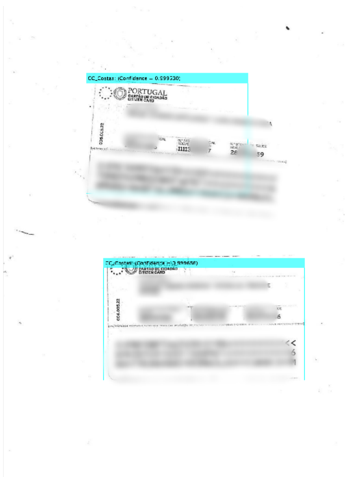


Figure 4.16: Test 4 made with the Faster R-CNN trained with an 800 images dataset

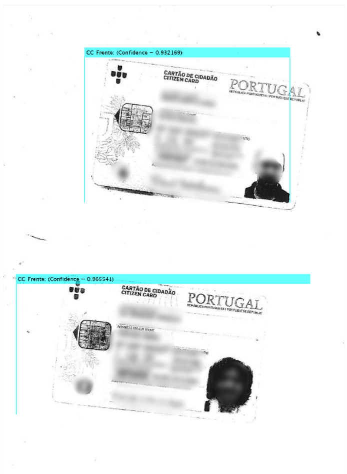


Figure 4.17: Test 5 made with the Faster R-CNN trained with an 800 images dataset

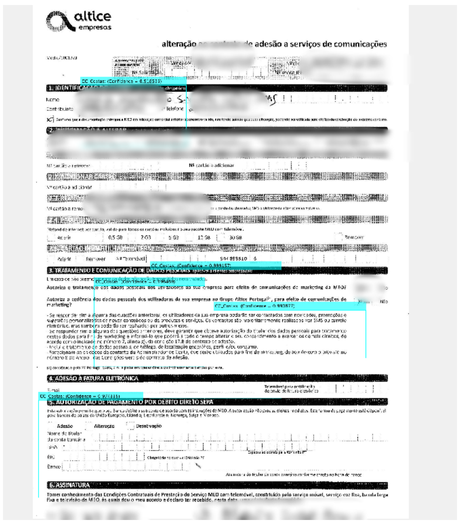
have also been recognized concerning the tests carried out with this model.

For instance, in the case presented in figure 4.18 it is visible a page commnonly found in the documents present in the CRM. This type of input is an example of a contractual page, usually in black and white with some fields identifying the customer and the service to which they are subscribing. However, in terms of the context of the problem, we are not interested in taking information from this type of scenario, so it is not favorable to make a classification evaluation and survey of the objects in this type of scenario because it is more than evident that there is no object (ID) of interest.

4.2 Data Annotation and Augmentation

The act of recognizing raw data (pictures, text files, videos, etc.) and adding one or more relevant and informative labels to give context so that a machine learning model may learn from it is known as data labeling. A number of use cases, such as computer vision, natural language processing, and speech recognition, need data labeling [26, 101]

Data annotation and labeling are two fundamental elements of machine learning that help machines recognize images, text and videos.When data is tagged, sophisticated algorithms



altice empresas

alteração [redacted] de adesão a serviços de comunicações

CC_Costas: (Confidence = 0.518583)

1. IDENTIFICAÇÃO [redacted]

Nome [redacted]

Contribuinte [redacted] Telefone [redacted]

2. [redacted]

Nº cartão a remover Nº cartão a adicionar

Plafond de internet por cartão, válido para todos os cartões incluídos no seu pacote MEO com telemóvel.

Aderir 0,5 GB 2 GB 5 GB 10 GB 30 GB Remover

Aderir Remover Nº Telemóvel 964 893510 6

CC_Costas: (Confidence = 0.999157)

3. TRATAMENTO E COMUNICAÇÃO DE DADOS PESSOAIS

CC_Costas: (Confidence = 0.996459)

Autoriza o tratamento dos dados pessoais dos utilizadores da sua empresa para efeito de comunicações de marketing da MEO?

Autoriza a cedência dos dados pessoais dos utilizadores da sua empresa ao Grupo Altice Portugal, para efeito de comunicações de marketing?

CC_Costas: (Confidence = 0.993877)

4. ADESÃO À FATURA ELETRÓNICA

CC_Costas: (Confidence = 0.[illegible])

5. AUTORIZAÇÃO DE PAGAMENTO POR DÉBITO DIRETO SEPA

6. ASSINATURA

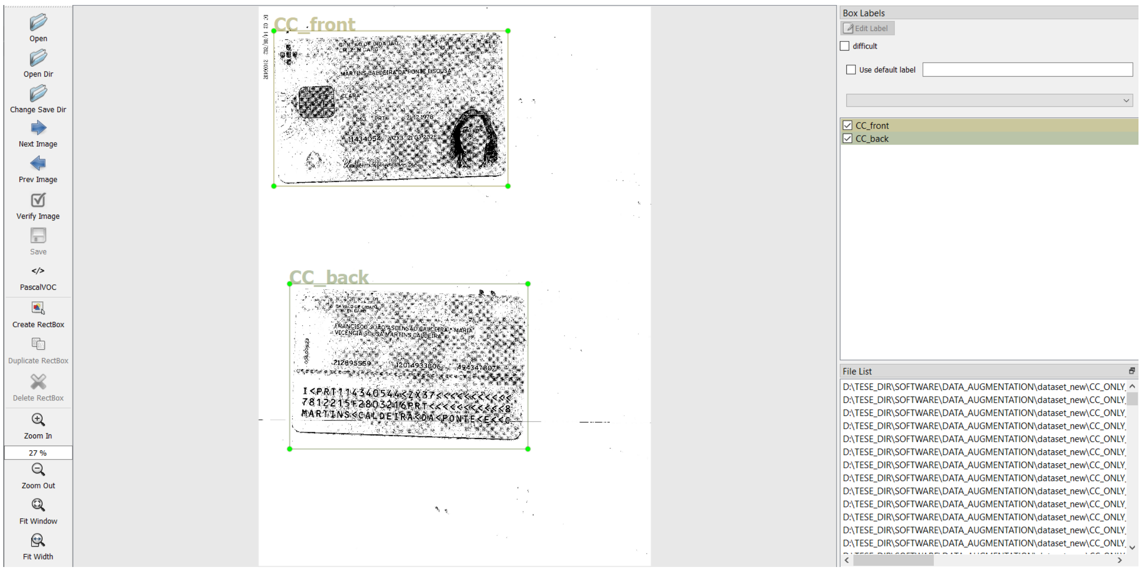
Figure 4.18: Bad Detection Example with the Faster R-CNN trained in MatLab

are trained to identify trends in the future. Labeling is the process of labeling or adding information to data in order to make it more relevant and useful for machines to comprehend and learn from.

As it was possible to see in the previous section, some efforts towards data labeling had already been made, namely the script that was written to automate the design of BBs to enrich the dataset used in an experimental phase in a GCP environment. Although the efforts made to automate this manual task have achieved fruit, in the presence of a larger sample of input data to feed a larger and more favorable dataset for DL model training this task becomes impractical and costly.

To assist in the task of mapping all available samples to feed the dataset for training the DL models, a tool called LabelImg [106] was used.

LabelImg is a graphical image annotation tool that allows us to label the BB of objects in pictures, in a more convenient way. It is written in Python and uses the PyQt5 module for its graphical interface (figure 4.19).



CC_front
CC_back

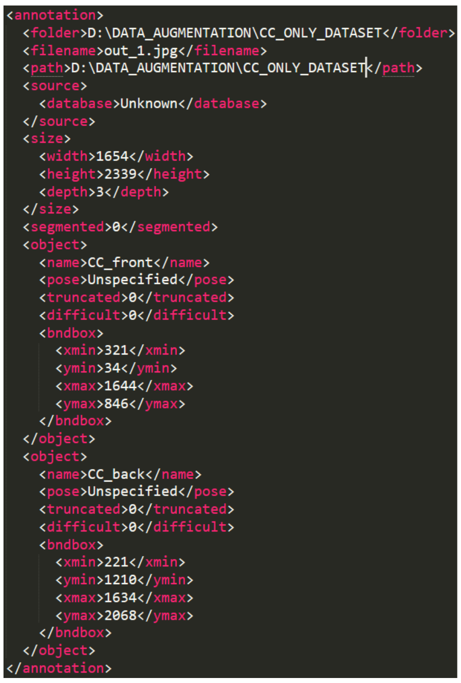
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_
D:\TESE_DIR\SOFTWARE\DATA_AUGMENTATION\dataset_new\CC_ONLY_

Figure 4.19: LabelImg Environment for multi annotation in the dataset for CC classe only

Pattern Analysis Statistical Modeling and Computational Learning (PASCAL) is a European Union-funded Network of Excellence. PASCAL hosted the Visual Object Challenge (VOC) from 2005 until 2012. PASCAL publishes object detection datasets and benchmarks every year [27].

In ImgLabel, annotations are saved as eXtensible Markup Language (XML) files in PASCAL VOC [27] format, the format used by ImageNet [24]. Besides, it also supports YOLO Darknet [83] and CreateML [46] formats.

Each picture includes an associated XML file defining the BBs contained in the frame, and PASCAL VOC annotations were published in an XML format. For example, in figure 4.20 a single XML annotation sample in the dataset for object detection is showed.



```
<annotation>
  <folder>D:\DATA_AUGMENTATION\CC_ONLY_DATASET</folder>
  <filename>out_1.jpg</filename>
  <path>D:\DATA_AUGMENTATION\CC_ONLY_DATASET</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1654</width>
    <height>2339</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>CC_front</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>321</xmin>
      <ymin>34</ymin>
      <xmax>1644</xmax>
      <ymax>846</ymax>
    </bndbox>
  </object>
  <object>
    <name>CC_back</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>221</xmin>
      <ymin>1210</ymin>
      <xmax>1634</xmax>
      <ymax>2068</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 4.20: Annotation in PASCAL VOC format (XML) in the dataset for CC only detection

On the other hand, the YOLO Darknet annotation is just a tet file that consists of a series of lines, where each one represents a detection or annotation of a detectable object in the target image. Each object or line in the file is represented by five values, where it respectively has the following values: $<class_{id}>$ $<x_{center}>$ $<y_{center}>$ $<width>$ $<height>$.

In terms of annotation, the LabelImg tool runs on the target directory where the images as input data for model training, and, through the help of some programmable hot-keys, it is possible to optimize the registration, not only of the BBs present but also, their respective categorization. After annotation, the program allows the user to choose whether the output format should be PASCAL VOC or YOLO, all easily and automatically. At the end of the annotation, the program registers all image/annotation pairs in the same input directory.

Although, when we talk about it, we are saying one way to battle the shortage of data, by artificially augmenting our dataset. The technique has proven to be so successful that it's become a staple of DL systems. As is the case with deep learning applications, the more data, the merrier.

Another approach to consider why data augmentation works is to imagine it as adding noise to our dataset. It is especially true in online data augmentation, which involves stochastically enhancing each data sample each time it is sent into the training loop.

Because of the stochastic data augmentation, each time the neural network views the same image, it is somehow different. This difference may be thought of as noise being injected into each data sample, forcing the neural network to learn generalized characteristics rather than overfitting on the dataset [94].

From the lessons learned from the exploratory experiments carried out at the beginning of this project, it is relevant to draw some strong points that will be fundamental to outline the way the formation and increase of the final dataset will be carried out:

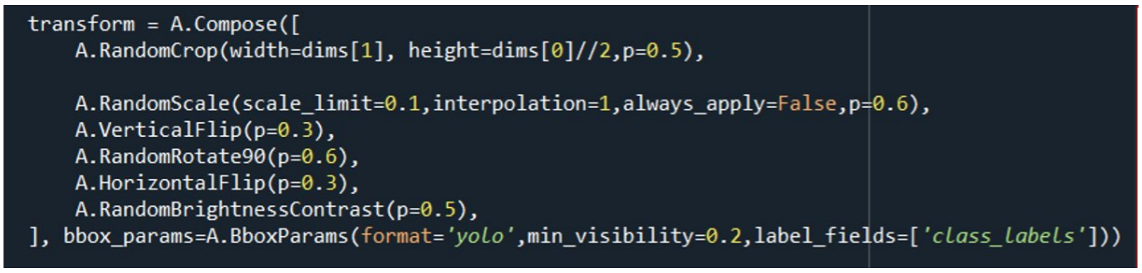
- In a first experiment it was possible to realize that with only one CC sample it is a bit complicated to get good results in terms of accuracy for the context of the problem, and on top of this poorly formulated detections and BBs;
- The low or non-existent ability to interpret the existence of more than one occurrence of ID per page or on different pages;
- Although the sample number of CCs was high, a second experiment, where there were about 1500 annotations, showed some problems such as a high register of False Positivess (FPs), which makes the process very slow, and in addition, some confusion in the separate classification of the class CC_{front} and CC_{back};
- Another problem evidenced within the negative registration of FP is related to the lack of negative registration elements and annotations in the dataset, that is, annotations that do not contain any ID, so that the model learns to distinguish target regions;
- From what was studied earlier, within the field of Object Detection, before forming and training a good object detection model, it is theoretically advised to have at least two thousand annotations per class [83, 86, 87].

In the final sample of FileNets provided by Altice/MEO, the available number of samples for each ID was collected and surveyed using the LabelImg tool (4.1).

As already mentioned in the previous chapter, due to the great imbalance existing in the different available classes in terms of sample number, it will not be possible yet to have a stable solution for a multiclass detection that contemplates all the types of IDs mentioned. **The focus will be on training and modeling both methods for the subclasses within the CC class**.

In addition to the large imbalance existing by the various classes, there is also the image quality and input component that has to be analyzed. As already discussed throughout this document, one of the main focuses in this project is the intelligent inference of strategies to adopt to circumvent extraction problems related to the image processing component. Thus, it is necessary to take into account, the set of operations chosen to perform data augmentation in operations that contemplate the "simulated" registration of the difficulties visible in many of the complicated scenarios. It is necessary at this stage to choose the right choice of the best parameters to use as a reference to create new and original strands of the original annotations made to the dataset. It is not only, for example, at the level of object orientation (ID), but also in terms of adding noise to the background or other morphological transformation available to try to recreate in the best possible way several of the challenges of the image.

Taking this into account, to increase the dataset, the project Albumentations [17] was applied, which is available in wrapper mode for the Python environment. With the Albumentations library it is possible to apply a series of transformations to the images present in the initial dataset. This way multiple scenarios with several different difficulties and anomalies can be created, this way, making the total dataset more robust to new and complicated inputs. Of the possible transformations applied, we can highlight the relevance of applying random 90-degree rotation, the application of high or low contrast (thus simulating the different light exposures that may be present in the FileNets), the use of a utility function to change the scale, and also reflections at the level of the two ordered axes. In the figure 4.21 you can see an example of a simple data augmentation pipeline using the Albumentations library [17]. The application of the six distinct operations in 4.21 is dependent on the probability of occurrence associated with them in their definition. Each receives as a mandatory argument the value of p (probability). In short, the pipeline takes each image from the dataset and creates new images with different changes to its composition, to produce new data, but always maintaining the integrity of the information taken from the BBs. That is, when the transformation occurs in the file, the transformed BBs must correspond in their entirety to the same area and coordinates once bounded by the detected and annotated object. For each class, the increase rate is defined according to the initial sample available.



```
transform = A.Compose([
    A.RandomCrop(width=dims[1], height=dims[0]//2,p=0.5),

    A.RandomScale(scale_limit=0.1,interpolation=1,always_apply=False,p=0.6),
    A.VerticalFlip(p=0.3),
    A.RandomRotate90(p=0.6),
    A.HorizontalFlip(p=0.3),
    A.RandomBrightnessContrast(p=0.5),
], bbox_params=A.BboxParams(format='yolo',min_visibility=0.2,label_fields=['class_labels']))
```

Figure 4.21: Example of a Image Data Augmentation Pipeline with the support of Albumentations

Looking once again at what was registered as the objectives and focus of this project, the initial sample of documents made available by Altice/MEO resulted in the formation of two final datasets. The first dataset is intended solely and exclusively for the training and creation of object detection models to detect only CCs and their parts. This dataset besides being composed of images annotated and augmented with IDs is also composed of approximately 38% of blank annotations, which will have the purpose of optimizing the model classification and reduce the number of false positives detected. The second and largest dataset is destined for the detection and identification of all the target IDs previously mentioned.

From the critical points mentioned above, it is relevant to note that for the training of a reliable object detector, it is necessary, in practical terms, to annotate at least two thousand or more images for a given class. Based on this principle, the formation of the second dataset will also have this principle for the different ones. Of the different types of IDs analyzed in the context of this project, three of them have two subclasses, that is, two parts that compose it and are detected separately. For these four classes (Bilhete de Identidade (BI), CC, and Autorização de Residência (AR)), in theoretical terms, we would expect to have at least four thousand samples (two thousand samples for each part composing the ID) in total for each class to comply with the principle.

As can be seen in table 4.1, the distribution of the different ID in the final dataset for multiclass detection is not uniform, because even if we use Data augmentation as a tech-

CLASS	**DATASET**			
	CC DATASET		**MULTICLASS DATASET**	
	Init	**New**	**Init**	**New**
CC	2270	1774	2270	1774
BI	0	0	31	2320
AR	0	0	23	1760
IF	0	0	5	800
BLANK*	0	2500	0	3500
TOTAL	6514		12543	

Table 4.1: Final distribuition of the number of categorized samples for each of the datasets that are going to be used.

nique to expand the number of varied samples in a dataset, it is impractical to try to balance classes, such as class Identificação Fiscal (IF) and class CC, where there is a huge discrepancy, and balancing the number of IF samples to meet the number of CC samples will most likely result in a case of overfitting for that class, something that is intended to be avoided. Despite this, the final IF sample number turned out to be well short of the recommended two thousand samples.

Similar to what happened to the dataset destined for the training of object detection for the CC class, the second dataset also ended up consisting of, approximately 30% of blank annotations. Overall, the number of CC samples is the sum of the ones available for the other classes, all added together(representing almost half of the sample universe).

Although the end sample values are too unpredictable and likely to settle on overfitted models for the multiclass dataset, it is crucial to understand to what degree it is possible to get decent results for ID extraction, even with a very small starting sample of data. Furthermore, it is critical to comprehend how the inclusion of this data augmentation and annotation method has improved phase 0 results compared to the results previously experienced in the preceding section, and how these modifications will aid in the near-perfect identification of IDs from the CC class.

4.3 Darknet

Darknet is a C and Compute Unified Device Architecture (CUDA)-based open-source neural network framework. It's quick to set up and supports both Central Processing Unit (CPU) and Graphics Processing Unit (GPU) processing [83]. Because its computational capacity is based on foundations written in C and management of processing power in a GPU environment with CUDA, this framework enables real-time training of DL models.

The Darknet framework not only allows the use of popular models such as ResNet and ResnNeXt [41] for image classification, but also allows the training of classifiers on the CIFAR-10 dataset [55], but is specialized and built for training one-stage models of the YOLO family.

The darknet project is an open-source framework that can be compiled on both Windows and Linux systems. During the training of phase 0, the darknet framework was used in a Linux environment compiled with CUDA and CUDA Deep Neural Network library (cuDNN) [75] embedded so that GPU power could be used to train the models. Furthermore, in the training environment, the cluster was composed of Tesla V100 graphics, so the parameter $CUDNN_HALF$ [75, 83] was used, which enables the use of color tensors, which in practical terms increase detection by two to three times. In addition, the YOLO models have in their configuration a flag to activate data augmentation throughout the training to decrease the loss value during training. This tool makes use of the dependencies of the OpenCV library [15]. It is also possible to associate CPU power by channeling put $OPENMP$, that is, accelerating YOLO by using multi-core CPU.

As already mentioned in this document, the YOLO family comprises an extensive number of versions and enhancements with diverging qualities from generation to generation .

Comparatively speaking, YOLOv4 is an improvement on the YOLOv3 algorithm by having an improvement in the Mean Average Precision (mAP) by as much as 10% and a number of Frames Per Second (FPS) by 12% (figure 4.22).

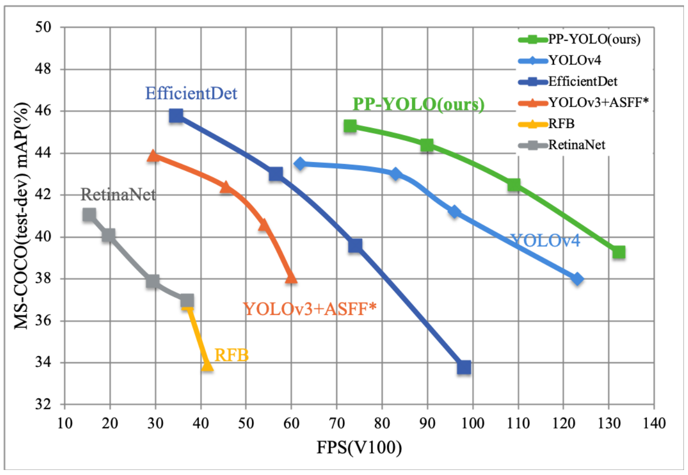


Figure 4.22: Comparison between the various state-of-the-art one-stage methods in terms of FPS in Microsoft Common Objects in COntext (MS COCO) , namely between the versions of YOLOv3 and YOLOv4

Still, within the YOLO family, there are versions whose architecture is lighter than the others of the same family, these are the so-called tiny versions, and two were built, a YOLOv3-tiny based on the architecture of YOLOv3 and a YOLOv4-tiny based also respectively by YOLOv4. Theoretically speaking, by observing the extent and complexity of the architectures of both tiny versions of the YOLOv3 and YOLOv4 models, one can see that they will have a much shorter inference time and closer to real-time than the root models that originated them. But, on the other hand, they will consequently sin in the accuracy and reliability of the extracted objects. It is necessary to study and train these branches of the family of one-stage methods to ascertain which model best fits the latest software, not only in terms of time but also in terms of confidence in the inference made by them.

4.3.1 Training Custom Object Detector

After building the datasets, as discussed in the previous section, it is time to train the models mentioned above. As mentioned, the primary dataset, which corresponds to the dataset destined solely and exclusively to CCs, will only have two subclasses in terms of training, CC_{front} , and CC_{back} (which corresponds to each part that composes the ID). While the other secondary dataset will have seven subclasses, because, as mentioned before, three of the classes of IDs comprise two subclasses, which correspond to the front and back of these three IDs.

In a Darknet environment, the dataset to be loaded should consist of images in Portable Graphics Format (PNG) or Joint Photographic Experts Group (JPEG) format along with their respective annotations in a text file format. Once the dataset is loaded, it is necessary to load the weights and configuration of the pre-trained network to shape the model according to our interests.

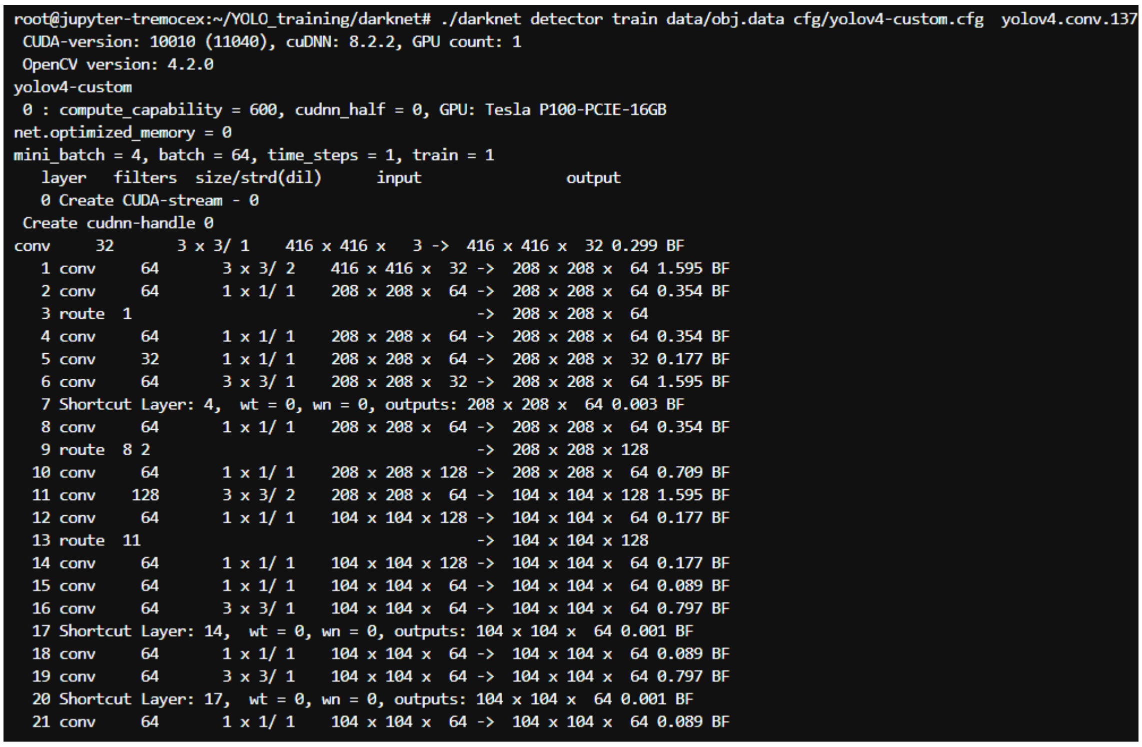
Briefly, among the parameters that may be changed in the configuration file, the batch value, subdivisions, maximum number of batches, steps, net size, number of classes, and also the value of some filters are some of them.

The batch number is the number of images the network loads per "iteration", the value of the subdivision is the number by which we divide the total amount of batches into mini-batches, i.e. we distribute the images over the different CPU and GPU colors. In this way, the process will repeat the cut eight times until the batch is complete and a new iteration will start with the number of images designated by the batch number. These values are parameters intended not only to speed up the training process but also to generalize the training.

The value put into the steps usually corresponds to 80% and 90% of the maximum value of batches (iterations). They serve as markers for adjusting the learning rate of the model. The standard general training setup for the YOLO family, for assistance and implementation in phase 0 of the project is as follows:

- $batch = 64$
- $subdivisions = 16$
- $n_{classes} = [2, 7]$
- $max_{batches} = n_{classes} * 2000$
- $steps = max_{batches} * 0.8, max_{batches} * 0.9$
- Network Size:
  - $width = 32n, \forall n \in N$
  - $height = 32n, \forall n \in N$
- Convolutional Layers:
  - $filters = (n_{classes} + 5) * 3$
- Gaussian Layers:
  - $filters = (n_{classes} + 9) * 3$

To send all the information about the custom training we want to perform, it is necessary to "pack" the data related to the images and respective annotations, as well as a file that tells the network which classes are being used to do so. Regarding the dataset, the validation within the YOLO net training is automatic, since the dataset is within the net configuration according to default configurations. The subclasses to be considered will then be in a ".names" type file which contains a list of the different classes in writing. In the following figure 4.23 we can see an example of an "obj.data" file that contains all the information and configuration needed to be accessed at once by the YOLO network.



```
root@jupyter-tremocex:~/YOLO_training/darknet# ./darknet detector train data/obj.data cfg/yolov4-custom.cfg  yolov4.conv.137
 CUDA-version: 10010 (11040), cuDNN: 8.2.2, GPU count: 1
 OpenCV version: 4.2.0
yolov4-custom
 0 : compute_capability = 600, cudnn_half = 0, GPU: Tesla P100-PCIE-16GB
net.optimized_memory = 0
mini_batch = 4, batch = 64, time_steps = 1, train = 1
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                                  ->  208 x 208 x  64
   4 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   5 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   6 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
   8 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   9 route  8 2                                ->  208 x 208 x 128
  10 conv     64       1 x 1/ 1    208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
  11 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
  12 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  13 route  11                                 ->  104 x 104 x 128
  14 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  15 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  16 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
  18 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  19 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
  21 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
```

Figure 4.23: Training Environment for Custom Object Detectors in Darknet (YOLOv4)

Once the information needed to run the model with the customized information for our problem is collected and compressed, it remains to upload to the darknet environment the pre-trained network that contains the weights of the original network trained on benchmark datasets (figure 4.23).

4.4 Tensorflow Object Detection API

Tensorflow is an open-source toolkit for numerical computing and large-scale machine learning that makes obtaining data, training models, serving predictions, and improving future outcomes easier using Google Brain TensorFlow [6]. Tensorflow combines Machine Learning and Deep Learning models and algorithms into one package. It makes use of Python as a user-friendly front-end and executes it in optimized C++. Developers may use Tensorflow to construct a graph of calculations to execute. Each connection in the network represents data, whereas each node represents a mathematical action. As a result, rather than worrying about little issues like how to connect the output of one function to the input of another, you can focus on the big picture.

In CV, creating effective machine learning models capable of localizing and recognizing numerous objects in a single picture remains a major issue. The TensorFlow Object Detection API is an open-source framework based on TensorFlow that enables building, training, and deploying object detection models simply. TensorFlow has a Model Zoo with symbolic models that may be utilized for inference. For their respective datasets, all of the models in this model zoo have pre-trained parameters. It comes with a set of pre-trained models that have been trained on a variety of datasets, including the MS COCO [59] dataset, the

KITTI [6] dataset, and the Open Images Dataset.

It is with this framework that we will train the remaining types of object detection models, both two-stage methods and one-stage methods. Namely the SSD [60], EfficientDet [102], and CenterNet [25] models within the one-stage methods, and from the R-CNN family the Faster R-CNN models, for the two-stage methods.

In the table 4.2 it is possible to check the networks and their results in terms of time and score for the MS COCO dataset that will be trained and provided by Tensorflow's Object Detection API to aid in phase 0 of this project.

Model name	Speed (ms)	COCO mAP	Output Type
CenterNet Resnet50 V1 FPN 512x512	**27**	**31.2**	**Boxes**
CenterNet Resnet101 V1 FPN 512x512	**34**	**34.2**	**Boxes**
CenterNet MobileNetV2 FPN 512x512	**6**	**23.4**	**Boxes**
EfficientDet D0 512x512	**39**	**33.6**	**Boxes**
EfficientDet D1 640x640	**54**	**38.4**	**Boxes**
EfficientDet D2 768x768	**67**	**41.8**	**Boxes**
SSD MobileNet V1 FPN 640x640	**48**	**29.1**	**Boxes**
SSD MobileNet V2 FPNLite 640x640	**39**	**28.2**	**Boxes**
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	**46**	**34.3**	**Boxes**
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	**57**	**57**	**Boxes**
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	**80**	**35.4**	**Boxes**
Faster R-CNN ResNet50 V1 640x640	**53**	**29.3**	**Boxes**
Faster R-CNN ResNet101 V1 640x640	**55**	**31.8**	**Boxes**
Faster R-CNN ResNet152 V1 640x640	**64**	**32.4**	**Boxes**

Table 4.2: List of models in Model ZOO (Tensorflow) used to train our custom object detectors

4.4.1 Training Custom Object Detector

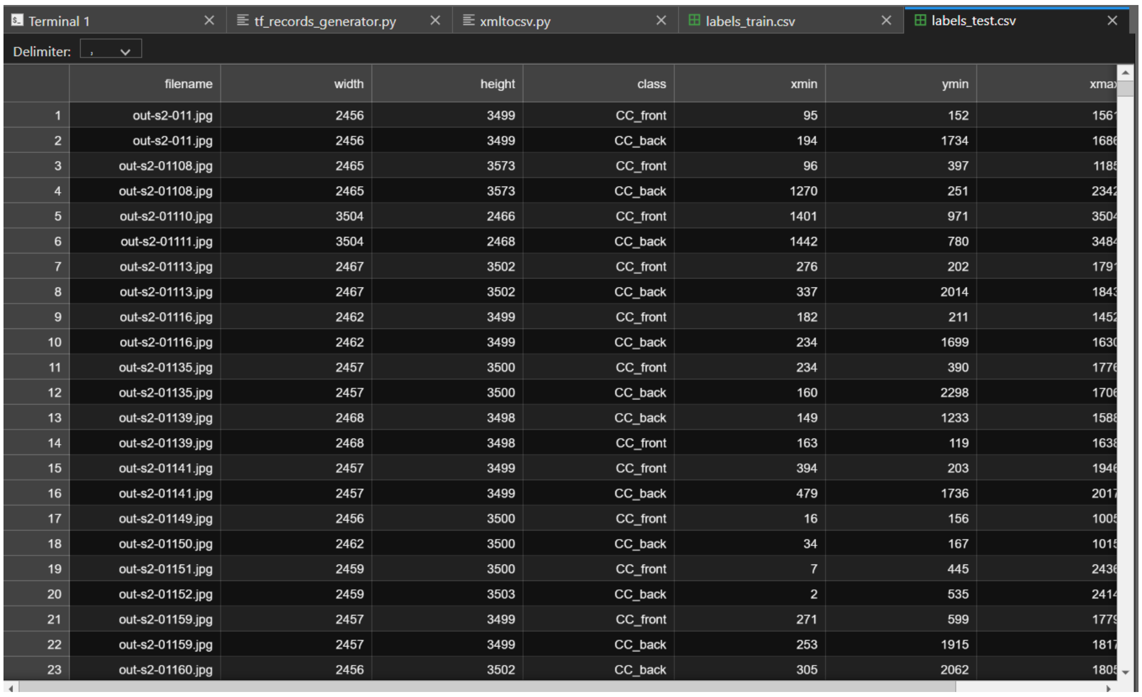
In terms of training custom object detectors in a Tensorflow environment, it is first necessary to initially install Tensorflow. In a Linux environment, this task becomes quite simple as only one command line is required in the terminal. Then you need to integrate Tensorflow with CUDA, i.e. enable GPU computing model in the Tensorflow framework. After this, all you need to do is integrate the Object Detection API into Tensorflow.

The Object Detection Application Programming Interface (API) only interprets XML format, i.e. PASCAL VOC annotation. Unlike when training models in the Darknet environment, Tensorflow does not process images and annotations individually. This is because Tensorflow uses its own binary data storage structure, the TFRecord format.

When dealing with big datasets, the use of a binary file format, like TFRecord, for data storage can have a considerable influence on the performance of our import pipeline and, as a result, on our model's training time. Binary data takes up less disk space, takes less time to transfer, and can be read from disk considerably faster. This is especially true if our data is stored on spinning drives, where read/write speed is significantly worse [6].

However, the TFRecord file format's speed is not its sole benefit. It is designed to work with Tensorflow in a variety of ways. To begin with, it makes combining numerous datasets simple and interacts smoothly with the library's data import and preparation features. This is a benefit, especially for datasets that are too big to be held entirely in memory, because only the data that is needed at the time (e.g. a batch) is loaded from disk and processed. Another important benefit of TFRecords is that they can be used to store sequence data — such as a time series or word encodings — in a form that is both efficient and (from a coding standpoint) flexible.

To build the TFRecords, it is first necessary to map all the input images and their respective annotations to a CSV file type with the help of a script. After this and with the help of the framework it is then possible to convert this mapping loaded in CSV to TFRecords. A function provided in the Tensorflow API helps in the creation of training and test TFRecords, where it is done automatically, according to the values in the CSV file (figure 4.24).



	filename	width	height	class	xmin	ymin	xmax
1	out-s2-011.jpg	2456	3499	CC_front	95	152	1561
2	out-s2-011.jpg	2456	3499	CC_back	194	1734	1686
3	out-s2-01108.jpg	2465	3573	CC_front	96	397	1185
4	out-s2-01108.jpg	2465	3573	CC_back	1270	251	2342
5	out-s2-01110.jpg	3504	2466	CC_front	1401	971	3504
6	out-s2-01111.jpg	3504	2468	CC_back	1442	780	3484
7	out-s2-01113.jpg	2467	3502	CC_front	276	202	1791
8	out-s2-01113.jpg	2467	3502	CC_back	337	2014	1843
9	out-s2-01116.jpg	2462	3499	CC_front	182	211	1452
10	out-s2-01116.jpg	2462	3499	CC_back	234	1699	1630
11	out-s2-01135.jpg	2457	3500	CC_front	234	390	1776
12	out-s2-01135.jpg	2457	3500	CC_back	160	2298	1706
13	out-s2-01139.jpg	2468	3498	CC_back	149	1233	1588
14	out-s2-01139.jpg	2468	3498	CC_front	163	119	1638
15	out-s2-01141.jpg	2457	3499	CC_front	394	203	1946
16	out-s2-01141.jpg	2457	3499	CC_back	479	1736	2017
17	out-s2-01149.jpg	2456	3500	CC_front	16	156	1005
18	out-s2-01150.jpg	2462	3500	CC_back	34	167	1015
19	out-s2-01151.jpg	2459	3500	CC_front	7	445	2436
20	out-s2-01152.jpg	2459	3503	CC_back	2	535	2414
21	out-s2-01159.jpg	2457	3499	CC_front	271	599	1779
22	out-s2-01159.jpg	2457	3499	CC_back	253	1915	1817
23	out-s2-01160.jpg	2456	3502	CC_back	305	2062	1805

Figure 4.24: Environment for TFRecords creation

The data set will be divided into training (train.record) and validation (validation.record) sets since we need to train and validate our model. The training set serves a simple purpose: it is a collection of instances from which the model learns. The validation set is a set of examples used to iteratively test model correctness during training.

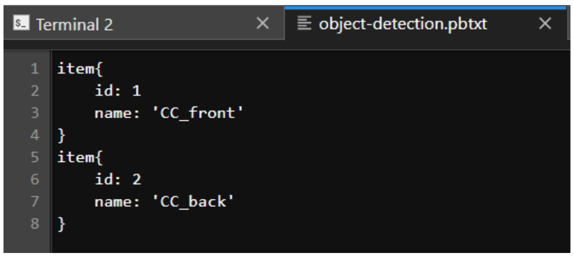
Once the input data for the model to be trained in the Tensorflow environment has been treated, it is now necessary to configure, similarly to what was seen before the configuration pipeline, that is, the parameters and data for the model to access. To do this we must first choose and extract the pre-trained model we want to train before we do its configuration.

Within the pre-trained model directory, in addition to the checkpoints of the previous training, i.e. the values of the weights of the pre-trained net, there is also a file called "pipeline.config". Within this file, some parameters must be changed to fine-tuning the pre-trained model to the input data from our dataset. These parameters are as follows:

- $batch_{size} = 8$
- $n_{classes} = [2, 7]$

- *fine_tune_checkpoint*: “Path to the checkpoint file ckpt-0”
- *fine_tune_checkpoint_type*: “detection”
- *train_input_reader*:
  - *label_map_path*: “label_map.pbtxt”
  - *tf_record_input_reader*:
    - *input_path*: “Path to training TFRecord file”
- *eval_config*:
  - *metrics_set*: "coco_detection_metrics"
  - *use_moving_averages*: false
- *eval_input_reader*:
  - *label_map_path*: "label_map.pbtxt”
  - *tf_record_input_reader*:
    - *input_path*: “Path to testing TFRecord file”

The file "label_map.pbtxt" (figure 4.25), similarly to what happened in the Darknet, is the file that contains the description and mapping in writing of the different classes to be used for training the Object Detection model. Since we are talking about a fine-tuning technique where we take advantage of the weights already pre-trained on a benchmark dataset to train custom models by knowledge transfer, when it comes to, for example, extracting features from images to compute objects. It is therefore necessary to provide to the configuration of our new model the path to the file "checkpoint/ckpt-0" that holds the optimal weights of the pre-trained model uploaded to the Tensorflow environment.



```
item{
    id: 1
    name: 'CC_front'
}
item{
    id: 2
    name: 'CC_back'
}
```

Figure 4.25: File "label_map.pbtxt", containing the information about the classes inside the dataset

Once all the conditions for model training have been met in Tensorflow’s Object Detection API, all that is needed now is to run the script inside the Object Detection project called ”*model_main_tf2.py*” and together give as arguments the path to the pre-trained model that will be used for fine-tuning and simply the path to the "pipeline.config" file that gathers all the necessary information for training the custom Object Detection model.

4.5 Evaluation Metrics

In terms of evaluation, it is important to emphasize that any metrics that can be used during the training of the two types of methods, either one-stage methods or two-stage methods, are relevant since they allow us to predict in advance what the behavior of the trained models will be. In addition, although a practical validation of the model running in random scenarios is always necessary during the project, it is important to have a preconceived notion of their behavior during their learning.

One of the metrics already mentioned in this report is the mAP, which compares the ground-truth BB inside the validation pre-selected dataset to the detected box and returns a score. The higher the score, the more accurate the model is in its detections. It is the product of precision and recall in detecting BBs. It is a good combined measure for how sensitive the network is to objects of interest and how well it avoids false alarms. The higher the mAP score, the more accurate the network is but that comes at the cost of execution speed which we want to avoid here. In Darknet, during training, we can track the progress of the model by setting the flag "-map", i.e., mAP.

It is from this value that we will later evaluate the final trained models in terms of accuracy. Furthermore, a relevant evaluation metric in training custom object detection models is the loss value. To calculate loss, YOLO employs the sum-squared error (4.1) between the forecasts and the ground truth. The classification and localization losses make up the loss function (errors between the predicted boundary box and the ground truth). Theoretically speaking a good training corresponds at the end of iterations to a low loss value and a high mAP value. The mAP value will stabilize over time until it reaches an optimum (figure 4.26.

$$\begin{aligned} &RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2 \\ &f(x_i) = \text{predicted value of } y_i \\ &n = \text{upper limit of summation} \end{aligned} \tag{4.1}$$

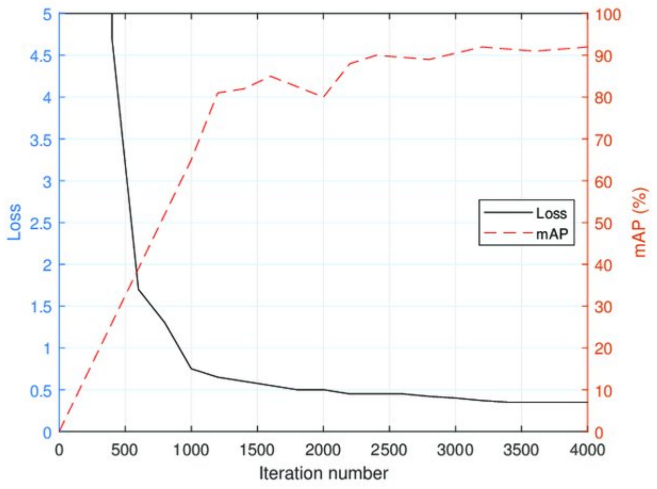
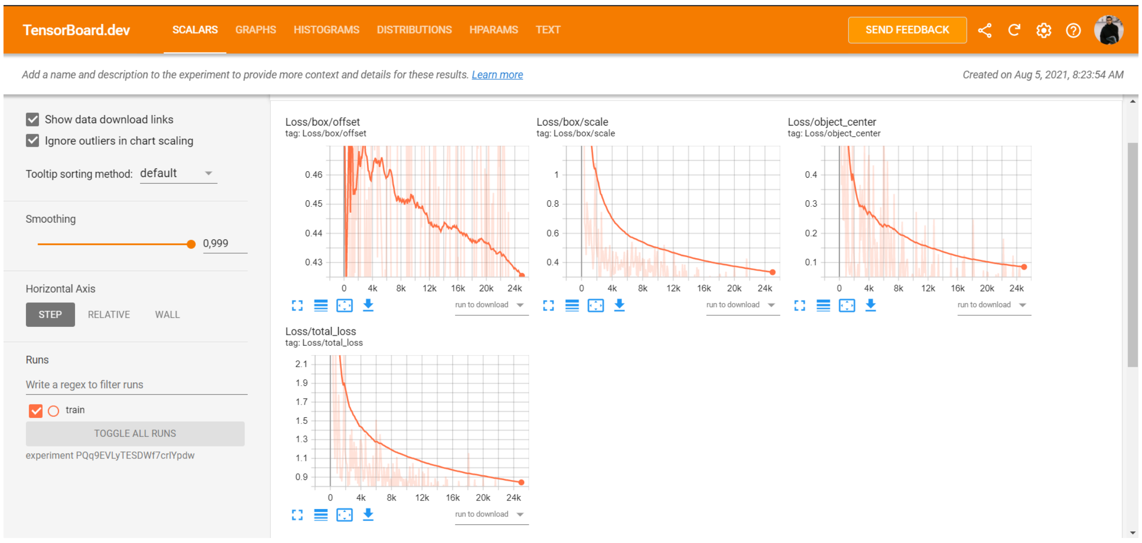


Figure 4.26: Loss and mAP graph depending on training iteration numbers [53]

Also, YOLO and any other algorithm that provides predicted bounding boxes as output can be evaluated using Intersection over Union (IoU). It is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

It is worth noting that, as the training progresses, the model saves the weights generated during the model's training in a backup directory every thousand iterations. Allowing not just for a more sophisticated iteration of the training, but also for the registration and categorization of the optimal weights for the issue at hand, using measures like IoU and mAP. As well as the values of the confusion matrix linked with the mAP value's constant metric evaluation, which is calculated every two thousand iterations.

TensorFlow has the advantage of allowing us to continually monitor and see a variety of training/evaluation variables as our model is being trained. Tensorboard is the specific technology that allows us to perform all of this [6].



Add a name and description to the experiment to provide more context and details for these results. Learn more

Created on Aug 5, 2021, 8:23:54 AM

☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0,999

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ train

experiment PQq9EVLyTESDWf7crfYpdw

Figure 4.27: Tensorboard API Environment - 4 Scalars regarding the Custom Object Detector Training

For machine learning projects, TensorBoard provides the required visuals and tools. This enables the tracking and visualization of metrics like loss and precision, as well as the visualization of the model's graph (ops and layers), the visualization of weights,scalar views, or other tensors as they change over time, the projection of embeddings in a lower-dimensional space, the display of image, text, and audio data, and the creation of TensorFlow profiles.

This page is intentionally left blank.

Chapter 5

Phase 1 - Image Preprocessing

As has been mentioned several times during this report, image preprocessing represents a critical factor and of utmost importance when dealing with an image classification problem, regardless of the phase we are considering. Preprocessing images is always essential and indispensable to safeguard the performance and quality of models.

This phase is relevant for Phase 2, where we make the practical application of an Optical Character Recognition (OCR) engine to extract and format the interesting data from the fields in the detected Identity Documents (IDs). Regarding the experiments performed in this area, all of them were focused more on the preprocessing part of Phase 2 of the pipeline to be created, that is, in all experiments that will be described the operations and algorithms implemented were intended to assist the preprocessing in the OCR task.

5.1 Input Type and Image Quality Problems

From the scenarios analyzed and studied throughout this project, and whose presence is included in the dataset provided by Altice/MEO, it was possible to collect a series of problems and divide certain cases into different types of operations and methods. In a first analysis, the problems in the input IDs were subdivided into, problems with color IDs, that is, when the scanning was done by the camera, and problems with black and white IDs, that is, when the ID document was scanned.

Within the first type, these usually have a good average quality, but some problems are detected. One of the problems is for example the existence in some cases of the reflection effect of the flashlight from digital or mobile cameras. This becomes a problem for example when it occurs in an area of the Machine Readable Zone (MRZ) code, where part of the characters end up being obscured and the pixels in that area end up having various interferences in terms of their mapping (figure 5.1. This leads to the OCR engine later on extracting the different fields present in the ID being incomplete, and in the specific case of the MRZ code, non-existent.

Another problem still within this type is that often some of the ID captures are blurry or have motion blur.

In the second type, where we focus on the general problems applied to common scans, some details must be taken into account. For example, right from the start, the light exposure that the document was subjected to at the time of scanning (figure 5.2). This analysis is relevant because numerous consequences in terms of image quality may emerge

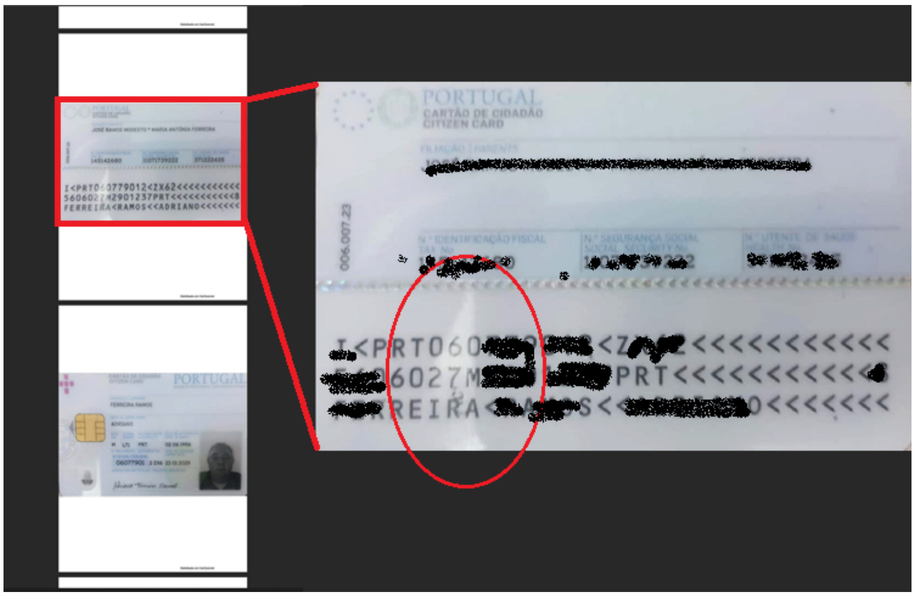


Figure 5.1: Input Type 1 - Colored Images (Digital Images)

from this. These can include problems with the contrast value and brightness of the pixels, deconstruction at the pixel level caused by image discoloration in various areas of the captured image, among others.

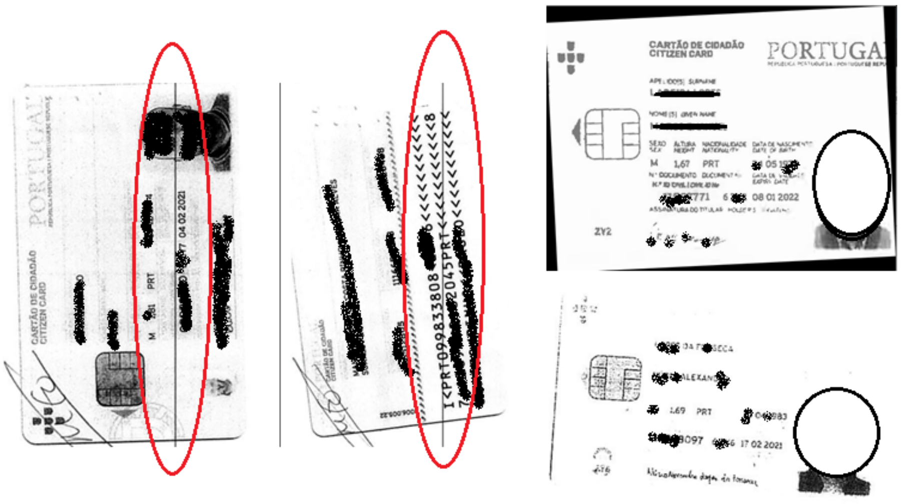


Figure 5.2: Input Type 2 - Grayscale Images (Digitized Documents)

Despite the division, other problems are encountered in most cases, regardless of the color scheme present in the scanned image. Within these problems, it is important to highlight the random orientation at which the ID can be found within the document. In terms of extraction, it is pertinent that a primary survey of the original angle that the document is subject to be done and a subsequent correction of its orientation so that it is possible for the OCR engine to correctly read the characters. Besides this problem, there may also be the appearance of handwritten characters or even geometric shapes (such as vertical or horizontal lines) in the form of noise, which may disturb the correct extraction of information within the ID (figure 5.2).

5.2 Methods and Experiments

Briefly, most of the algorithms and morphological operations performed and already studied applied in this phase were implemented with the help of the Open Source Computer Vision Library (OpenCV) tool [15].

OpenCV is a free software library for computer vision and machine learning. OpenCV was created to offer a standard infrastructure for Computer Vision (CV) applications and to let commercial goods incorporate machine perception more quickly. More than 2500 optimized algorithms are included in the library, which contains a comprehensive range of both traditional and cutting-edge computer vision and machine learning techniques.

Three of the methods used in the construction of this phase comprise the use of thresholding algorithms. In an OpenCV environment, for each pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value.

The thresholding is done with the function $cv2.threshold$ The first parameter is the source picture, which should be a grayscale image, implying that we must first convert an Red, Green and Blue (RGB) image to a grayscale image before processing it. This may be accomplished by utilizing OpenCV's *cvtColor* function. The threshold value is the second parameter, and it is used to categorize the pixel values. The maximum value allocated to pixel values above the threshold is the third parameter. The fourth argument of the function in OpenCV allows several forms of thresholding. But in this approach, we use one global value as a threshold.

As has been mentioned several times, the probability of several distinct lighting conditions occurring in a scanned image is quite a lot, so fixing the overall threshold value may not be the best-generalized solution. Therefore, it is necessary to explore a method that can better handle these sorts of scenarios, adaptive thresholding.

The method establishes a pixel's threshold based on a tiny region surrounding it. As a consequence, various thresholds are obtained for different parts of the same image, resulting in improved results for images with variable illumination.

In the OpenCV library, the function $cv2.adaptiveThreshold$ takes as the same as the normal threshold function plus three. A first argument of the adaptive method decides how the threshold value is accomplished:

- $cv2.ADAPTIVE\_THRESH\_MEAN\_C$: The threshold value is the mean of the neighbourhood area minus the constant C.
- $cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C$: The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant C.

A second argument which is the blockSize determines the size of the neighborhood area, and a third and final argument, C, is a constant that is subtracted from the mean or weighted sum of the neighborhood pixels.

We utilized an arbitrarily selected value as a threshold in global thresholding. Otsu's technique, on the other hand, eliminates the need to pick a value by determining it automatically.

Consider a bimodal picture, which has only two unique image values and hence only two peaks in the histogram. In the center of those two numbers would be a fair threshold.

Similarly, Otsu's approach uses the picture histogram to calculate the best global threshold value.

The *cv2.threshold* function is used for this, with $cv2.THRESH\_OTSU$ given as an additional flag. The threshold value can be set at any point in time. The method then returns the first output, which is the ideal threshold value.

One of the problems already raised is the contrast problem that can exist in the detected IDs. To work around this problem use of the Adaptive Histogram Equalizatio (AHE) technique provided by OpenCV can be used.

AHE is a computer image processing approach for enhancing picture contrast. The adaptive technique varies from traditional histogram equalization in that it computes many histograms, each corresponding to a different part of the picture, and utilizes them to disperse the image's brightness values. As a result, it's ideal for boosting local contrast and sharpening edge definitions in different parts of an image. AHE, on the other hand, has a propensity to exaggerate noise in relatively homogenous areas of a picture. Contrast Restricted Adaptive Histogram Equalization (CLAHE), a kind of AHE, prevents this. CLAHE that corrects for contrast overamplification. Works on tiles, which are tiny areas of a picture rather than the full image. To remove the false borders, the nearby tiles are merged using bilinear interpolation. The contrast of pictures may be improved with this technique. CLAHE may also be used on color pictures, where it is generally applied to the luminance channel, and the results of equalizing only the luminance channel of an HSV image are significantly better than equalizing all of the channels of an RGB image.

The image is split into tiny blocks called "tiles" in this approach (tileSize is 8x8 by default in OpenCV). Then, as is customary, each of these blocks is histogram equalized. As a result, in a tiny area, the histogram would be limited to a small area (unless there is noise). Noise will be increased if it exists. Contrast limiting is used to avoid this. Before implementing histogram equalization, every histogram bin that exceeds the given contrast limit (default 40 in OpenCV) is clipped and distributed evenly to other bins. Bilinear interpolation is used after equalization to eliminate artifacts in tile borders.

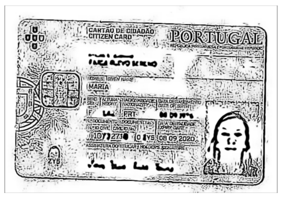
In figure 5.3 we can observe a quick demonstration of how internally the usage of adaptive threshold and OTSU binarization works.

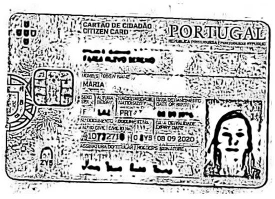
One of the resolutions to the problem of occurrence of horizontal or vertical lines or type of noise enhancing shapes in the detected IDs involves the application of these thresholding techniques, namely, the adaptive thresholding technique.

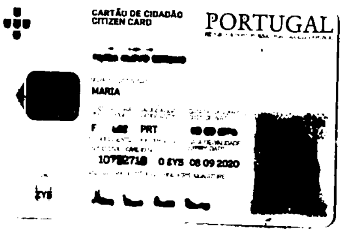
A proposal for the automatic detection and removal of lines and other types of contours using the OpenCV library consists of the following steps:

1. Thresholding (Binarization with Otsu Approach);
2. $BitewiseNOT$ function to invert the image pixel map;
3. Adaptive Mean Thresholding;
4. Erosion followed by a Dilation;
5. Define kernel Length;
6. Create structure element for extracting vertical lines through morphology operations;
7. A vertical kernel of $kernel_{shape} = (6, 1)$, which will detect all the vertical lines from the image;

(a) Original ID


(b) Adaptive Thresholding (Gaussian Average)


(c) Adaptive Thresholding (Mean)


(d) Thresholding with OTSU Binarization

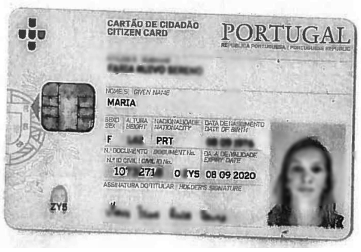

(e) CLAHE

Figure 5.3: Different Applications of OpenCV functions to deal with problems of type 1 Input

8. A horizontal kernel of $kernel_{shape} = (7, 1)$, which will help to detect all the horizontal lines from the image.

9. Morphological operations (erosion followed by dilation) to detect vertical and horizontal lines from an image

10. Subtract the founded pixels that corresponds to lines or other geometrical objects of such kind from the binarized image;

11. $BitewiseNOT$ function to invert the image pixel map.

In the figure 5.4, we can observe this proposal for removing ID lines in action.

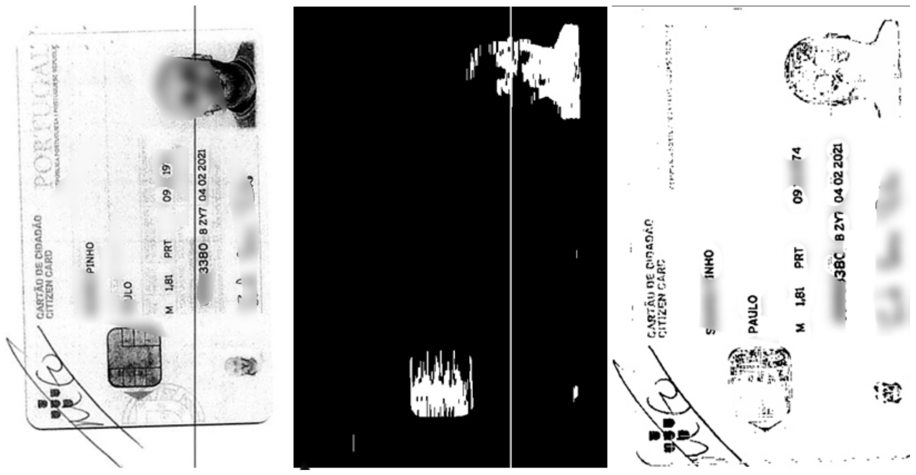


Figure 5.4: Lines Removal Process

To conclude what are the methods adopted and implemented to be part of this preprocessing phase of the pipeline of the system to be built, it remains to be seen how we can get around the reflection problem in IDs captured by digital camera. One possible approach is to first preprocess the ID by applying a noise cleaning function such as the function *cv2.medianBlur*. After this, the first phase of the approach begins.

In the first step, we perform binarization from the median value locally in a slice of the original image. From here we need to assume that the majority of pixels in the slice are background, the median value of the intensity probably belongs to the background. We allow a soft margin of $DELTA$ (default value is 25), but we need to keep everything other than the background. We also do simple morphological operations here to get a robust result.

The advanced approach set up to remove glare in a non-damaging way from detected IDs comprises the following steps:

- Step 1 - Gamma Correction;
- Step 2 - Adaptive Binarization to Detect the Text Blobs;
- Step 3 - Soft Binarization.

This may appear to be a superfluous step at first, but its value cannot be overstated: in a way, it normalizes the ID to identical exposure distributions, allowing us to pick relevant hyper-parameters afterward. The text blobs are then adaptively binarized out by splitting the picture into blocks of size $BLOCK\_SIZE$ in the next phase. The challenge is to pick a size that is large enough to include a substantial piece of text and background (i.e. larger than any symbols), yet small enough to avoid being affected by lighting conditions (i.e. "large, but still local"). We conduct locally-adaptive binarization inside each block: we check at the median value and assume it is the background (since the $BLOCK\_SIZE$ is large enough for the majority of it to be the background). Then we define $DELTA$, which is essentially a "how far out from the median we will still consider it a threshold".

We can finally begin the whitening operation after we have the blobs that cover the symbols. Also, we want to keep the effect of progressive transfer from black to white in this last stage. The goal is to make it printable at the end of it. The basic principle is that the more a pixel's value departs from the local minimum value (after thresholding), the more likely it is to belong to the background. A family of Sigmoid functions, rescaled to the range of local block sizes, can be used to describe this (so that this function is adaptively scaled through the image). In figure 5.5 we can see this pipeline approach working in the context of our problem.

Of the various difficulties that can be encountered when performing the OCR task, we should highlight the fact that in many cases there is the possibility that the orientation of the document is not always in what is expected for the application of the OCR engine. As illustrated in figure 5.6, the scanned original document presents an orientation that does not correspond to normal. Therefore, it is necessary to use an algorithm, which not only automatically detects its orientation, but at the same time corrects it.

To do this, the OSD [53] algorithm is used. It is possible to find out which is the original orientation and correct the orientation so that the OCR engine is capable of detecting and extracting characters in the input document. The OSD is a simple yet effective algorithm to combine script and page orientation detection using the Tesseract shape classifier.

Despite the OSD algorithm's good average results, failure cases do occur, such as when documents contain degraded or handwritten text, or when documents contain unusual images or line drawings that are not removed from consideration in the preprocessing step,

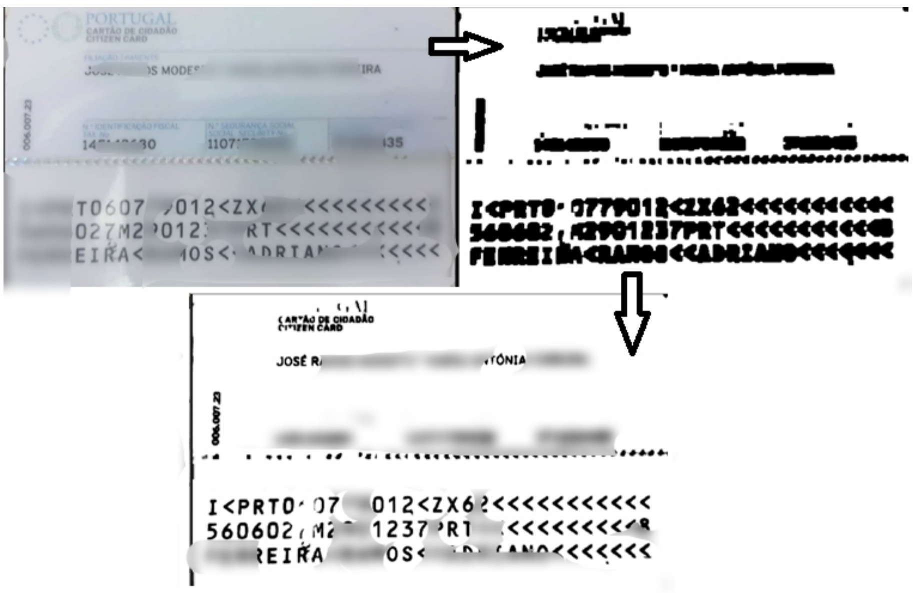


Figure 5.5: Glare Removal on ID of Type 1

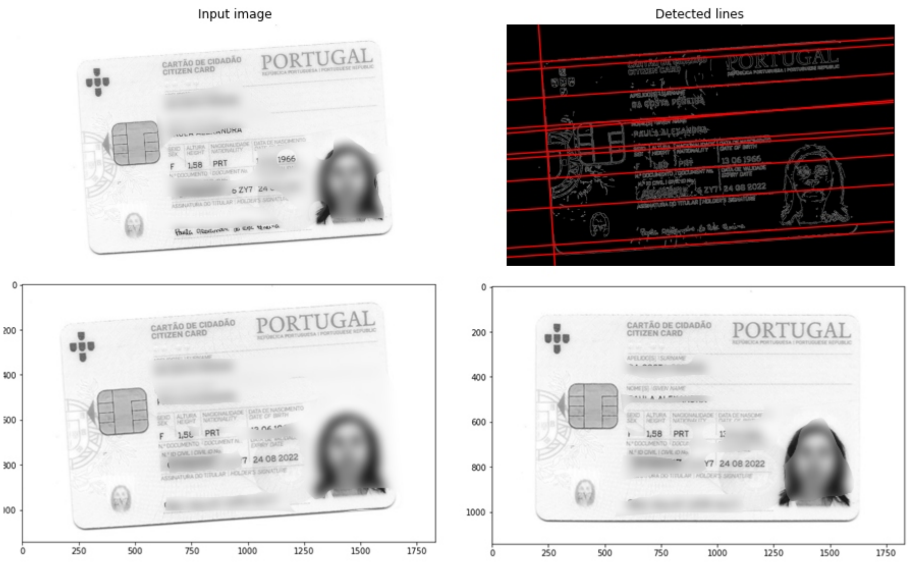


Figure 5.6: Orientation Script Detection (OSD) algorithm correcting the deskew angle of the detected ID

or when documents contain scripts in fonts that have not been trained on. Many of these sources of inaccuracy are linked to the broader OCR issue and are the focus of ongoing research.

From the experiments performed internally with some samples collected it was possible to realize that the algorithm works in most cases, but there are rare exceptions in which it considers correct the orientation in which the text is with a 180º orientation. So that there is no loss of information, there is a phase after the detection and correction of the angle of the ID, where a replica of it is made but rotated 180º.

5.3 Proposed Adaptive Preprocessing Pipeline

In addition to applying different preprocessing methods and algorithms, it is necessary to ascertain in a first instance, whether or not it is really necessary to apply them. As mentioned in the chapter 2, we can resort to intelligent image quality inference algorithms to ascertain, on a predefined scale, the quality of the detected IDs.

To perform a theoretical survey of the overall quality of the ID detected by the previous phase using the chosen Object Detection model, this inference can be performed, as mentioned before, using the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) algorithm [68]. Nearly, BRISQUE can be implemented using a package available for Python that receives as input the target image and returns its value in terms of image quality, on a scale from 0 to 100.

Based on this value collected by the algorithm, we can then infer whether it is worthwhile in terms of computational and temporal resources to input the target image into a preprocessing pipeline, where sometimes costly operations in terms of memory and processing are applied.

The proposal for the adaptive pipeline initially uses the BRISQUE method to determine the score of the image in terms of image quality. If this is less than 35.0, the ID that enters the pipeline is not processed and normally only goes forward with orientation correction, where two output images are created. Otherwise, the detected ID will go through a phase of different types of preprocessing, wherein each of these a version of the ID originated from the application of these methods is stored. Once we know the basic concept for assembling our adaptive preprocessing pipeline and the methodologies at our disposal we can then observe in the figure 5.7 the adaptive pipeline proposed for phase 1.

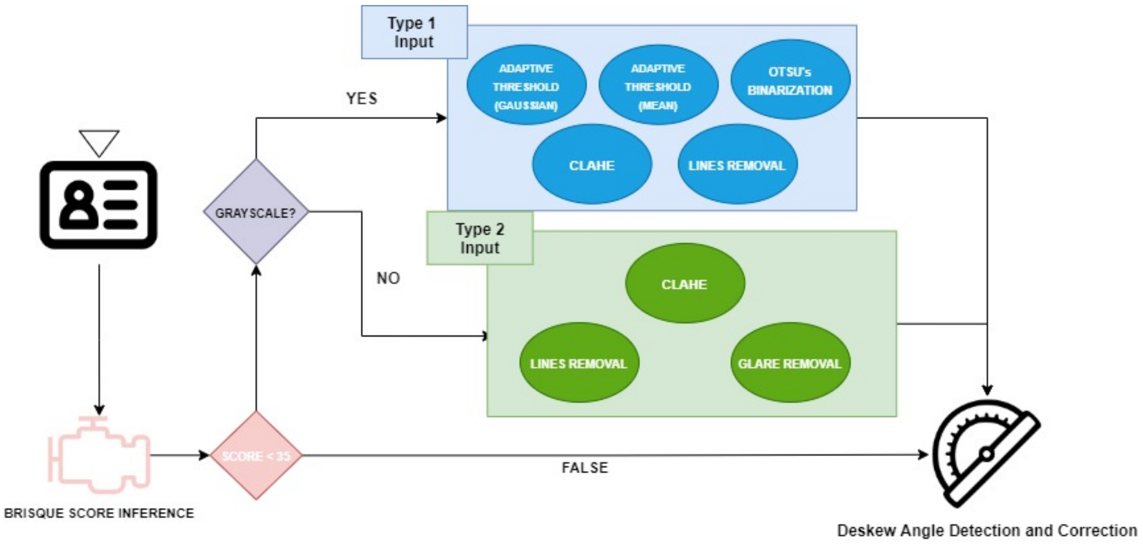


Figure 5.7: Adaptive Preprocessing Pipeline

This page is intentionally left blank.

Chapter 6

Phase 2 - Optical Character Recognition

As mentioned before, both the first phase and the second phase have a very relevant role in what are the future good results of the pipeline under construction.

Putting aside the hypothesis of failure in the remaining two phases it is ultimately up to the Optical Character Recognition (OCR) engine chosen to be incorporated into the solution to extract and compute the information present in the detected Identity Documents (IDs). Therefore it is imperative to spend all available resources to try to optimize the OCR engine's task as much as possible.

Advanced and powerful OCR engines allow us to save a lot of time and effort when creating, processing and repurposing various documents.

But this task becomes more difficult when we enumerate the difficulties that exist today in open-source OCR engines such as the Tesseract and EasyOCR engine already mentioned in chapter 2.

Both function best when the foreground text and background text are separated. In actuality, as we saw before with the Image Quality issues, ensuring these sorts of setup may be quite difficult. The higher the quality of the image (size, contrast, and lighting), the better the identification result. To improve OCR performance, some preprocessing is required; pictures must be sized correctly, have as much image contrast as feasible, and the text must be horizontally aligned. These engines are extremely strong, but they are not without their drawbacks. For example, both EasyOCR and Tesseract struggle with pictures that have artifacts such as partial occlusion.

On the other hand, the EasyOCR engine has powerful post-processing and character recognition tools superior to Tesseract, as was studied in chapter[SoA]. These limitations and advantages have to be put later side by side to find out which of the two solutions is more suitable for the good results acquired by the final software.

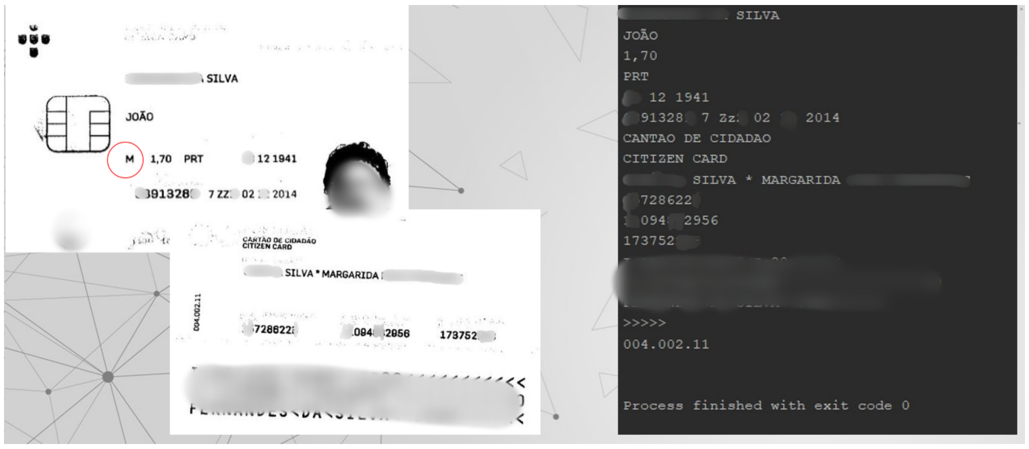
In general, poor-quality scans may produce poor quality OCR. Following the details and explanations provided in the previous two chapters, it is necessary to understand what strategies may be used to improve the confidence extraction of the OCR motors under study, as well as how these will be included in the pipeline under construction. With this in mind, a brief description of the methods used in this phase, as well as some of the strategies employed to improve and format the data extracted by the OCR motor in the post-processing phase, will be provided.

6.1 Exploratory Work

Although we have focused and spent more time on exploring the technology inherent to Phase 0 of the project, it is extremely important to keep in constant contact with the novelty related to the advances made in the OCR area, as well as the practical applications of it.

Still, within the offer given by Google Cloud Platform (GCP), the API Vision, already mentioned above, is Google's cloud tool responsible for the optical reading of characters in images, in other words, it is the OCR service created and maintained by GCP, based on Rest API.

As previously done, several tests have been made regarding the applicability of GCP API Vision with some documents provided by Altice/MEO. In figure 6.1, we can see a very fast and direct example of the application of the API Vision service.


[redacted] SILVA
JOÃO
M 1,70 PRT [redacted] 12 1941
[redacted] 91328 [redacted] 7 ZZ [redacted] 02 [redacted] 2014
CARTÃO DE CIDADÃO
CITIZEN CARD
[redacted] SILVA * MARGARIDA [redacted]
004.002.11
[redacted] 728622 [redacted] 094 [redacted] 2956 173752 [redacted]
[redacted]

```
[redacted] SILVA
JOÃO
1,70
PRT
[redacted] 12 1941
[redacted] 91328 [redacted] 7 Zz [redacted] 02 [redacted] 2014
CANTAO DE CIDADAO
CITIZEN CARD
[redacted] SILVA * MARGARIDA [redacted]
[redacted] 728622 [redacted]
[redacted] 094 [redacted] 2956
173752 [redacted]
[redacted]
>>>>>
004.002.11


Process finished with exit code 0
```

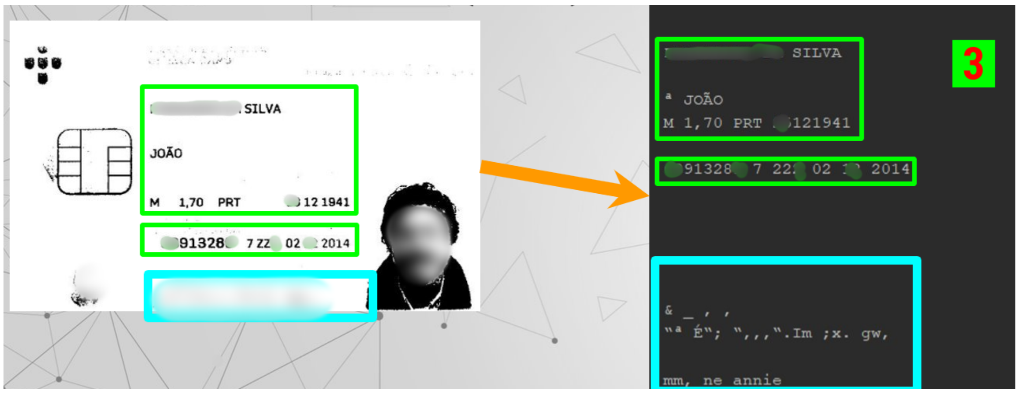
Figure 6.1: API Vision of GCP performing OCR task in an all input document (which is upside down)

From the case shown above, it is worth highlighting that this OCR engine is invariant to the conditions within the document. It automatically detects the orientation of the document and has good performance when recognizing characters in the entire document, regardless of the complexity of the image background and the quality of the image elements themselves. This leads us to believe that, in addition to being a state-of-the-art OCR tool, it has a series of preprocessing processes that ultimately assist the OCR task itself from the Vision API in an automatic way and in real-time.

To try to mirror the optimal performance found in GCP's Vision API, the same experiment was executed in an open-source OCR engine-Tesseract OCR. Tesseract OCR [105] is an open-source Optical Character Recognition software, originally developed by HP and for a long time hosted, adopted, and improved by Google. Currently, the project is hosted at GitHub. At the moment, the most recent version of Tesseract OCR is the 4.0. 0, but it is still in beta state, and therefore not all its features are 100% stable, such as the adaptation of training datasets from previous versions. It has no mechanisms for layout recognition, so it is not recommended for texts that have images, formulas, or more than one column.

Considering what was established about Tesseract OCR, it was necessary to use the version prior to version 4.0.0 taking into account that being in beta testing, it still does not have all the necessary functionalities to do OCR surveys for characters of Portuguese origin. Besides this, and as it was also discussed previously, Tesseract OCR is also insensitive to the orientation, contrary to what happens with the Vision API, and for this reason, it is necessary to initially use the Orientation Script Detection (OSD) algorithm before the subsequent OCR task. In addition, Tesseract OCR is not able to detect the language automatically also contrary to what happens with the Vision API from GCP. Therefore it had to be done a practical survey of the best possible combination of datasets to be used in the development of the OCR engine. In this case, the best combination achieved was when the English and Portuguese datasets were joined (figure 6.2).

Finally, it is important to say that the tests made with the different OCR engines studied are still very weak, so it is crucial to allocate time in seeking for an algorithm or solution that best fits the needs we have within this phase, which ends up being the fundamental part of the whole system. Although we have only experimented with existing solutions, it is important to say that the manual implementation of an OCR algorithm is always a solution and that it should be taken into account.



[redacted] SILVA
JOÃO
M 1,70 PRT [redacted] 12 1941
[redacted]91328[redacted] 7 ZZ[redacted] 02 [redacted] 2014
[redacted]
3

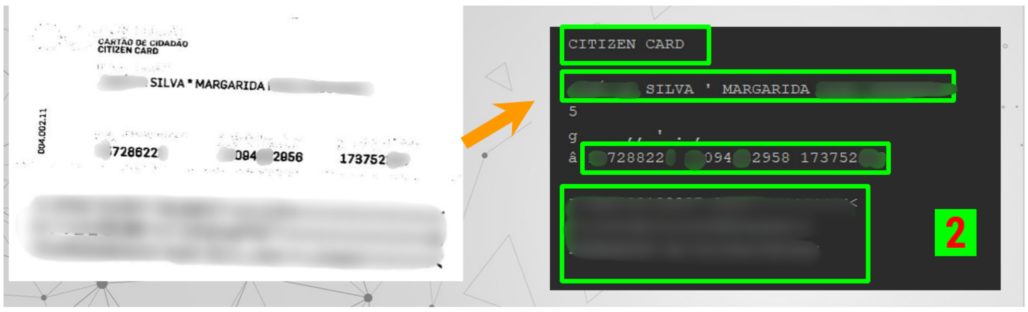
```
[redacted] SILVA

ª JOÃO
M 1,70 PRT [redacted]121941
[redacted]91328[redacted] 7 22[redacted] 02 [redacted] 2014

& _ , ,
"ª É"; ",,,".Im ;x. gw,

mm, ne annie
```

(a) Front Side of the Cartão de Cidadão (CC) results



CARTÃO DE CIDADÃO
CITIZEN CARD
[redacted] SILVA * MARGARIDA [redacted]
[redacted]728822[redacted] [redacted]094[redacted]2958 173752[redacted]
[redacted]
2

```
CITIZEN CARD
[redacted] SILVA ' MARGARIDA [redacted]
5
g  , , ' . ,
â [redacted]728822[redacted] [redacted]094[redacted]2958 173752[redacted]
[redacted]
```

(b) Back Side of the CC results

Figure 6.2: Tesseract OCR - Combination of English and Portuguese datasets

6.2 Methods

Despite the efforts already collected in the experimental tests, the focus remains, as mentioned before, on the use of open-source OCR engines, to benefit, from a commercial perspective, Altice/MEO. Of the OCR engines studied and implemented in the Python environment, the Tesseract engine [105] and the EasyOCR engine [47] were implemented.

For Tesseract to be implemented in Python we need first to get a wrapper named pytesseract [105] that will be used to invoke the tesseract script. In the case of the EasyOCR motor, there is also a wrapped project called easyocr-1.4 [47] that can be used easily in a Python environment.

Contrary to what we saw in the chapter 4 about phase zero of this project, no changes will be required in this phase in terms of configuration and implementation; that is, the OCR motors will be used in their default settings, in terms of architecture and inherent technology.

Both engines accept as input an image of any format (Portable Graphics Format (PNG), Joint Photographic Experts Group (JPEG),...). In the case of the Tesseract engine, it has several functions and formats for returning the extraction results to the image received as input. Of all the existing possibilities, in what is the output format of the extracted data, the format chosen is a dictionary. This output configuration is possible through the use of a function from the pytesseract library called *image_to_data*. This function not only receives as input the target image for extraction, but also receives as argument the list of languages to be detected during OCR processing, the expected output type (in this case *output.DICT*), and optionally an argument where we can place flags that modify the extraction configuration. The extraction performed with the Tesseract engine in the Python environment can be done by executing the following code line:

$$pytesseract.image\_to\_data(TARGET\_ID, lang = [lang,], output\_type = output.DICT)$$

Internally, when we type the code above, we're using an API to acquire information about the text in our image. The HTTP response from this call is a composition of boxes (rectangles) on the image highlighting text areas. Result-set contains values for multiple different levels. By checking the value of the level key, we can see what level box belongs to. When we say "level", we are talking about the different elements present in the document as input, such as page, block, paragraph, line, or word. Image can contain multiple blocks of the same type and these attributes are used to define the position of the block in a list or parent hierarchy. Each box shape (result) is defined by the top, width, height, left values.

Using the "level" values in the dictionary, we will get all of the occurrences of single words or characters logged by Tesseract in the API call with the target picture. It is important to extract textual information collected by the tesseract motor, as well as its relative coordinates within the ID, from the data stored in the returned dictionary, so that we may have a metric for the motor's performance, detection confidence, and recognition.

On the other hand, EasyOCR is a Python module that enables computer vision engineers to do Optical Character Recognition with ease. EasyOCR is by far the most basic approach to implement Optical Character Recognition when it comes to OCR. This is because the EasyOCR package has few dependencies, making it simple to establish the OCR development environment. The PyTorch library is used internally by the EasyOCR engine, which is necessary for executing detection models in a Central Processing Unit (CPU) or Graphics Processing Unit (GPU) context. Pytorch [77] is an open-source machine learning library based on the Torch library, which was developed primarily by Facebook's AI Research team.

Once EasyOCR is installed, only one import statement is required to import the package into the project. From there, all it is needed to perform OCR is the following line:

$$Reader([langs,]).readtext(TARGET\_ID)$$

In the first instance, a "Reader" object is created, which will store and retrieve the detection and recognition model for the pre-defined idioms. Following that, once the model has been saved to memory, it no longer needs to be saved since its contents are stored in an object of the type "Reader." After that, it becomes relatively simple to run the OCR engine with the image as the argument, by employing the single-line reading method "read-text," which, in addition to the image as input, can also accept as arguments the type of decoder (Connectionist Temporal Classification (CTC) by default), the beam width value (5 by default), batch size (1 by default), and, for example, a list of permitted characters ("allowlist" parameter).

In terms of output, we can also specify as input the type of format we want for the results obtained by the EasyOCR engine. By default, the standard format returns an array of arrays, which are made up of the detected Bounding Boxs (BBs) coordinates, the extracted text, and the confidence associated with the extraction of those in this order.

In both approaches, it is critical to recognize the importance of the coordinates related to the BBs of the characters or words detected while using OCR. This is because, based on this, we may deduce and investigate how the detection of reclaimed fields is carried out. In practical terms, the use and analysis of detected coordinates can be used to deduce, for example, the dimensions of a word or a group of words. From here, it may be possible to create a system that filters them according to project-specific requirements. Furthermore, it is used to demarcate areas of interest for extraction, preventing the addition of characters or background noise. As we will see in the following sections, several configurations and strategies were taken into consideration to improve internal data extraction from target fields. It is possible to simplify the OCR extraction process by using regular expressions and transformations in the picture of entry and to make it more particular for each extraction task that must be completed.

6.3 Extraction Strategies

To simplify and delimit the area of interest of extraction of the different information fields, several attempts were put to the test, and these resulted in the application of specific strategies to improve not only the quality but also the quantity of extracted information.

As seen previously, both the EasyOCR engine and the Tesseract engine allow the addition of different parameters at the time of the extraction of the characters present in the ID returned by the previous stage. Within these parameters, we will highlight the parameter that filters the list of characters allowed in the recognition phase of the engines. This parameter receives as value a string with the list of alphanumeric and special characters that delimit the search universe. The applicability and usefulness of this parameter bring numerous advantages and new extraction strategies.

The focus of this project will remain on studying the applicability of OCR engines specifically just for the class of CC IDs, as mentioned in chapter 3. Beyond this, and looking now at the ID itself, there are numerous ways, supported by Open Source Computer Vision Library (OpenCV) methods, to delimit and isolate the fields we are looking to extract in CCs. Here another different strategy comes in in addition to delimiting the recognition base of the engine. With these two strong components, we can then define upfront several strategies that will facilitate the extraction process in both parts that make up the CCs. As already seen, the two parts that make up the CC have different features and as such, it is necessary to analyze each one separately.

From the analysis of the CC layout and features in chapter 3, it is worth noting the uniform distribution that the different target fields have in the front part of the CC. Based on this knowledge we proceeded to implement two types of extraction from the front of the CC. The first method performs the extraction of the entire processed ID, while the other makes use of cutouts to delimit the different extraction areas, i.e. the location of the different fields in the ID (figure 6.3).

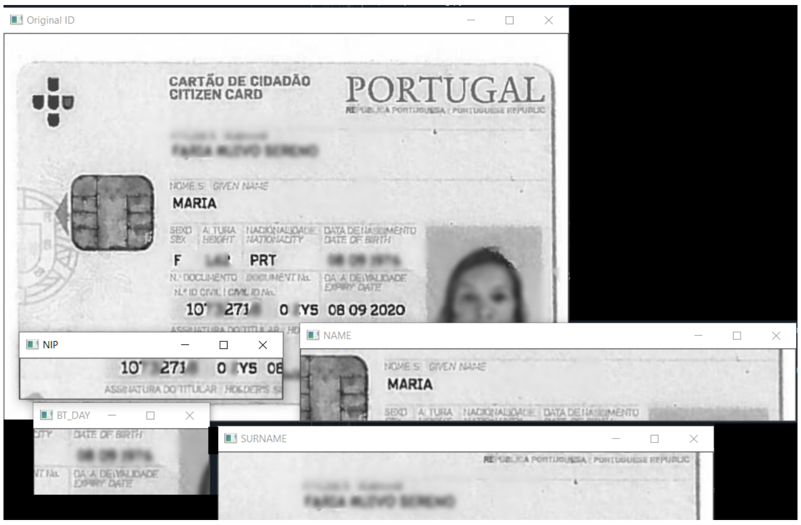


Figure 6.3: Extraction type 1 applied to the CC_{front}

As is possible in figure 6.3 the different identifying fields in the CC are first clipped from the original image and used individually to be extracted by the OCR engine, i.e. in this strategy, each field is an input image. In this way, we are not only able to get more accurate results with high confidence values. Besides this, within this strategy, there is yet another procedure that helps even more in extracting the information contained in the CC. Considering that we are extracting separately each of the fields from the CC, we can apply, as said before, filters for the characters to be recognized by the loaded model.

According to the nature of each field present in the CC, various character lists and delimitations were considered (table 6.1). This strategy turns out to have huge potential in what is the mission of reducing as much as possible the loss of information extracted from the front end of the CC. Given this, in what is the extraction of information from the front of the CC this type of strategy found becomes primary over the second strategy mentioned. Despite this, and taking into account the randomness of the scenarios present in a Customer Relationship Management (CRM) environment, it is very likely that this first strategy will not always present the ideal results, and therefore the application of the second strategy serves more as a backup so that there is always the possibility of having information collected and processed for the front part of the CC.

Regarding the back of the CC, and from what was analyzed in chapter 3, we are interested in focusing our efforts on the extraction of the Machine Readable Zone (MRZ) code and the Número de Identificação Fiscal (NIF) field. While the MRZ field features prominently and uniquely in the arrangement of the information fields on the back of the CC, it is necessary to initially apply effort to what is the extraction of the NIF field, since it only appears on this part of the card.

The extraction of the NIF field raises right from the start its centered disposition in the document, which hinders the effort that can be applied in terms of delimitation. This is because next to it are two more information fields of similar length. In terms of extraction by the OCR, this leads to the detection of three possible values for NIF in a global extraction. To avoid the additional cost in post-processing the three different possible outputs, and similarly to what was done for the front part of the CC, an area delimitation strategy is performed. It is relevant to highlight that, also similar to what happens in the front part, the extraction of the NIF field complements the global extraction strategy in the CC.

Finally the MRZ field. This field represents huge importance in what is the validation and compensation of the total extraction. This is because, as mentioned before, the MRZ code, when well extracted and complete, can replace the need to extract the front part of the CC. From this code, we can sometimes extract part or all of the ID holder, the checksum-validated Número de Identificação Pessoal (NIP), and even the date of birth. For the complete extraction of the 91 digits that make up the MRZ (figure 6.4) two strategies were taken into account, to preserve its extraction and processing. In a first step, similarly to what was done for the NIF and the fields in the front part of the CC, the MRZ field is delimited by applying the clipping to the original image (table 6.1).

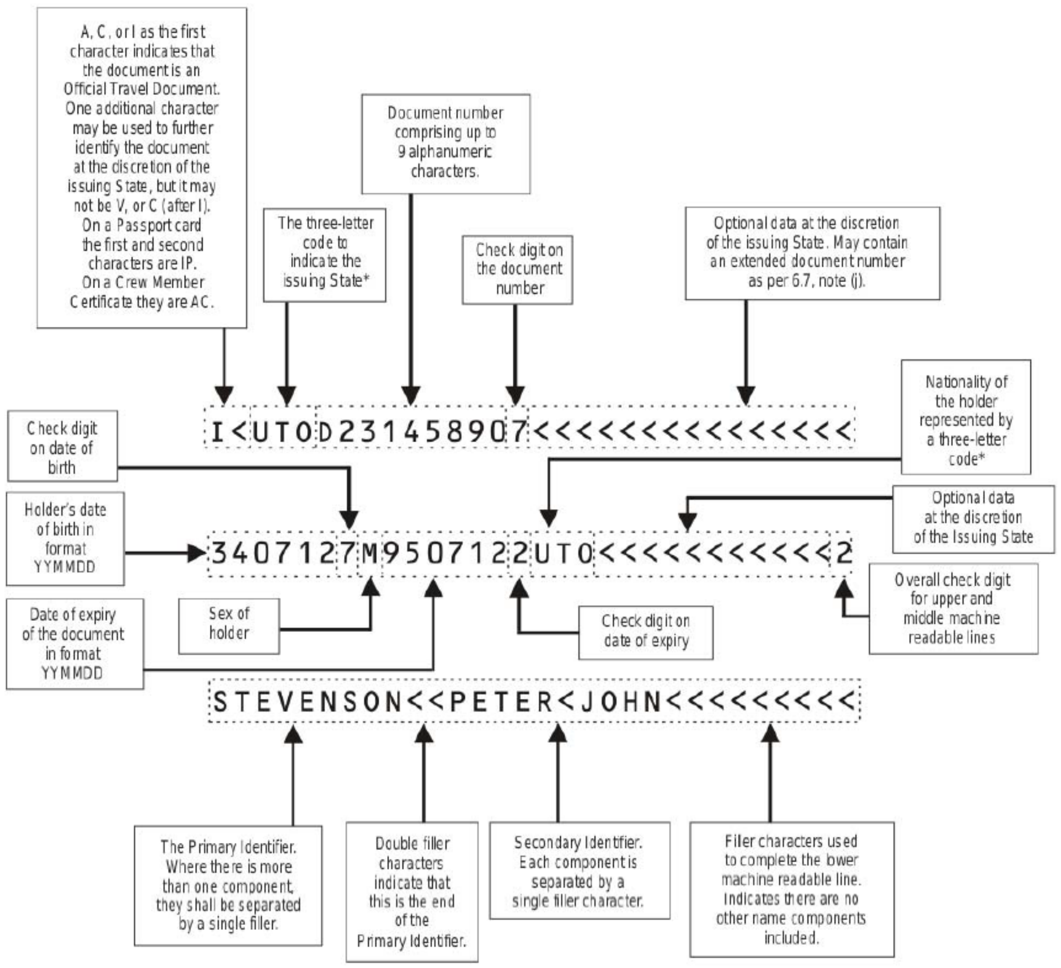

A, C, or I as the first character indicates that the document is an Official Travel Document. One additional character may be used to further identify the document at the discretion of the issuing State, but it may not be V, or C (after I). On a Passport card the first and second characters are IP. On a Crew Member Certificate they are AC.
The three-letter code to indicate the issuing State*
Document number comprising up to 9 alphanumeric characters.
Check digit on the document number
Optional data at the discretion of the issuing State. May contain an extended document number as per 6.7, note (j).
I<UTOD231458907<<<<<<<<<<<<<<<
Nationality of the holder represented by a three-letter code*
Check digit on date of birth
Holder's date of birth in format YYMMDD
3407127M9507122UTO<<<<<<<<<<<2
Optional data at the discretion of the Issuing State
Overall check digit for upper and middle machine readable lines
Date of expiry of the document in format YYMMDD
Sex of holder
Check digit on date of expiry
STEVENSON<<PETER<JOHN<<<<<<<<<
The Primary Identifier. Where there is more than one component, they shall be separated by a single filler.
Double filler characters indicate that this is the end of the Primary Identifier.
Secondary Identifier. Each component is separated by a single filler character.
Filler characters used to complete the lower machine readable line. Indicates there are no other name components included.

Figure 6.4: MRZ - Elements Description [89]

Based on the position and shape of the MRZ code, it is possible from the start to try to take advantage of its geometric arrangement on the back of the CC to extract it. That is, given that the three lines of characters that compose it are aligned and centered concerning the CC, they end up being bound by a rectangular shape that is quite noticeable at first sight. Putting this concept to the test, we can define based on some functions and methods of the OpenCV library [15] to extract and delimit the area referring to the MRZ code in an automated way and with great precision.

This strategy involves looking for contours within the back of the CC. These contours after processing will correspond to bold areas resulting from the application of a series of thresholding functions. Since we are looking for a rectangular shape in the ID, initially a rectangular and square kernel structure is built and after binary conversions in the ID, it is subjected to a search for dark or high contrast areas in the ID that may correspond to rectangular structures. After later collecting the contours resulting from this search, the function $cv2.findcontours$ is applied to extract the BB coordinate list of each of the detected rectangular contours, and which, potentially may correspond to that of an MRZ code.

Considering that several contours like those in the NIF field can arise, two decision criteria need to be defined for matching an MRZ contour. A first parameter is the value of A_r (equation 6.2) which gives us the quotient between the length and height of the detected contour and a second parameter, $crWidth$ (equation 6.2) which represents the quotient between the length of the jagged contour and the total length of the back of the CC.

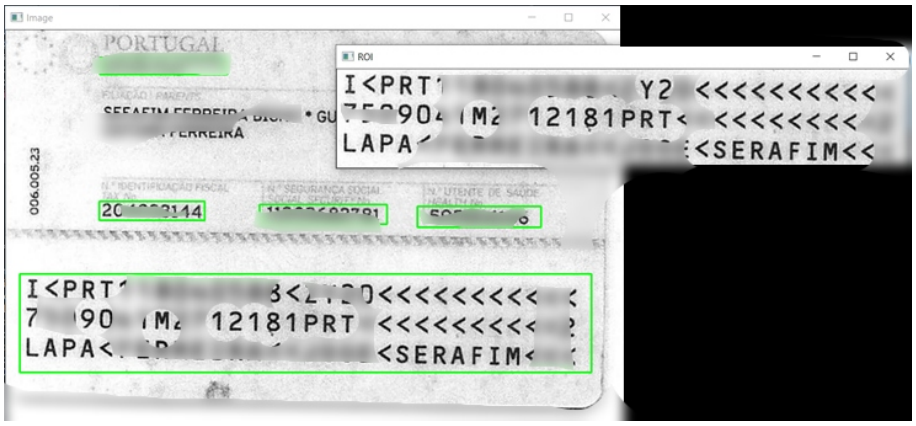
$$A_r = \frac{w_{contour}}{h_{contour}} \tag{6.1}$$

$$crWidth = \frac{w_{contour}}{w_{ID}} \tag{6.2}$$

For detection of MRZ-type contours, the value of A_r must be greater than 7 and the value of $crWidth$ must be greater than 0.7. The applicability of this strategy can be seen in figure 6.5.

Once the contour corresponding to the MRZ code has been detected, its coordinates are used to correctly cut the MRZ code from the entire backside of the CC.

It is important to say that within these strategies is implicit the correction or not of the fields delimited by the field delimitation strategy (table 6.1). For the front end, it becomes for example harmful to correct the orientation after the clipping, because it may result in a bad reading of the deskew algorithm considering the reduced character area and its consequent orientation. In the case of the MRZ code, it is imperative to have this post-processing since it is a field of a very fragile nature, where any misreading, however minimal, may result in incorrect data extraction.



I<PRT1[redacted]Y2<<<<<<<<<<
7[redacted]90[redacted]M[redacted]12181PRT<[redacted]<<<<<<<<
LAPA<[redacted]<SERAFIM<<

I<PRT1[redacted]3<[redacted]0<<<<<<<<<[redacted]
7[redacted]90[redacted]M[redacted]12181PRT[redacted]<<<<<<<<<[redacted]
LAPA<[redacted]<SERAFIM<[redacted]

Figure 6.5: MRZ extraction results using contour approach with OpenCV methods

CC PART	FIELD	Height Crop Interval (%)	Width Crop Interval (%)	Allowlist	Apply OSD	Resize (width)
CC_{front}	Name	[18,35]	[15-100]	ÃÁÀÂABCDEÉ ÊFGHIÍÎÌÇJKLMN OÓÒÔÕPQR STUÚÙVXYZ'-	No	200
	Birth Date	[52-69]	[50-80]	" 0123456789"	No	100
	NIP	[69-79]	[15-60]	"0123456789"	No	100
CC_{front}	NIF	[35-60]	[6-40]	"0123456789"	Yes	75
	NSS	[35-60]	[40-70]	"0123456789"	-	-
	NSU	[35-60]	[70-90]	"0123456789"	-	-
	MRZ	[60-100]	[0-100]	"ABCDEFGHIJ KLMNOP QRSTUVXYZ 0123456789<"	Yes	700

Table 6.1: Delimitation Area Strategy - Parameter's value in function of the field's nature

6.4 Post-processing

Once we know how the extraction of the different fields within the CC is performed, we still need to know how it is possible to pair the information collected from both parts of the CC, when both are detected and processed by the OCR engine.

Internally, the pipeline has a data structure in dictionary format that contains as a key the different files that enter the pipeline. Each key, that is, each FileNet, has an array with the different extractions achieved in the previous phases of this process. For each extracted part of a CC, a flag is placed indicating which part corresponds to the extraction, followed by the textual values extracted by the OCR and their respective confidence and validation flags. In the last step, when the total extraction within the pipeline for the target FileNet comes to an end, it is necessary to search and pair within this dictionary the different pairs of matching CC parts. This task is made possible by searching for the similarity of the extracted fields in both parts. First, the separation is made within the dictionary between the front and back parts. After this, the logic to apply has as a criterion the number of CCs present in the document.

In a first approach, in case there is more than one CC present in the FileNet in question, it is necessary to iterate simultaneously over both data structures that store the information about the back and front parts stored in the original dictionary, and if a certain percentage of similarity occurs between the content present in one of the detected front parts and one of the detected back parts a match is recorded in the final array of CC pairs, which will then be used to process and compile all the extracted joint information. In terms of similarity criteria for the different fields to be extracted, the Name field is used as the pairing criteria when the similarity between both parts is greater than or equal to 60%. For the numeric fields NIP and NIF, the matching criterion is greater than or equal to 80%.

In the case where there are only one of the parts present in the FileNet, the extraction is still performed, since it is preserved within the logic implemented in the post-processing for this phase of the pipeline. If only the front part of the citizen card is detected, the NIP field is not returned. In the case of the back, if it occurs, it has the ability, as previously mentioned, to be able to return all or part of the data (in the case of the Name field, it may not be complete, depending on its total length).

Taking again the causal approach, where the correct detection of both parts of one or more CCs within FileNet occurs, it is then necessary to proceed to choose between the two captured values for the name, NIP, and date of birth fields. For the selection criteria of the best final output for each field, it is important to refer to the extraction confidence associated with the character value and, in addition, the composition of the field according to the expected character format. In addition to these criteria, the use of validation on fields such as date of birth, NIP, and NIF makes the selection criteria easy. For the different fields mentioned there is a way to validate, so it is necessary to separately build functions that can validate them.

In the case of the date of birth, validation is done using an auxiliary tool datetime, which is a Python library that helps to manipulate date formatted variables. For the other numeric values, both the NIP and the NIF are validated using checksum functions, that is, functions that sum the elements of a code or number and validate it based on a final value defined in the checksum digit. In the case of the NIF, the validation operation is quite simple and only includes the steps specified in the Algorithm 1.

Algorithm 1 Algorithm for NIF validation

```
 1: procedure Validate NIF
 2:     len ← length of NIF
 3:     VALID_NIF ← False
 4:     DIGIT_SUM ← 0
 5:     EXPECTED_DIGITS ← 9
 6:     if NIF is digit or len not equal to EXPECTED_DIGITS then
 7:         VALID_NIF ← True
 8:     end if
 9:     for i ← 0 to len do
10:         DIGIT_SUM ← NIF[i] * (EXPECTED_DIGITS - i)
11:     end for
12:     DIV_REST ← DIGIT_SUM mod 11
13:     if NIF[-1] equal to 0 and DIV_REST equal to 1 then
14:         DIV_REST ← (DIGIT_SUM + 10) mod 11
15:     end if
16:     if DIV_REST equals to 0 then
17:         VALID_NIF ← True
18:     else
19:         VALID_NIF ← False
20:     end if
         return VALID_NIF
21: end procedure
```

Finally, in the case of the NIP field, the validation formula performs the verification using the complete document number. The NIP consists of four separate elements and is visible on the front of the CC. In 6.3 we can see the format of NIP and the meaning of each digit.

$$DDDDDDDDCAAT \tag{6.3}$$

Where,

- D is a digit of NIP , $D \leftarrow [0, 9]$
- C - Check Digit of NIP , $C \leftarrow [0, 9]$
- A - Version , $A \leftarrow [A - Z, 0 - 9]$

- T - Check Digit Document Number ,$T \leftarrow [0, 9]$

To be validated, the NIP must pass the following tests:

1. Performing a right-to-left count of the document number, duplicating the value of each second element, with the letters to be replaced according to the conversion table provided;
2. If the result of the duplication is equal to or more than 10, subtract 9 from the value;
3. Comprehend the whole of the obtained values;
4. With the obtained value, divide the remainder by ten, with the result being that if the result is zero, the document number is valid.

As an example, and using **00000000 0 ZZ4** as the all NIP information extracted, we start by removing the existing spaces. Next, the letters should be replaced by their respective value, and for this case, we would have the value **000000000(35)(35)4**. Once this primary conversion phase is over, the application of the previous steps would result in the final validation inference through the following steps:

- $(0{\times}2) = 0$, $(0{\times}2) = 0$, $(0{\times}2) = 0$, $(0{\times}2) = 0$, $(0{\times}2) = 0$, $((35{\times}2) = 70 - 9) = 61$
- $(0) + 0 + (0) + 0 + (0) + 0 + (0) + 0 + (0) + 35 + (61) + 4 = 100$
- $100 \bmod 10 = 0$, which means that the NIP is valid.

This page is intentionally left blank.

Chapter 7

maryHeaven

Throughout the last three chapters, the strategies and methodologies adopted and studied were discussed to try the best way to correct and solve the problem imposed by Altice/MEO in chapters 1 and 3. During this project, several efforts were made to achieve the ultimate success of extracting information from digitized documents. Not only in the task of detecting and collecting Identity Documents (IDs) but also in the efforts for the correct recognition and extraction of their fields.

As new strategies emerged, and these worked, others lost momentum. The final software took shape until it stabilized on the final version, which was named *maryHeaven*. These continuous efforts resulted in the improvement and progression of the good results of the proposed pipeline, and its consequent integration into the internal processes of Altice/MEO.

maryHeaven is the result of several Deep Learning (DL) approaches together with varied techniques and recent technologies of the Computer Vision (CV) world. Through its robust and integrated pipeline over its different phases and its integration based on multiprocessing and parallel computing, *maryHeaven* can perform the extraction of the personal and sensitive scanned information in seconds. Moreover, it makes use of different validation and post-processing techniques to ensure high-quality information delivery and extraction.

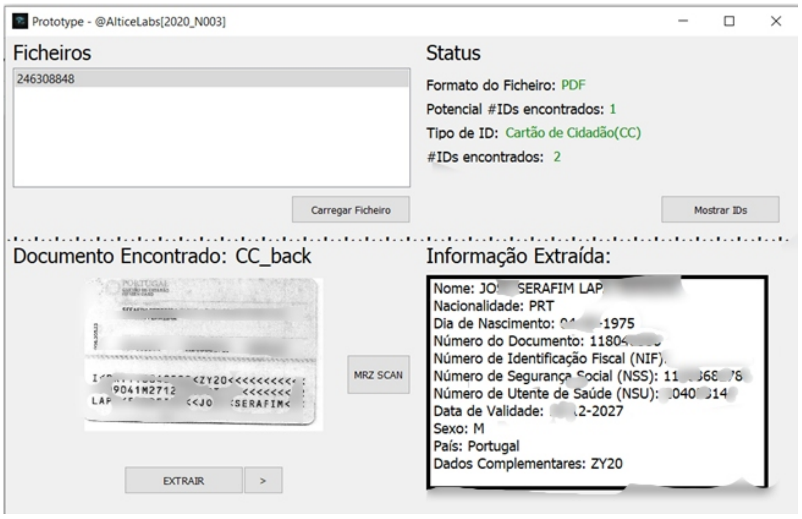
In this chapter, the path taken to arrive at the *maryHeaven* solution will be briefly explored, as well as its final architecture.

7.1 Prototyping

According to the scenarios that emerged and with which we worked within the Customer Relationship Management (CRM) environment of Altice/MEO, changes and strategies were taken to adapt the final solution as best as possible.

In a first approach and to exemplify the power of the pipeline understudy, the prototype with Graphical User Interface (GUI) was developed (figure 7.1). Initially, it served to show in presentations, for example, how the Altice/MEO problem could be solved and how the software acted in the different phases. This first approach was made in a Python environment using the PyQt5 graphical solution library.

As you can see from the figure 7.1, prototype 1.0 was composed of four quadrants. Each of these tries to represent a phase. In the first quadrant, there is a *"drag&drop"* area



246308848

Formato do Ficheiro: PDF
Potencial #IDs encontrados: 1
Tipo de ID: Cartão de Cidadão(CC)
#IDs encontrados: 2

Nome: JO[redacted] SERAFIM LAP[redacted]
Nacionalidade: PRT
Dia de Nascimento: [redacted]-1975
Número do Documento: 11804[redacted]
Número de Identificação Fiscal (NIF)[redacted]
Número de Segurança Social (NSS): 11[redacted]
Número de Utente de Saúde (NSU): [redacted]
Data de Validade: [redacted]2-2027
Sexo: M
País: Portugal
Dados Complementares: ZY20

Figure 7.1: *maryHeaven* - Prototype 1.0

where the user can upload types of PDF, image, or compressed file. Once all the desired FileNet documents are listed, we can then proceed to extract them individually. To do this, we will select a FileNet from the list in the first quadrant and press the "Show IDs" button. This button is in the quadrant referring to the processed statistics of the extraction performed on the target FileNet document. When extraction of the IDs is done at this stage, information such as the file type and the number of Cartão de Cidadãos (CCs) found is shown in the "Status" quadrant.

The purpose of this first approach would be more to demonstrate the power that phase 0 has on the good results of the project. Once the extraction is done, in the third quadrant it is shown in "slideshow" mode, the clippings obtained from the different IDs detected in the FileNet document. Here the user could select from the IDs present in the slideshow, one by one, to perform their extraction. In addition to this, the contouring solution for the Machine Readable Zone (MRZ) code would have already been implemented, so that even the prototype itself included a button just for performing this task. Once we had chosen a part of CC that corresponded to a back part, by pressing the "MRZ SCAN" button we would be able, besides isolating the MRZ code, to extract and process its contents cleanly.

The internal structure of this prototype was simple. Its phase 0 was composed of a Faster Region Based Convolutional Neural Network (R-CNN) model (MatLab Engine [67]) trained with about 800 images of annotated CCs. At that time, the proposed preprocessing pipeline was not yet, implemented. Only simple preprocessing functions were applied, such as the application of local and global thresholding. For the next phase, the implemented logic was still based on the general extraction of the characters from the ID without applying the two strategies documented in chapter 6. The built-in Optical Character Recognition (OCR) engine was Tesseract.

Despite the success and the proven potentiality, this prototype was far from being a viable final solution, not only because it did not comprise a solution that could be integrated within the internal processes of Altice/MEO, but also because it was not built to process several sources of information at the same time, i.e, was not robust enough for the needs of the company. Either way, the concept of the pipeline proposed at the beginning of the project was proven. It was proven with this prototype that phase 0 represents a determining role in the quantity and quality of extracted information, as well as the detection and extraction of the content present in the MRZ code, is an asset for the quality of the

RISK ID	Description	LEVEL OF RISK (1-5)
0	Lack of mix class data to improve the DataSets content and feature extraction	2
1	Not being able to make good Models with the Cloud Platform solution, and only have good results with MatLab	1
2	Model used in the prototype is too weak to handle the scenarios complexity imposed in the project context	4
3	Phase 0 model (R-CNN) too slow and compromises the good functionality of all program (prototype)	5
4	Although IDs are found, there is sometimes misclassification of elements within the document that are not considered "ID" type objects	4
5	There are BBs that for the front side of CC do not do their calculation well, but for the back side no problems have arisen	4
6	MRZ scan is too sensitive to different sizes	3

Table 7.1: Table of implementation risks raised from the implementation of Prototype 1

information obtained (figure 7.1). Besides this, from the several problems detected in this prototype, we highlight the increased risks of the implementation of this first solution registered in the table 7.1.

Once some of the problems to the implementation of the prototype had been detected and isolated, the idea would be to continue and further improve the pipeline solution found, for this purpose focusing efforts on the risks and difficulties highlighted.

Prototype 2.0, similarly to its ancestor had GUI. The interface is much more detailed than the first version, where it is possible to see that there is less freedom of action on the user's side since this approach only has two buttons and one mandatory field, the output directory. And here is the biggest difference. As the project and solution progress, the closer we need to be to the interests of our client, Altice/MEO. It is then necessary to already start thinking about how to return the data extracted from the process. In the case of the 2.0 prototype, the user must first put an output directory in the input box, because this is where the extraction results will be saved in text file format (figure 7.2).

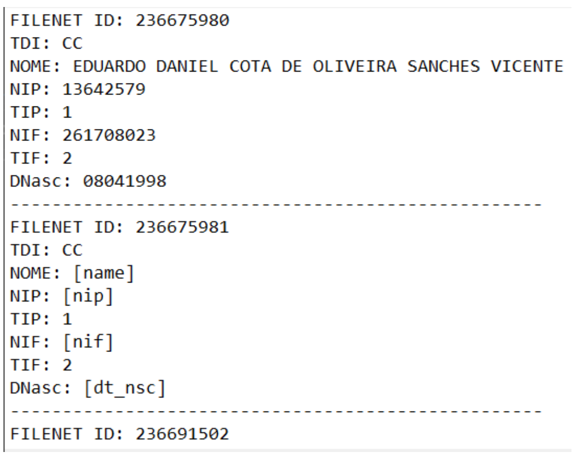

```
FILENET ID: 236675980
TDI: CC
NOME: EDUARDO DANIEL COTA DE OLIVEIRA SANCHES VICENTE
NIP: 13642579
TIP: 1
NIF: 261708023
TIF: 2
DNasc: 08041998
--------------------------------------------------
FILENET ID: 236675981
TDI: CC
NOME: [name]
NIP: [nip]
TIP: 1
NIF: [nif]
TIF: 2
DNasc: [dt_nsc]
--------------------------------------------------
FILENET ID: 236691502
```

Figure 7.2: *maryHeaven* - Output formatting (*Output.txt*)

Continuing to list the significant differences between one prototype and the other, we can note that prototype 2.0 is much more oriented towards the metric evaluation of results rather than the visualization of internal processes, unlike the previous approach. This is due to the confidence already gained in the previous approach due to the solid gears that have been proven to exist between phases. Looking further at the external part, of what the GUI contemplates, it should be noted that about half of the interface area is occupied only by extraction classifying metrics, among them, the trust of the extracted IDs and the individual trusts of the various fields that compose it. Besides this, relevant data, such as the number of corrupted files or the total execution time, came to be useful to mark the beginning of a journey of increasing learning and growth. Additionally, the physical separation that exists between the statistics of the different pipeline phases in the GUI allows for a separate systematic evaluation of each part.

Although listed, the IDs of the Autorização de Residência (AR), Identificação Fiscal (IF) and Bilhete de Identidade (BI) classes were not tested in any of the tools developed throughout this project. As already mentioned, only the theoretical survey of the possibility of adding multiclass to ID detection in scanned documents was done.

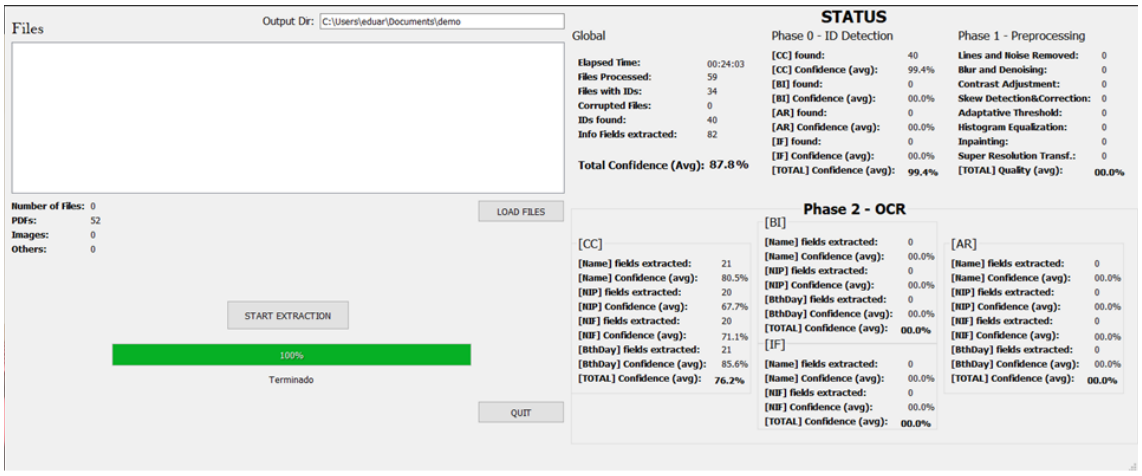
Internally, prototype 2.0 was built to be as close as possible to the feasible solution for Altice/MEO. Unlike what happened in the prototype 1.0 (figure 7.1), where it was only possible to process one FileNet file and one ID at a time, this new approach can process several FileNet files simultaneously with the support of multiprocessing oriented programming and parallel computing with the help of a Thread Pool, as well as being able to process them independently. These are the most significant changes in the prototype. This new way will not only increase the rate of processed FileNet files but also optimize the entire process of extracting and processing scanned information.

In phase 0 for the extraction of IDs, and taking into account the risks and problems raised earlier, the integration of a You Only Look Once (YOLO)v3 model with a dataset of only 1500 annotations was done. In addition, the application of strategies such as line removal and glare removal in phase 1 were integrated into this solution. As for phase 2, this prototype was set up with the possibility of using the two engines referenced earlier in the chapter 6, the Tesseract engine [105], and the EasyOCR engine [47]. From here we can immediately admit that the 2.0 prototype was built to improve not only the performance of the desired solution but also to improve and explore good strategies for phase 2 of the pipeline.

It should be noted that, in a first analysis, the integration of EasyOCR was an added value, because thanks to its computational power with the aid of Graphics Processing Unit (GPU) computing, it was possible to integrate this solution side by side with Tesseract at minimal cost to assist its extraction, thus playing the role of backup OCR engine. However, further testing needed be done to determine which of these was the best solution to integrate into the final pipeline.

Overall, in terms of post-processing little had yet been achieved. At this stage, it was just a matter of investigating the average reliability achieved by the two implemented OCR engines, and then later investigating the best strategies for improving the quality of the output data. Despite this, the pair matching logic associated with the similarity matching part of the CCs had already been implemented, which shows that at this stage, the approach created would already be very close to the final desirable solution.

Later on, the second version of this prototype would be created, prototype 2.1 (figure 7.3), where the only significant differences were in the way statistics and additional information were displayed in the GUI. Besides this, in this version, the Comma-separated values (CSV) output format was already contemplated. Where, in addition to the text file format, a CSV



```
Number of Files:  0
PDFs:             52
Images:           0
Others:           0
```

STATUS

Global

```
Elapsed Time:             00:24:03
Files Processed:          59
Files with IDs:           34
Corrupted Files:          0
IDs found:                40
Info Fields extracted:    82

Total Confidence (Avg):   87.8%
```

Phase 0 - ID Detection

```
[CC] found:                    40
[CC] Confidence (avg):         99.4%
[BI] found:                    0
[BI] Confidence (avg):         00.0%
[AR] found:                    0
[AR] Confidence (avg):         00.0%
[IF] found:                    0
[IF] Confidence (avg):         00.0%
[TOTAL] Confidence (avg):      99.4%
```

Phase 1 - Preprocessing

```
Lines and Noise Removed:       0
Blur and Denoising:            0
Contrast Adjustment:           0
Skew Detection&Correction:     0
Adaptative Threshold:          0
Histogram Equalization:        0
Inpainting:                    0
Super Resolution Transf.:      0
[TOTAL] Quality (avg):         00.0%
```

Phase 2 - OCR

[CC]

```
[Name] fields extracted:       21
[Name] Confidence (avg):       80.5%
[NIP] fields extracted:        20
[NIP] Confidence (avg):        67.7%
[NIF] fields extracted:        20
[NIF] Confidence (avg):        71.1%
[BthDay] fields extracted:     21
[BthDay] Confidence (avg):     85.6%
[TOTAL] Confidence (avg):      76.2%
```

[BI]

```
[Name] fields extracted:       0
[Name] Confidence (avg):       00.0%
[NIP] fields extracted:        0
[NIP] Confidence (avg):        00.0%
[BthDay] fields extracted:     0
[BthDay] Confidence (avg):     00.0%
[TOTAL] Confidence (avg):      00.0%
```

[IF]

```
[Name] fields extracted:       0
[Name] Confidence (avg):       00.0%
[NIF] fields extracted:        0
[NIF] Confidence (avg):        00.0%
[TOTAL] Confidence (avg):      00.0%
```

[AR]

```
[Name] fields extracted:       0
[Name] Confidence (avg):       00.0%
[NIP] fields extracted:        0
[NIP] Confidence (avg):        00.0%
[NIF] fields extracted:        0
[NIF] Confidence (avg):        00.0%
[BthDay] fields extracted:     0
[BthDay] Confidence (avg):     00.0%
[TOTAL] Confidence (avg):      00.0%
```

Figure 7.3: Prototype 2.1 - One step closer to *maryHeaven*

file was stored in the directory inserted by the user, with the exact content as in figure 7.2.

It was considered that the arrangement of the information in CSV format was an added value to not only save the relevant information for data extraction but also to save additional information that can often compensate for some of the information that is returned in post-processing. This is safeguarded because, for example, in addition to the four fields expected for output, data such as the back and frontlist data, the entire MRZ code, and the confidence of each field is stored in a CSV file.

The strategies described for phase 2, where the application of character area delimitation among other techniques is done, is established in version 3.0. Prototype 3.0 uses the same graphical support as version 2.1 (figure 7.3), only changing internally. In addition to the integrated phase 2 strategies mentioned above, validation functions for the fields Número de Identificação Pessoal (NIP), Número de Identificação Fiscal (NIF), and date of birth were also integrated to improve the quality of the output data as described in chapter 6. For the name, the efforts applied were based on the attempt to apply an automatic correction method for Portuguese words, based on an application of Peter Norvig's, the Spellchecker algorithm [74]. The idea was to search the dictionary with a Knowledge Data Base of Portuguese first names and surnames. In addition to the post-processing efforts, a text file named "blacklist" was added to the prototype's directory, which provides a list of the different terms and fields that are not interesting for extraction, such as, the headers of the fields, or other common features in all CCs.

The prototype, which comes from the Greek words proto (first) and typos (form), is nothing more than a low-fidelity or high-fidelity sketch of a project, with the main goal of validating the functions and requirements of what is being developed to align the customer's expectations and market needs with the final product that will be launched. In the context of this project, it is considered that the course of assembly and improvement of the various prototypes had a huge impact on what was the easy and quick implementation of the final software in a remote environment for the use of Altice/MEO. The relevance that each version had in the growth and better perception of the problems and strategies to adopt, became impactful in the achieved final result. In a futuristic vision, the development of the prototyping stage helped draw new perceptual lines that helped the implementation of new and improved processes to the final intelligent system, *maryHeaven*.

This page is intentionally left blank.

Chapter 8

Results

After the survey of the methodologies and strategies to be adopted for implementation in the final pipeline, we will now observe, report and discuss the different results obtained in the various phases of the project. Each one was reviewed and we adopted the needed strategies to consolidate a stable performance of the solution in a global prism. In each of them, evaluation metrics were used to guide us in choosing the best solutions found during the tests performed under the construction phase. We must also point that, all the tests carried out were subject to the limitation of a relatively non-uniform and small dataset.

8.1 Phase 0 - Object Detection Results

As was mentioned several times in chapter 4 about Phase 0, we have several different models and training environments at our disposal. Within the available methods for creating and training Object Detection models, we can subdivide these into: the one-stage methods and the two-stage methods.

One-stage detectors (You Only Look Once (YOLO) family, Single-Shot Detector (SSD) models, EfficientDets, and CenterNets) offer fast inference speeds, whereas two-stage detectors (Region Based Convolutional Neural Network (R-CNN) family) have excellent localization and recognition accuracy but are also the slowest bibliometric. This phase will be the only tool for extracting the various Identity Documents (IDs) that can be found in FileNets in the context of this project. They are classified, or labeled, based on their position, isolation in the context of the inserted scenario, and subsequent categorization. It is important to stress that selecting the most appropriate model is critical since this phase, as the first and most important, will have a significant impact on the rest of the process. Also, because this phase, as the first and most important, will influence all of the output values of the subsequent pipeline phases, it is necessary to perform all types of parametric evaluations to the training performed with the chosen dataset and the metrics used to evaluate their performance and accuracy in a digitized information process.

Within phase 0, as it was possible to observe, multiple types of object detection models were trained in two different training environments, the Darknet environment, and the Tensorflow environment.

In the Darknet environment, the following models were trained:

- YOLOv3-tiny

- YOLOv4-tiny
- YOLOv4
- YOLOv3
- YOLOv3-SPP
- YOLOv4-Scaled
- YOLOv4-CSP-SWISH
- YOLO-P5
- YOLO-P6
- CSRESNEXT50-SPP-PANET

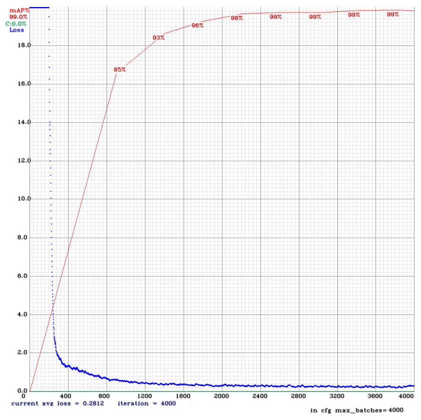
As we could see, only models from the YOLO family were trained in the Darknet environment. The remaining models, already specified in the table 4.2 in chapter 4, were trained in the Tensorflow environment. In the further subsections, we will analyze the results obtained by all these methods, not only in terms of their performance in the training phase but also, in the evaluation phase. To accomplish that were used tools and metrics available for each type of model and training environment occurred.

8.1.1 Training

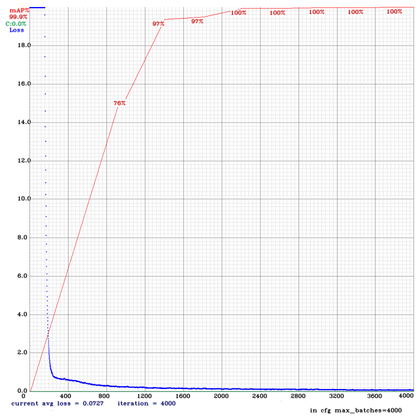
In the darknet environment, the evaluation metrics for models trained are based on the calculation of the loss and the Mean Average Precision (mAP) value over recursive iterations. Theoretically, a good object detection model will optimally have a high mAP value contrasted with a relatively low loss value when training the model. This is a strong indicator that there was a positive trade-off during model training. In addition to these metrics that are present in the graphs generated throughout the training in the Darknet environment, another relevant metric used in the evaluation and validation of the model is Intersection over Union (IoU). The IoU is used to measure the accuracy of the trained object detector, in terms of the location of the Bounding Boxs (BBs), it is, therefore, an optimal indicator in the context of our problem, because the priority above all, besides the correct classification, is the correct location of the ID.

As described, the dataset for the unique and exclusive detection of Cartão de Cidadão (CC) subclasses is composed of about 6514 annotated CCs. From the results obtained in the training phase in the Darknet environment, it is relevant to say that most of the models had an expected and positive behavior in terms of the trade-off between loss and mAP. Models such as the two tiny versions (figure 8.1), the YOLOv4 and YOLOv3 model (figure 8.2), and the models, $CSRESNEXT50 - SPP - PANET$ and YOLOv3-SPP (figure 8.3).

These models mentioned above had a relatively low average training loss and at the same time a percentage of mAP higher than 90% in all cases. At first glance, one could admit that there is overfitting. But this is not the case, because here, besides the fact that the dataset given to train has gone through a pipeline process of data augmentation, the Darknet environment itself does the second round of data augmentation as the training is performed every 1000 iterations, which, makes the possibility of overfitting decrease. Furthermore, methods such as dropout regularization or techniques like L1 and L2 Regularization help

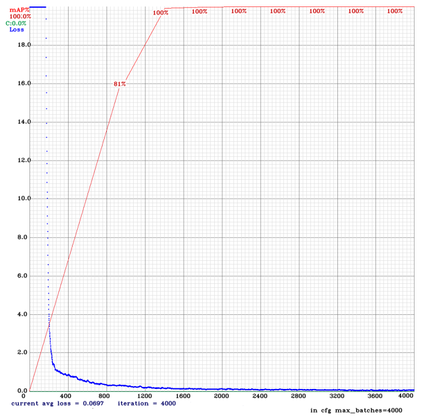


(a) YOLOv3-tiny

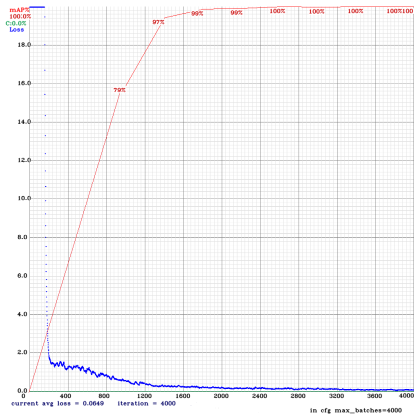


(b) YOLOv4-tiny

Figure 8.1: Loss and mAP plot results for the tiny versions of YOLOv3 and YOLOv4, respectively

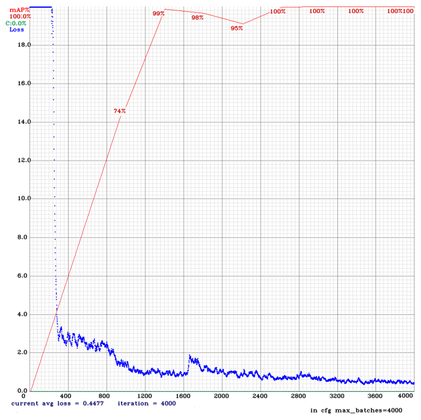


(a) YOLOv3

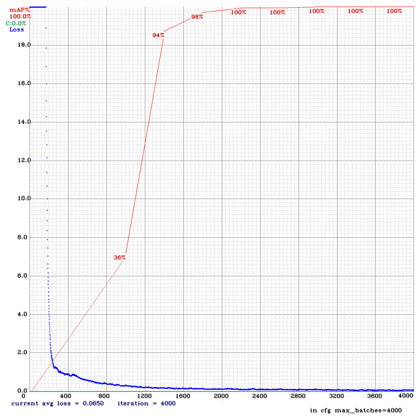


(b) YOLOv4

Figure 8.2: Loss and mAP plot results for YOLOv3 and YOLOv4, respectively



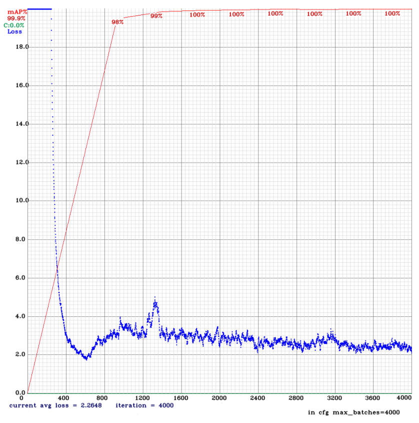
(a) $CSRESNEXT50 - SPP - PANET$



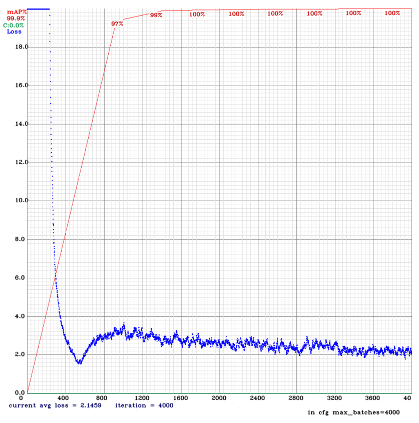
(b) YOLOv3-SPP

Figure 8.3: Loss and mAP plot results for $CSRESNEXT50 - SPP - PANET$ and YOLOv3-SPP, respectively

in the difficult task of not creating a final overfitted model. Somewhat different were the results for the YOLO P5 and P6 models (figure 8.4), and for the YOLOv4 Scaled and YOLOv4-CSP-SWISH (figure 8.5).

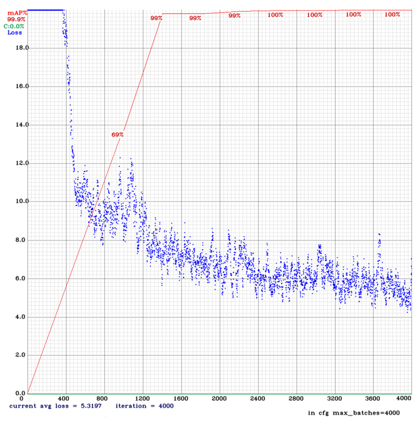


(a) YOLO-P5

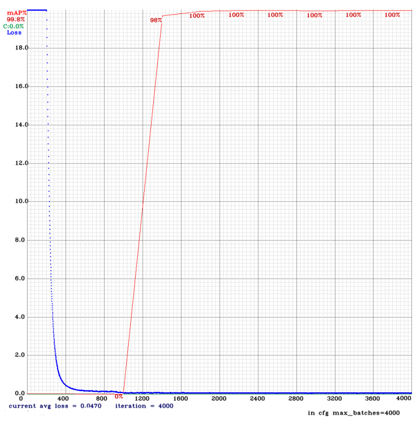


(b) YOLO-P6

Figure 8.4: Loss and mAP plot results for YOLO-P5 and YOLO-P6, respectively



(a) YOLOv4-Scaled



(b) YOLOv4-CSP-SWISH

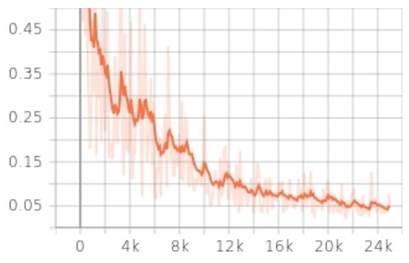
Figure 8.5: Loss and mAP plot results for YOLOv4-Scaled and YOLOv4-CSP-SWISH, respectively

In the case of the models mentioned above, the training results were a little more negative in relation to the others set as an example, taking into account what is expected of an object detection model. From the analysis made to these graphs it is possible to conclude that, in general, they present a higher loss value during training and also, in some cases, large and recurrent oscillations of this value during training, as it happens, for example, in the plot 8.5a. Despite these less positive results it is still necessary to validate the training, doing tests with scenarios from a validation dataset consisting of novel annotations to the model under test, in order to get the most accurate results possible.

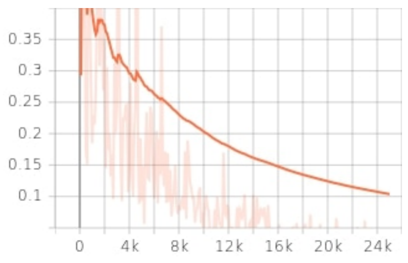
Despite thus, **from the training performed in the Darknet environment with the primary dataset only intended for the detection of CC parts, the results were very similar for all models except for a few cases that deviate a little from the expected rule.**

In the Tensorflow environment, the training conditions change slightly. Unlike in Darknet, where we only had access to a plot during and after training the YOLO models, in Tensorflow it is possible, with the help of the Tensorboard tool, to collect more relevant information regarding the training of the models. When we run the model for training, we can simultaneously run Tensorboard on the path to the model and Tensorflow, as new data (scalars) comes in, the listening process sends it to the link specified by Tensorboard, where our workspace will be.

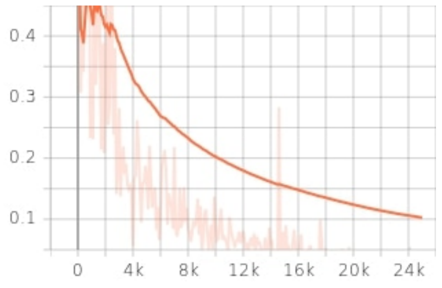
In the context of training in Tensorflow, it is more interesting to analyze the training results by dividing the models by the type of method they are. That is, divide the analysis between one-stage methods and two-stage methods. Regarding the two-stage methods, we have several different metrics at our disposal. In addition to the total loss value, there are then metrics such as the loss in classification and localization of BBs, the loss in object and localization for the Region Proposal Network (RPN), and also the regularization loss value (figure 8.6).



(a) Faster-R-CNN - ResNet50



(b) Faster-R-CNN - ResNet101

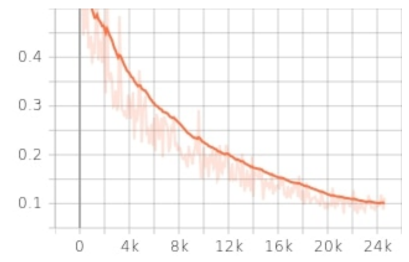


(c) Faster R-CNN - ResNet152

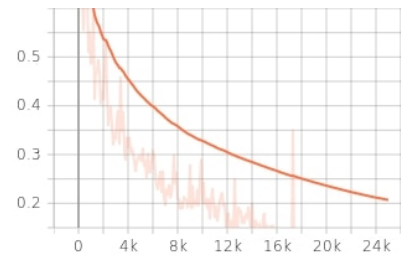
Figure 8.6: Total training loss plots for the Faster-R-CNN Models

From the analysis that can be done on the results of the training imposed in two-stage methods, it is important to emphasize that the notes taken in the theoretical part are proven. Within these models, the use of different backbone network topologies, as is the case in ResNet, also leads to a difference, between the training results for the three different versions implemented. It is therefore proven that a simpler network using a ResNet50 network as a backbone will have a significantly much lower total loss value compared to the other heavier and more complex networks, and of course, taking into account the nature and simplicity of the problem for CC class detection only.

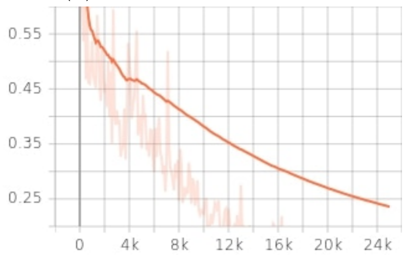
Then within the one-stage methods, the EfficientDet and SSD family models have the same training metrics: localization loss, classification loss, regularization loss and total loss. In figure 8.7 we can see the results for the SSD models. In reading them we can assume at the outset that overall, version 2 of the SSD network with a dimension of 640x640 obtained the lowest total loss value. When it comes to SSDs with ResNet as the backbone, these show quite positive results. Another particularity of the latter is also the fact that there is a constant increase in the total loss value as we increase the complexity and size of the backbone network (ResNet101, ResNet152). Within the EfficientDet family (figure 8.8) all trained models obtained optimal results and, in addition, the best of all the trainings performed in Tensorflow environment. In the three versions of EfficientDet trained, all of them had a total loss value below 0.2. Only one SSD was able to achieve results as optimal as the EfficentDet networks, all the others had results slightly above this so-called "optimal" value.



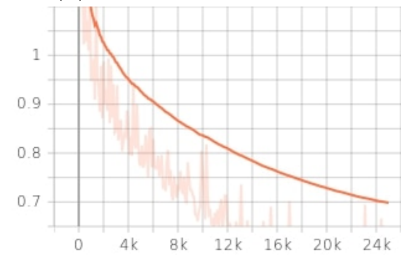
(a) SSD with ResNet50



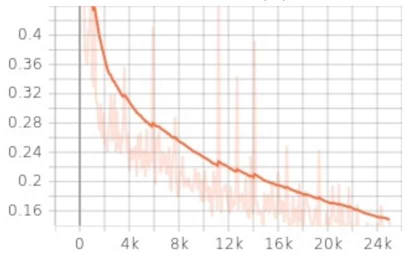
(b) SSD with ResNet101



(c) SSD with ResNet152

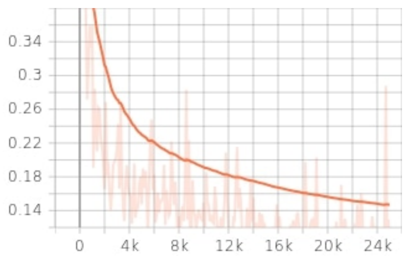


(d) SSD with MobileNet Version 1

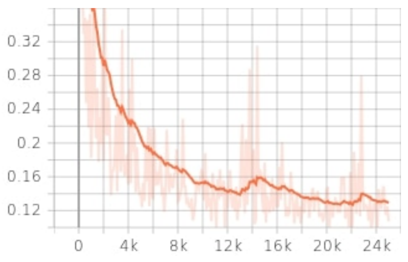


(e) SSD with MobileNet Version 2

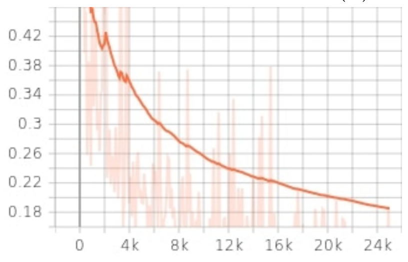
Figure 8.7: Total training loss plots for the SSD Models



(a) EfficientDet - D0



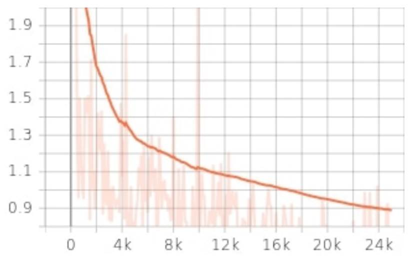
(b) EfficientDet - D1



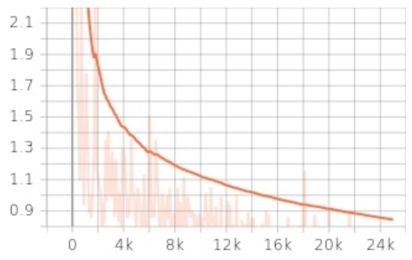
(c) EfficientDet - D2

Figure 8.8: Total training loss plots for the EfficientDet Models

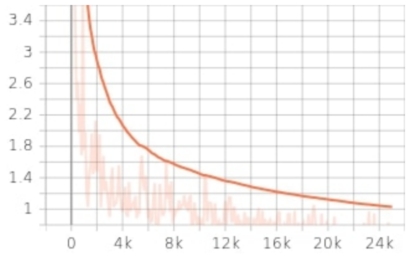
Finally, in training the CenterNet networks (figure 8.9) a total loss value above 0.8 was recorded for all trained models. Theoretically, this record will be a later indicator that the network does not have the most correct architecture for this task, and therefore will return results that are probably inadequate and out of context. This network has a very heavy and complex architecture and therefore, a bit like what happens in Faster R-CNN networks, it ends up not being so robust in inference time, and in addition to this, the fact that we are training with a network that is already configured to support multiple layers of convulsion with different sizes makes, not only the training process time consuming but also ineffective for the problem at hand.



(a) CenterNet with ResNet50



(b) CenterNet with ResNet101



(c) CenterNet with MobileNet Version 2

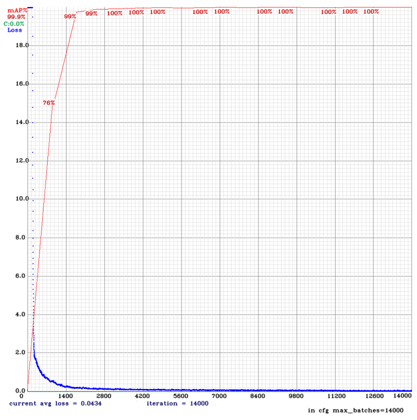
Figure 8.9: Total training loss plots for the CenterNet Models

Once the training results are known for the primary dataset, intended only for the detection of parts of CCs, it is now still necessary to know what results were obtained with the secondary dataset intended for the evaluation of the positive integration or not of a multiclass phase 0 mechanism. For this study, three trials were conducted with three models of the YOLO family (figure 8.10). Since this test is only an experiment to investigate the possibility of expanding phase 0 to a multiclass detection, the tests were reduced to the models that, in theoretical terms and from the analysis made to the previous trainings, will behave better. Therefore, the $CSRESNEXT50 - SPP - PANET$ (figure 8.10c), YOLOv3 (figure 8.10a), and YOLOv4 (figure 8.10b) models were trained with the multiclass dataset and had quite acceptable results overall.

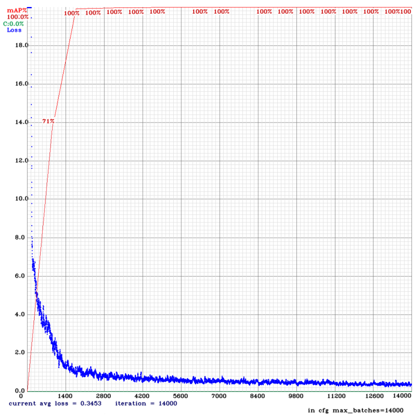
In terms of growth in mAP, the rate followed a normal trajectory concerning the reduction of the total loss value. In the case of training with the YOLOv3 model, it should be noted that through more careful observation we can see that the final stable total loss value is too small considering the complexity presented by the dataset at hand. Hence, it is necessary to first decide by applying the trained model in several scenarios, whether or not the trained model is stable or needs a better understanding of the needs underlying the type of losses that affected them the most. These may include a greater loss associated with the location of the object or its classification, or be associated with the application of the learning factor in early iterations, or even associated with the loss in the regularization value.

8.1.2 Evaluation

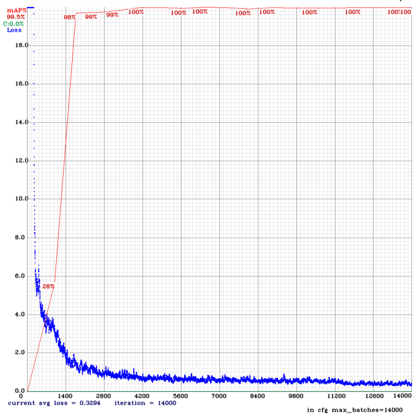
Object detection models are difficult to evaluate since each image might contain a large number of items, each of which can belong to multiple classes. This implies we'll need a



(a) YOLOv3



(b) YOLOv4



(c) $CSRESNEXT50 - SPP - PANET$

Figure 8.10: Loss and mAP plot results for the multiclass dataset

mechanism to check if the model discovered all of the objects, as well as a way to validate that the items identified are of the right class. This means that an object detection model must do two things: first, discover all of the items in a picture, and second, determine whether the objects detected are of the right class. We utilize the IoU as a similarity metric to see if an item was found. It's calculated by dividing the overlap's area by the combined size of the two bounding boxes.

The ranking of the bounding boxes each model produces is used to solve this problem. The Average Precision (AP) measure is calculated in this manner. We begin by sorting the probability scores and then use an IoU threshold to determine if the bounding box is True Positive or False Positive. Further, we count the accumulated TP and the accumulated FP and compute the precision/recall at each line. These values will then be used to calculate the f1-score for each model that has been trained.

To evaluate and validate the previously trained models, for the dataset intended for CC detection, an evaluation test was performed on about 1200 documents. This set of documents may contain examples of FileNets with or without CCs inside. Furthermore, a whole different variety of complex or non-complex scenarios may exist within it. Overall, the validation dataset has been fabricated to be a good ground truth (table 8.1) for the problem at hand.

Before any analysis, it must be said that not all models have reached this stage. For example, for some of the cases of the models trained in the Darknet environment. This is because several problems were detected at the time of their evaluation, namely, for example,

SUBCLASS	DETECTIONS
CC_back	514
CC_front	489
TOTAL	**1003**

Table 8.1: Evaluation Ground-Truth

the YOLO P5 and P6 models, and also models such as YOLO-Scaled and YOLO-CSP-SWISH. As for the approach referenced in the Tensorflow environment, all models were successfully tested.

Given the size and complexity of the result tables obtained for both environments, they have been placed in Appendix A for the YOLO networks trained in the Darknet environment and Appendix B for the remaining networks trained in the Tensorflow environment.

Considering what was said at the beginning of this subsection, and by extracting all the specific evaluative metrics on all detections performed on each model in the tables that are present in Appendices A and B, it was possible, in the end, to express the values for accuracy, precision, recall, f1-score and specificity of all the models trained for the detection of CC-type objects. Therefore, from what are the results in Annexes A and B, the summary of the information taken from these can be seen in the table 8.2.

TRAIN ENV.	MODEL NAME	MODEL LOADING TIME (s)	PRECISION	ACCURACY	RECALL	F1-SCORE	SPECIFICITY
DARKNET	YOLOv3-TINY	0,356	0,810	0,643	0,678	0,738	0,544
	YOLOv4-TINY	**0,251**	0,926	0,877	0,906	0,916	0,792
	YOLOv3	1,910	0,991	0,967	0,964	0,977	0,974
	CSRESNEXT50-PANET-SPP	1,736	**0,996**	**0,989**	**0,989**	**0,992**	**0,989**
	YOLOv3-SPP	1,851	0,982	0,939	0,935	0,958	0,952
	YOLOv4	2,069	0,986	0,977	0,983	0,985	0,960
TENSORFLOW	SSD_RESNET50_V1_FPN	56,504	0,996	0,974	0,969	0,982	0,989
	SSD_RESNET101_V1_FPN	84,391	0,996	0,973	0,968	0,982	0,989
	SSD_RESNET152_V1_FPN	122,353	0,994	0,969	0,964	0,979	0,983
	SSD_Mobilenet_V1_FPN_640x640	**27,968**	0,995	0,973	0,969	0,982	0,986
	SSD_Mobilenet_V2_FPNLite_640x640	44,532	0,993	0,944	0,931	0,961	0,980
	FASTER_R-CNN_ResNet152	62,168	**1,000**	**0,999**	**0,998**	**0,999**	**1,000**
	FASTER_R-CNN_ResNet101	47,986	**1,000**	**0,999**	**0,998**	**0,999**	**1,000**
	FASTER_R-CNN_ResNet50	32,170	**1,000**	**0,999**	**0,998**	**0,999**	**1,000**
	EfficientDet_D0_TRAIN	110,611	0,950	0,864	0,862	0,904	0,869
	EfficientDet_D1_TRAIN	142,161	0,982	0,948	0,947	0,964	0,952
	EfficientDet_D2_TRAIN	149,851	0,994	0,970	0,965	0,979	0,983
	CenterNet_ResNet101_V1_FPN_TRAIN	97,582	0,994	0,970	0,966	0,980	0,983
	CenterNet_ResNet50_V1_FPN_TRAIN	59,260	0,992	0,959	0,953	0,972	0,977
	CenterNet_Mobilenet_V2_FPN_TRAIN	58,824	0,979	0,943	0,943	0,961	0,943

Table 8.2: Final evaluation results of the models trained in both environments.

As we can see from table 8.2, the results have been divided into two parts, just as was done in training. The separation results in two very distinct groups in terms of size. What had already been discussed concerning the environments is that there is a huge disparity in terms of performance from one to the other, as had been previously analyzed in theoretical context within this project. This huge difference that is said to exist between one environment and another is due to the way they are internally programmed. On one side we have the Darknet environment, written in C and Compute Unified Device Architecture (CUDA), which makes it very powerful in terms of inference time of the models it builds. On the other hand, the models created in the Tensorflow environment have the negative aspect of being trained on the foundations of an Application Programming Interface (API), which is not built in an optimized way. Right from the start, it would be expected that much better results in terms of average inference time would be obtained for models trained in the darknet environment than for models trained in Tensorflow. Something that can be proved by the time it takes for the models to be loaded into memory, in this case, when in phase 0 it is loaded to start extracting IDs from the different FileNets it receives. This difference, as said, is easily seen when we look at the third column of data, relative to times (table 8.2). Its visible right from the start that the loading times for the models trained on the Darknet are much lower than the average loading times for the models created on Tensorflow. Of all the models trained on Tensorflow, none of these obtained loading times better than 30 seconds, except for the only isolated case, the SSD Mobilenet version 1 network. Moreover, looking at the results achieved in the Darknet environment with the YOLO networks, all of them achieved loading times below 3 seconds, which is exceptional. What is more, this also proves the theories raised earlier. In terms of classification, all models presented quite acceptable values, both in the Darknet and Tensorflow environments.

In the darknet environment we can immediately point the spotlight on the $CSRESNEXT50-PANET-SPP$ model, which was the only model, within this environment, to achieve the maximum value in all fields. Behind it, side by side were the YOLOv3 and YOLOv4 models, which also achieved very acceptable results. From the analysis of the best f1-score and specificity values within the Darknet environment, it is necessary to point out that, although the $CSRESNEXT50-PANET-SPP$ model has reached the best values, this does not put aside the use or not of the other networks, which also had excellent results.

On the other side, in the Tensorflow environment, the results were not the best. This is because, in addition to the results already explored for the inference and model loading times in phase 0 of the pipeline, there was still the training of models that most likely have to overfit. This is visible for example in the Faster R-CNN networks, where these, in all parameters achieved maximum values, and, more specifically in specificity, obtained values around 100%. This was also possible to verify at the time of training, where these and other possibilities were raised. Focusing a bit on the positive part, of the models trained in the Tensorflow, it is necessary to highlight the high utilization of the SSD networks. More specifically, both versions of the models that use the Mobilenet network as a backbone obtained results well above average compared to the other models trained on Tensorflow, namely, regarding the positive trade-off that exists between the loading time, the specificity value, and f1-score. Besides these two, the SSD network with ResNet50 as backbone also behaved overall quite well in what was the network training in the Tensorflow environment. Despite this small local aspect, overall, the training of the models in Tensorflow obtained satisfactory results but was far from the optimum achieved in the Darknet environment.

8.2 Phase 1/2 - Image Quality Assessment/Optical Character Recognition Results

In an integrated way, the analysis was conducted to phase 1 and phase 2 comprising a series of different tests and methodologies throughout the process of finalizing the pipeline. Once it was stipulated that during phase 1 the adaptive preprocessing is performed on the ID through the use of the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) algorithm and in phase 2 the subsequent extraction and processing of the data, it is necessary to ascertain how we can evaluate these two simultaneously. This is because phase 1 and phase 2 end up working as one. Once the ID enters the adaptive processing pipeline, it goes through a series of operations and transformations to improve its image quality. At the output of this pipeline, we will find the properly processed ID to give input to the stipulated Optical Character Recognition (OCR) engine. Once the ID is entered, the various identifying fields are extracted, following the logic described in the previous chapter 6.

Since we are approaching both phases as one, we can also assume that the evaluation of the same processes can and should be done together. Therefore, the evaluation metrics that can be collected from phase 1 will be used together with the metrics collected for phase 2.

Once the complete software pipeline was established we proceeded to test it in a Customer Relationship Management (CRM) environment. To do this, approximately 300 FileNet documents of different formats were loaded into the software.

Internally, in phase 2 the EasyOCR engine was used simultaneously with the Tesseract engine to collect as much information as possible since from what was tested in particular between Tesseract and EasyOCR is that in terms of utilization, the system works better when the power of the two is combined rather than alone. In addition, the ID detection model used in phase 0 is the YOLOv4 model. The results of the extraction performed are stored in a Comma-separated values (CSV) file that contains each row 1 CC found with the textual information of the fields that compose the ID, as well as the confidence associated with each of them and the information about its validation. From the experimental extraction performed with the final system, about 350 new entries were collected. That is, technically speaking, 350 CCs.

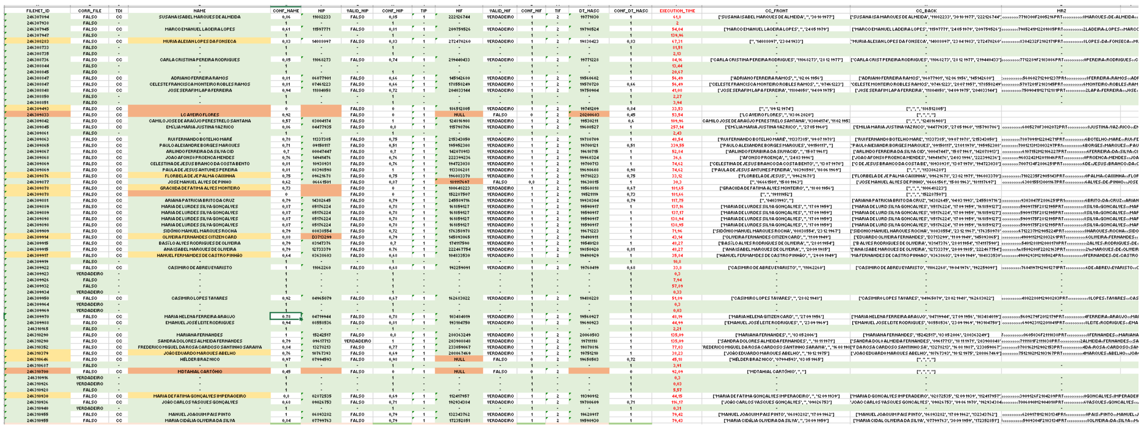


Figure 8.11: Output.csv file Example

Each of these new entries was initially compared manually with Ground Truth. In the CSV file, each field in each new row was assigned a color depending on the comparison to Ground Truth (figure 8.11). For example, in the case of the Name field, if the name is 100% complete, with all accents and with all full names, then it is considered a "Correct" entry and is therefore assigned the color green. If there is some kind of incorrectness in the detected name or it is incomplete, because it is missing a name or some accents, then it is given the yellow label, "Incomplete". Finally, in the event of a non-existent or completely random and erroneous detection of the Name field, it is marked as "Incorrect" in red (table 8.3).

STATE	DETECTIONS	PERCENTAGE
CORRECT	167	50%
INCOMPLETE	76	23%
INCORRECT	30	9%
-	62	19%
Total	335	100%

Table 8.3: Table of Names State Distribution

In addition to this "raw" analysis of the data, the extraction confidence metric was also used to determine the success of the experiment. Overall, the distribution of probabilities and their frequencies follows a distribution close to normal. In summary, **an average name extraction confidence of around 80% was achieved** (table 8.4).

CONFIDENCE GROUPS	DETECTIONS	PERCENTAGE
1	15	4%
>0.9 & <1	24	7%
>0.8 & <= 0.9	73	22%
>0.7 & <= 0.8	69	21%
>0.6 & <= 0.7	49	15%
>0.5 & <= 0.6	18	5%
>0.4 & <0.5	6	2%
0	18	5%
-	63	19%
TOTAL	335	100%

Table 8.4: Confidence Groups Distribution - Name Field

As for the remaining numeric fields to be extracted, these, similarly to what was done with the Name field, were extracted with attention to their confidence and extractions, and with attention to their validation. As seen in the previous chapter 6, the fields Número de Identificação Pessoal (NIP), Date of Birth, and Número de Identificação Fiscal (NIF) have functions that validate them, i.e., flags that guarantee total confidence in the extraction of the target field. Still, concerning the numeric fields, it should be noted that, unlike the Name field, they do not have a second state called "Incomplete", they are only "Correct" or "Incorrect".

STATE	DETECTIONS	PERCENTAGE
CORRECT	248	74%
INCORRECT	27	8%
-	60	18%
Total	335	100%

Table 8.5: Table of NIP State Distribution

CONFIDENCE GROUPS	DETECTIONS	PERCENTAGE
1	63	19%
>0.9 & <1	22	7%
>0.8 & <= 0.9	50	15%
>0.7 & <= 0.8	75	22%
>0.6 & <= 0.7	44	13%
>0.5 & <= 0.6	7	2%
>0.3 & <0.5	3	1%
0	8	2%
-	63	19%
TOTAL	335	100%

Table 8.6: Confidence Groups Distribution - NIP Field

VALIDATION STATE	AVG. CONFIDENCE	DETECTIONS
VALIDATION OK	100%	26
VALIDATION NOK	78%	246
NO EXTRACTION	-	63
Total Geral	65	335

Table 8.7: NIP Validation Analysis

For the NIP field, **about 248 correct entries were achieved from the 335 entries collected, which represents about 74% of the total NIPs that could be extracted** (table 8.5). In terms of the distribution of trusts in the NIP field, these followed a slightly more unusual distribution (table 8.6). This is because several empty entries were detected in the course of surveying the results. This more significant absence of NIP-type entries meant that, consequently, there was also a reduction in the overall average confidence of the NIP field extraction assigned by the OCR engines internally. In addition to these elements, there is the validation factor. As already mentioned, all fields except the Name field go through a validation process, where for each field type there is a specific validation function. In the table 8.7 we can extract two very useful pieces of information. The first is that the validation works because for all validated IDs the average confidence is 100%. And beyond this, we can take away that the validation rate is very low. Out of 335 detected and extracted entries, only 26 NIPs were validated (table 8.7). This result is mainly since the validation algorithm for the NIP field is extremely difficult to complete without any error in capturing the check digits required for its validation. Unlike the NIF field, the NIP field needs much more information to compute its checksum function, which leaves the NIP field in a much weaker position when it comes to its validation.

STATE	DETECTIONS	PERCENTAGE
CORRECT	199	59%
INCORRECT	73	22%
-	63	19%
Total	335	100%

Table 8.8: Table of NIF State Distribution

CONFIDENCE GROUPS	DETECTIONS	PERCENTAGE
1	203	61%
>0.9 & <1	8	2%
>0.7 & <0.9	8	2%
0	52	16%
-	63	19%
TOTAL	335	100%

Table 8.9: Confidence Groups Distribution - NIF Field

Taking the NIF field from the collected data, it turns out that about **59% of the extracted NIFs are correct, and that 61% of the collected NIFs have extraction confidence of 100%** (table 8.9). Looking at the table 8.10, we can see right away what we were previously struggling with. Where of the 335 entries collected 199 of these have validated NIFs.

VALIDATION STATE	AVG. CONFIDENCE	DETECTIONS
VALIDATION OK	100%	199
VALIDATION NOK	25%	73
NO EXTRACTION	-	63
Total Geral	65	335

Table 8.10: NIF Validation Analysis

Finally, the gathered data showed that **73% of the collection date of birth entries were correct** (table 8.11). Furthermore, when it comes to extraction confidence (table 8.12), and thanks again to the validation phase in post-processing, **41% of the extracted dates have extraction confidence of 100%, and over 85% of these have the confidence of over 70%.**

STATE	DETECTIONS	PERCENTAGE
CORRECT	245	73%
INCORRECT	30	9%
-	60	18%
Total	335	100%

Table 8.11: Table of Birth Date State Distribution

CONFIDENCE GROUPS	DETECTIONS	PERCENTAGE
1	138	41%
>= 0.9 & <1	15	4%
>= 0.8 & <0.9	31	9%
>0.7 & <0.8	34	10%
>0.6 & <= 0.7	22	7%
>0.3 & <= 0.6	14	4%
0	18	5%
-	63	19%
TOTAL	335	100%

Table 8.12: Confidence Groups Distribution - Birth Date Field

When a field is validated, it automatically receives the confidence of 100%. Turning now to phase 1, in addition to the values presented so far, the BRISQUE score is also stored in the CSV file. This information was stored to try to understand how impactful the application of the adaptive preprocessing pipeline becomes on the intelligent system. In the following table 8.13 we can see the distribution of the number of detected fields as a function of the score assigned by the Image Quality Assessment (IQA) algorithm.

IQA SCORE	Names	NIPs	Birth Dates	NIFs	MRZs
<35.0	255	246	123	246	123
>= 35.0	39	38	18	35	18
Total	294	284	141	281	141

Table 8.13: Fields count in function of the IQA score obtained by the ID

Although it is not possible to visualize these results at a first glance, we can assume two assumptions right from the start. One is that the dataset chosen does not include such an exorbitant number of complex scenarios and therefore, from this assumption we should not draw conclusions about the usefulness of the preprocessing pipeline installed. Another is that, despite the low sample size of data captured in the previous table for the IDs with a score greater than 35.0, these turn out to be indicators that the pipeline is a success. This is because of the totality of IDs with low BRISQUE scores detected in the IQA algorithm most of these (more than 70%) resulted in a perfect extraction.

In official terms, the total experience extraction time lasted approximately 8955 seconds or an average of 29.85 seconds of inference per document. In addition, the system was able to filter and identify the documents that were corrupted. Of all the loaded FileNets, eight of them were corrupted.

After this experiment, the data collected in the CSV file (figure 8.11), was sent to the CRM team, where they also manually tested the insertion of results in the informative tables, internal to CRM. These data were entered one by one into the appropriate fields. **In the CRM environment, in the process that gave rise to this experience in the Altice/MEO company, the following indicator results were achieved:**

- **From the records sent that had valid NIF and NIP, or only the NIF field validated, the data was useful to correct approximately 25% of the matching records in the Altice/MEO customer registry (including Name andr NIP);**
- **In the case where only the NIP was correct or validated since we got less matching, we identified anyway, an improvement although less expressive, in Names (completing missing names or surnames), around 10%;**
- **For valid dates of birth, in 90% of the cases, the information came to fill in the absence (the vast majority) or correct information that was loaded in CRM (occasional cases).**

Despite the superficial test carried out with a few hundred FileNet documents, it was possible to practically test the extraction capacity of the technology inherent to the project developed. From what had been said and explored concerning previous work done in a CRM environment, there were quite significant improvements to the extraction of the target field of date of birth. Where once it was only possible to have an extraction rate of around 60%, in contrast to the 90% achieved with this new open-source approach. Something that, even despite the positive results, always leaves room for improvement, especially when it comes to validating these dates, taking into account the needs of Altice/MEO. Extracting part of the information present in a CC when you don't have both also proved to be a very important strategy for what is the use and recycling of data that is often lost due to lack of "complete information", i.e. the entire ID.

The usefulness and necessity of the pipeline built to process and consolidate digitized information is thus demonstrated.

8.3 System Description and Integration

maryHeaven is an intelligent pipeline for the extraction of digitized information in CRM context for Altice/MEO. It is the result of the combined efforts of the various prototyping versions to which this project was subjected, plus some specific features that make it versatile and user-friendly for internal use at Altice/MEO.

maryHeaven is a solution programmed in a Python environment, which contemplates an intelligent system to collect and distribute documents in parallel by different processors, to parallelize and optimize the whole process of collection and extraction of digitized information. The system begins initially by allocating a pool of workers. This pool will have a size equal to the number of colors available in the machine, something that, in theoretical terms, means that the more computational power available to *maryHeaven*, the faster it behaves and the better its performance.

Based on the previously analyzed data, it was determined that both the $CSRESNEXT50-PANET-SPP$ and the YOLOv4 networks had the best results in the phase 0 inertia to the *maryHeaven* system. As a result of what happened in phase 2 of the pipeline, it was decided to beef up the extraction and use both trained networks. This decision is justified by the fact that both networks have demonstrated a low impact on the performance of *maryHeaven*, whether at the level of model storage in memory or in the medium time of document inference.

Still speaking in internal terms, each processor when working with one FileNet file at a time, internally also uses asynchronous Threads to handle in parallel the different phases of the pipeline if, for example, there is more than one element to sort within the overall document. A practical example of this is when, for example, two of the parts that make up a CC are detected in phase 0. Here, the two parts are processed in parallel by two asynchronous Threads within the same process. Implementing the *maryHeaven* software with multiprocessing and parallel computing has made the pipeline a more robust and better-performing solution.

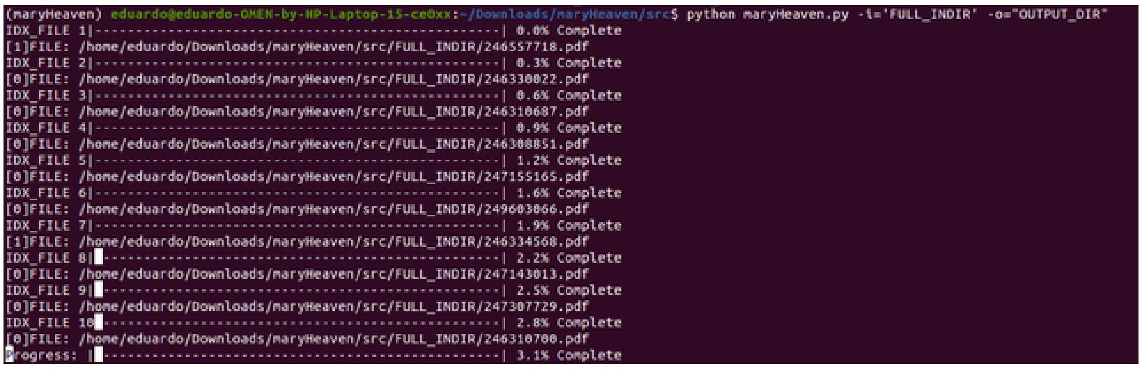
Within what are the functionalities of *maryHeaven*, it is worth noting the insertion of a new output data, which is a list of similar or corrected words. This list consists of words found or detected in the Name field which, going through a dictionary of Portuguese first names and surnames, are not recognized. Typically, when this happens, it is either a word belonging to the blacklist of words but which is not properly pointed out and so is detected anyway, or else it may be the occurrence of a name that is not in *maryHeaven*'s knowledge base, and so is interpreted as incorrect (but saved anyway). In terms of phase 1, *maryHeaven* has integrated the adaptive preprocessing pipeline proposed in chapter 5, where it makes use of BRISQUE's algorithm to infer the quality of the detected ID and proceed to preprocess it appropriately.

In terms of the second phase, that is, the part that has integrated the OCR engine plus all the post-processing and validation tools, this has two integrations. One is adapted to the Tesseract model and the other to the EasyOCR model. These implementations arose from the need to have extraction and reliability tests done with both engines, to ascertain which of the two should be implemented at the end of the system. After several discoveries and analyses made to the extraction power of both, it was concluded that both engines have their qualities, which, complemented, would be in favor of good system results. That said, the possibility was made at the time of document extraction to internally activate or deactivate one of the two modules, or else leave both active for joint processing. This approach not only greatly offsets the average of confidence extracted for each of the fields, but also the certainty in the extraction made to the characters that compose the values of the personal information fields.

As for the post-processing logic, where the pairing between parts of the same CC is done, this logic was maintained. Except for the fact that now, in this phase, a cut is made in what is the search for the different output IDs from the preprocessing phase. That is, in a normal situation, instead of processing all the IDs resulting from the passage through the adaptive pipeline, a cut is made, along the extraction process, of the extracted trusts. The cut is made if, for example, the extraction with the maximum confidence registered up to height passes a threshold value, in this case, the value used in a general way was 85%. In practical terms, instead of processing about six or seven different inputs for the same ID and then analyzing these values, the strategy presented involves immediately breaking the internal cycle that iterates over these input IDs, which will consequently have a positive impact on the system's performance and on the total inference and extraction time.

From what was previously presented as prototype solutions for the project in question, all of them had Graphical User Interface (GUI). Now, taking into account that the software has to support an above-average performance of manual processing done by a human being, the final intelligent system must support as few extra features as possible, which consequently excludes the need for a GUI.

In the terminal (figure 8.12), *maryHeaven* receives as arguments the input directory $--[i]nput_dir$ and the output directory $--[o]utput_dir$. The input directory must contain all the format types that are acceptable in the CRM environment and also accepted by the pipeline induced in the *maryHeaven* system. The output directory is then specified to receive the output files that contain, in addition to the information collected for the customer's fields, statistical data about the extraction that occurred, as well as a text file with the path to the files that *maryHeaven* detected as corrupt. Once the *maryHeaven* system is activated with the proper arguments, a status bar will appear on the terminal that will tell us over time what percentage of FileNets was processed (figure 8.12).



```
(maryHeaven) eduardo@eduardo-OMEN-by-HP-Laptop-15-ce0xx:~/Downloads/maryHeaven/src$ python maryHeaven.py -i='FULL_INDIR' -o="OUTPUT_DIR"
IDX_FILE 1|--------------------------------------------------| 0.0% Complete
[1]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246557718.pdf
IDX_FILE 2|--------------------------------------------------| 0.3% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246330822.pdf
IDX_FILE 3|--------------------------------------------------| 0.6% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246310687.pdf
IDX_FILE 4|--------------------------------------------------| 0.9% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246308851.pdf
IDX_FILE 5|--------------------------------------------------| 1.2% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/247155165.pdf
IDX_FILE 6|--------------------------------------------------| 1.6% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/249603066.pdf
IDX_FILE 7|--------------------------------------------------| 1.9% Complete
[1]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246334568.pdf
IDX_FILE 8|█-------------------------------------------------| 2.2% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/247143013.pdf
IDX_FILE 9|█-------------------------------------------------| 2.5% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/247307729.pdf
IDX_FILE 10█-------------------------------------------------| 2.8% Complete
[0]FILE: /home/eduardo/Downloads/maryHeaven/src/FULL_INDIR/246310700.pdf
Progress: |█-------------------------------------------------| 3.1% Complete
```

Figure 8.12: *maryHeaven* - Running Environment (Ubuntu 20.04)

At the end of the extraction, *maryHeaven* returns as output a summary about the extraction (figure 7.2). Apart from seeing relevant information about our FileNets detected in CRM environments, such as the total number of FileNets, the count of each format detected, and the number of corrupted FileNets, we can also see the information gathered from the IDs detected and the number of extraction for each field, as well as their respective extraction confidence levels. This output can then be saved, as shown in figure 8.12, to a text file in the output directory designated at the beginning of the pipeline as an argument to the *maryHeaven* system.

It is also important to say that during the extraction process, as a FileNet is processed, there is the creation of an asynchronous Thread of the process which is responsible for writing a new line of information in the output files.

In summary, the Output directory in CRM environment, where the results will later be processed on the biography tables, will consist of four different files in total:

- *Output.csv* - This is the file in CSV format that contains the table information collected by *maryHeaven* during the extraction process. Each line in the CSV file represents one detected CC. In each line the following information about the collected ID is present:
  - `FILENET_ID` - Each file collected in the input directory has a nine digits number, the FileNetID. Which is also the name of the file. This id is the key to establish a connection with the informative sheets inside CRM;
  - `IQA_SCORE` - BRISQUE average score between 0 and 100, that represents the average image quality in the detected ID;
  - `CORR_FILE` - Boolean flag that says if the file in consideration is corrupted or not;
  - `TDI` - Type of ID detected ('CC');
  - `NAME` - Final post-processed Name field;
  - `CONF_NAME` - Confidence of the final Name field detected;
  - `NAMES_AUX` - Auxiliary list with words or set of words that were extracted in the demarked area around the name field, and can have potential other names not consider by *maryHeaven* logic. Works as a backup information bucket;

  - `NIP` - Final post-processed NIP field;
  - `VALID_NIP` - Boolean flag that says if the final; NIP in consideration is valid or not;
  - `CONF_NIP` - Confidence of the final NIP field detected;
  - `TIP` - Type of NIP detected ('1', for CC);
  - `NIF` - Final post-processed NIF field;
  - `VALID_NIF` - Boolean flag that says if the final NIF in consideration is valid or not;
  - `CONF_NIF` - Confidence of the final NIP field detected;
  - `TIF` - Type of NIF detected ('2', for CC);
  - `DT_NASC` - Final post-processed birth date field;
  - `VALID_DATE` - Boolean flag that says if the final birth date in consideration is valid or not;
  - `CONF_DT_NASC` - Confidence of the final birth date field detected;
  - `EXECUTION_TIME` - Total elapsed time of the information extracted into the row in target;
  - `CC_FRONT` - All the set of information from the CC_{front}, in list format;
  - `CC_BACK` - All the set of information from the CC_{back}, in list format;
  - `MRZ` - All the 91 alphanumeric characters extracted from the CC_{back}.

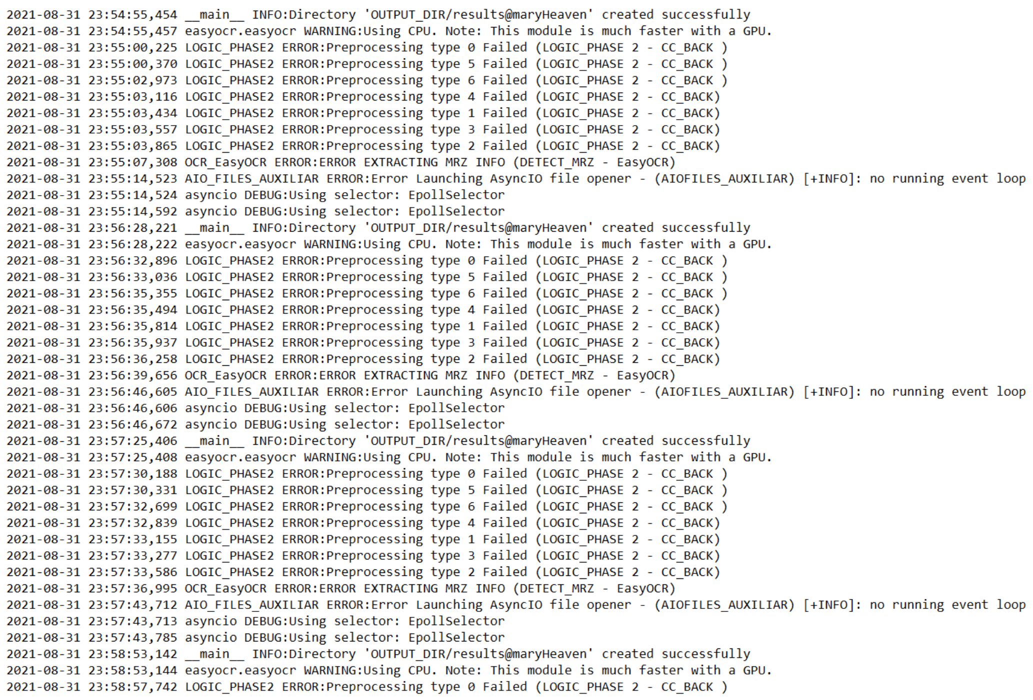
- *Output.txt* - This file is a light version of the above file, as it only contains the raw information needed for integration with the CRM. In the figure 7.2, you can see the normal formatting expected for each CC detected within this file, it is also, the expected format in a CRM environment.

- *corrupted_files_list.txt* - Text format file containing in each line the path to the file detected as corrupted by *maryHeaven*. It is a supplementary tool but offers support to also motivate a start in the search for compromised data in a CRM environment.

- *stats_file_extraction_id_date_format.txt* - As already mentioned, this file is the result of the user's choice to save in text file format the statistical results obtained from the extraction performed. These follow the structure shown in figure 8.12.

maryHeaven in addition to returning all collected results and metrics also has internally implemented a system of log messages (figure 8.13). All the fundamental modules that makeup *maryHeaven* internally import Log methodologies so that they can store in the background errors or pertinent debug information that can later aid in the interpretation of new problems and, consequently, new approaches. This factor is something that implements the intelligent *maryHeaven* system stand out as it allows not only the implementer of the system to fix bugs and change certain data processing logics but also allows for a more independent dynamics of the system within the CRM work environment. Where any element with competence in the area can, by analyzing the log files, try to interpret and correct certain gaps inherent in the implemented system.

These log files are automatically created at the beginning of new extraction and are stored internally in the *maryHeaven* system path in the directory called "Log". In terms of integrating the *maryHeaven* system in a CRM environment, taking into account the sensitivity of the data inherent in this project, the integration was done based on the need to have control on data output outside the CRM desktop. Taking this into account, the proposal



```
2021-08-31 23:54:55,454 __main__ INFO:Directory 'OUTPUT_DIR/results@maryHeaven' created successfully
2021-08-31 23:54:55,457 easyocr.easyocr WARNING:Using CPU. Note: This module is much faster with a GPU.
2021-08-31 23:55:00,225 LOGIC_PHASE2 ERROR:Preprocessing type 0 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:55:00,370 LOGIC_PHASE2 ERROR:Preprocessing type 5 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:55:02,973 LOGIC_PHASE2 ERROR:Preprocessing type 6 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:55:03,116 LOGIC_PHASE2 ERROR:Preprocessing type 4 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:55:03,434 LOGIC_PHASE2 ERROR:Preprocessing type 1 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:55:03,557 LOGIC_PHASE2 ERROR:Preprocessing type 3 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:55:03,865 LOGIC_PHASE2 ERROR:Preprocessing type 2 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:55:07,308 OCR_EasyOCR ERROR:ERROR EXTRACTING MRZ INFO (DETECT_MRZ - EasyOCR)
2021-08-31 23:55:14,523 AIO_FILES_AUXILIAR ERROR:Error Launching AsyncIO file opener - (AIOFILES_AUXILIAR) [+INFO]: no running event loop
2021-08-31 23:55:14,524 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:55:14,592 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:56:28,221 __main__ INFO:Directory 'OUTPUT_DIR/results@maryHeaven' created successfully
2021-08-31 23:56:28,222 easyocr.easyocr WARNING:Using CPU. Note: This module is much faster with a GPU.
2021-08-31 23:56:32,896 LOGIC_PHASE2 ERROR:Preprocessing type 0 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:56:33,036 LOGIC_PHASE2 ERROR:Preprocessing type 5 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:56:35,355 LOGIC_PHASE2 ERROR:Preprocessing type 6 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:56:35,494 LOGIC_PHASE2 ERROR:Preprocessing type 4 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:56:35,814 LOGIC_PHASE2 ERROR:Preprocessing type 1 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:56:35,937 LOGIC_PHASE2 ERROR:Preprocessing type 3 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:56:36,258 LOGIC_PHASE2 ERROR:Preprocessing type 2 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:56:39,656 OCR_EasyOCR ERROR:ERROR EXTRACTING MRZ INFO (DETECT_MRZ - EasyOCR)
2021-08-31 23:56:46,605 AIO_FILES_AUXILIAR ERROR:Error Launching AsyncIO file opener - (AIOFILES_AUXILIAR) [+INFO]: no running event loop
2021-08-31 23:56:46,606 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:56:46,672 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:57:25,406 __main__ INFO:Directory 'OUTPUT_DIR/results@maryHeaven' created successfully
2021-08-31 23:57:25,408 easyocr.easyocr WARNING:Using CPU. Note: This module is much faster with a GPU.
2021-08-31 23:57:30,188 LOGIC_PHASE2 ERROR:Preprocessing type 0 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:57:30,331 LOGIC_PHASE2 ERROR:Preprocessing type 5 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:57:32,699 LOGIC_PHASE2 ERROR:Preprocessing type 6 Failed (LOGIC_PHASE 2 - CC_BACK )
2021-08-31 23:57:32,839 LOGIC_PHASE2 ERROR:Preprocessing type 4 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:57:33,155 LOGIC_PHASE2 ERROR:Preprocessing type 1 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:57:33,277 LOGIC_PHASE2 ERROR:Preprocessing type 3 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:57:33,586 LOGIC_PHASE2 ERROR:Preprocessing type 2 Failed (LOGIC_PHASE 2 - CC_BACK)
2021-08-31 23:57:36,995 OCR_EasyOCR ERROR:ERROR EXTRACTING MRZ INFO (DETECT_MRZ - EasyOCR)
2021-08-31 23:57:43,712 AIO_FILES_AUXILIAR ERROR:Error Launching AsyncIO file opener - (AIOFILES_AUXILIAR) [+INFO]: no running event loop
2021-08-31 23:57:43,713 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:57:43,785 asyncio DEBUG:Using selector: EpollSelector
2021-08-31 23:58:53,142 __main__ INFO:Directory 'OUTPUT_DIR/results@maryHeaven' created successfully
2021-08-31 23:58:53,144 easyocr.easyocr WARNING:Using CPU. Note: This module is much faster with a GPU.
2021-08-31 23:58:57,742 LOGIC_PHASE2 ERROR:Preprocessing type 0 Failed (LOGIC_PHASE 2 - CC_BACK )
```

Figure 8.13: *maryHeaven* - Log File Example

considered for integration is based on a kind of secure bridge between the data input channel (FileNet files) and the environment where the intelligent *maryHeaven* system runs. In an integrated manner, the *maryHeaven* system has access via a secure VPN connection to Altice/MEO's internal CRM network. From here an intermediary is used that will speed up document transfer internally, an File Transfer Protocol (FTP) server. Through an FTP server, it is possible to send and transfer documents between machines on a company's network. In Altice/MEO's case is no exception, therefore the use of the FTP server comes in response to the complexity that exists in transferring documents of a sensitive nature within the company's network.

Once this FTP communication is established, *maryHeaven* operating in the same environment processes the files stored on the server and returns the collected information to the specified output board. In this case, the output directory, preferably, will be the same as the input one, so that it can have the information together. It should be noted that this integration is theoretical and is in the process of being finalized, taking into account that some internal artifacts need to be settled. Anyway, this integration proposal will be the proposal approved by Altice/MEO to host the *maryHeaven* intelligent system to not only improve it and integrate it in other future applications and challenges but also to make it public to improvements and beneficial changes, that help in the internal processes of consolidation and processing of digitized personal information, at Altice/MEO.

This page is intentionally left blank.

Chapter 9

Conclusion

During the project, it was possible to gradually answer the business question imposed at the beginning by Altice/MEO, and during this long journey, an answer was given, the *maryHeaven* intelligent system.

maryHeaven is an intelligent system that internally integrates a pipeline that makes use of Deep Learning (DL) technology and techniques and technologies inherent to the Computer Vision (CV) area. This pipeline inherent to *maryHeaven* is its heart, the engine that pumps information to be extracted from documents with scanned information and channels that information in a continuous flow to databases in a Customer Relationship Management (CRM) environment.

This final pipeline, consists of three primary processing phases. The first phase is based on DL technology, where Object Detection models are used to try as best as possible to answer the first objective inherent to the matrices of this project, to correctly locate and identify the types of Identity Documents (IDs) present in FileNets, documents that make up the database to CRM. In this phase, several models with different methodologies and specifications were evaluated. From the results obtained in this phase, it is important to highlight the advances achieved in the location and classification of the Cartão de Cidadão (CC) type IDs. For these models, the results achieved, taking into account the required and exercised evaluation metrics, showed to have very good detection and recognition power. In specific terms, the values achieved by the $CSRESNEXT50-SPP-PANET$ network should be highlighted, as it obtained very high specificity and accuracy values compared to the other models studied, and therefore this network was used in parallel with a You Only Look Once (YOLO)v4 network to set up phase 0 for the *maryHeaven* system.

Still, within phase 0, a theoretical survey of the possibility of expanding the *maryHeaven* system to the detection of other types of IDs presented in this project, beyond the detection already achieved for CCs, was also performed. In this test, it was possible to conclude that the possibility of expansion is confirmed and that the results already achieved by the trained models are quite satisfactory and are considered ready for a further evaluation phase. This phase has not been carried out at the moment because the logic inherent in *maryHeaven* is still specifically designed to detect and optimally extract CC data. In any case, it is necessary, from a future work perspective, to gather the necessary conditions to strengthen this inherent need for Altice/MEO. More explicitly, solutions for future work would be to collect as many documents available in FileNet as possible to try to broaden the horizons of extraction and complex scenarios. This will make the object detection models for multiclass more generalized to detection in complex environments within the target documents and to much higher confidence in detecting the different types of IDs than already achieved during this project. During the project, it was possible to gradually answer the business question imposed at the beginning by Altice/MEO, and during this long journey, an answer was given, the *maryHeaven* intelligent system.

maryHeaven is an intelligent system that internally integrates a pipeline that makes use of DL technology and techniques and technologies inherent to the CV area. This pipeline inherent to *maryHeaven* is its heart, the engine that pumps information to be extracted from documents with scanned information and channels that information in a continuous flow to databases in a CRM environment. This final pipeline, consists of 3 primary processing phases. The first phase is based on DL technology, where Object Detection models are used to try as best as possible to answer the first objective inherent to the matrices of this project, to correctly locate and identify the types of IDs present in FileNets, documents that make up the database to CRM. In this phase, several models with different methodologies and specifications were evaluated. From the results obtained in this phase, it is important to highlight the advances achieved in the location and classification of the CC type IDs. For these models, the results achieved, taking into account the required and exercised evaluation metrics, showed to have very good detection and recognition power. In specific terms, the values achieved by the $CSRESNEXT50-SPP-PANET$ network should be highlighted, as it obtained very high specificity and accuracy values compared to the other models studied, and therefore this network was used in parallel with a YOLOv4 network to set up phase 0 for the *maryHeaven* system.

In addition to the variety of scenarios and their inherent complexity, a survey of example documents with more classes than the CC class should be done, to not only better the role played by the object detectors in phase 0, but also, consequently, in the extraction phase, where the quality of the extracted information is, one way or another, dependent on what happened in the previous phases. In the phase after the one analyzed just now, phase 1, what was accomplished, in terms of image quality inference was considered a success. It can be seen that from some of the results shown in the previous chapter, not only the application of the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) algorithm was considered to have added value, but also the logic implemented behind this phase came to show results that were quite compensatory for what would be an initial approach. Bridging with what was contextualized at the beginning of this document, and taking into account the nature of the data present in the datasets produced and enunciated throughout the progress within the project, it is concluded that, after the numerous citations about the image quality problems, these were successfully answered, but will always be the target of improvement. This is because, besides the fact that we are working on a very limited scope of what is the CRM environment as a whole, it is also, on the one hand, super complicated to manage all the possible preprocessing options as well as to infer for each specific case a specific strategy to process it. To sum up, the efforts made in phase 1 were of utmost importance, because the main problem, which has always been detected, is inherent to the quality presented by the FileNet documents.

Having said that, and now focusing on the most pertinent phase within the *maryHeaven* intelligent system pipeline - phase 2 - it should be noted that this had very satisfactory results in what is expected for the Altice/MEO company. Of these, it is important to point out that of the data obtained by the *maryHeaven* system during the various test versions in documents stored internally in an environment In this phase, as mentioned, two Optical Character Recognition (OCR) engines of an open-source nature, EasyOCR, and Tesseract, were used and tested. Both engines proved to have qualities in certain types of extraction. On the one hand, the EasyOCR engine proved to be more robust in detecting characters of numeric rather than alphabetic origin, and therefore this specialty was "transferred", in terms of pipeline, to the detection and extraction of mostly numeric fields, such as Número de Identificação Fiscal (NIF) and NIF. In the case of fields such as Name, Tesseract proved to be far superior, which proves the theoretical survey about both done at the beginning of this project. Despite the differences found, we conclude that it would be more profitable for the whole system if it complemented the computational power of both.

Something that was concluded to be positive after the experiments carried out in a CRM environment. In the first approach in the CRM environment, and as described in more detail in the previous chapter, the data collected by *maryHeaven* resulted in the correction of about a quarter of the biographical fields, where they had validated or corrected NIFs and Número de Identificação Pessoals (NIPs). In addition, in the same experiment, an improvement of about 10% was achieved in the information about the Name field. In the case of dates of birth, it was possible, in 90% of the cases, to fill a large set of empty fields or correct data that had already been uploaded before. Although we have focused and spent more time on exploring the technology inherent to Phase 0 of the project, it is extremely important to keep in constant contact with the novelty related to the advances made in the OCR area, as well as the practical applications of it.

More specifically concerning the prospects for future work that can and should be carried out in the context of improving and progressing the results achieved in phase 2 of the pipeline inherent in the *maryHeaven* intelligent system. In a future vision, an approach where DL technology is used, somewhat like in phase 0 of this project. In this proposal, the second phase would consist of two networks, a character detection (backbone) network, which would be adapted from an already pre-trained network, and a recognition network specialized for the type of font and size we are seeking to find in the IDs detected in the CRM FilentNet. This approach would have to rely upon, somewhat similar to what was proposed for phase 1, on a database rich in diverse inherent scenarios and complexities. This, so that in a way, the losses in character detection and extraction would be minimal. As can be seen, there are still plenty of implementation possibilities to help Altice/MEO in the processing and consolidation of digitized customer information. Something only possible if there is, once again, a strengthening in is the knowledge base of the models that can be trained to assist *maryHeaven* in a future expansion.

References

[1] eportugal.gov.pt - portal de serviços públicos. https://eportugal.gov.pt/. Accessed: 2021-09-01.

[2] Mathworks - contrast enhancement techniques. https://www.mathworks.com/help/images/contrast-enhancement-techniques.html. Accessed: 2020-09-28.

[3] Sef - serviços de estrangeiros e fronteiras. https://imigrante.sef.pt/. Accessed: 2021-08-28.

[4] Wikipedia - bilhete de identidade. https://en.wikipedia.org/wiki/Bilhete_de_Identidade_(Portugal). Accessed: 2020-09-28.

[5] Rosebrock A. Text skew correction with opencv and python. https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/. Accessed: 2020-12-29.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Arwa Al-Khatatneh, Sakinah Ali Pitchay, and Musab Al-qudah. A review of skew detection techniques for document. In *2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*. IEEE, March 2015.

[8] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[9] Yasser Alginahi. Preprocessing techniques in character recognition. In *Character Recognition*. Sciyo, August 2010.

[10] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection, 2019.

[11] Veena Bansal and RMK Sinha. Integrating knowledge sources in devanagari text recognition system. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(4):500–505, 2000.

[12] Deepa Berchmans and SS Kumar. Optical character recognition: an overview and an insight. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 1361–1365. IEEE, 2014.

[13] Jürgen Beyerer. *Machine vision : automated visual inspection : theory, practice and applications.* Springer, Berlin, 2016.

[14] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[15] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] Thomas M Breuel. Two geometric algorithms for layout analysis. In *International workshop on document analysis systems*, pages 188–199. Springer, 2002.

[17] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[18] Huaigu Cao, Rohit Prasad, Prem Natarajan, and Ehry MacRostie. Robust page segmentation based on smearing and error correction unifying top-down and bottom-up approaches. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 392–396. IEEE, 2007.

[19] Pragnan Chakravorty. What is a signal? [lecture notes]. *IEEE Signal Processing Magazine*, 35(5):175–177, September 2018.

[20] Sukalpa Chanda, Oriol Ramos Terrades, and Umapada Pal. Svm based scheme for thai and english script identification. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 1, pages 551–555. IEEE, 2007.

[21] Chee Kheng Chng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 01:935–942, 2017.

[22] David Crandall, Sameer Antani, and Rangachar Kasturi. Extraction of special effects caption text events from digital video. *International journal on document analysis and recognition*, 5(2):138–157, 2003.

[23] E. Roy Davies. *Computer vision: principles, algorithms, applications, learning.* Elsevier Academic Press, 2018.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[25] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection, 2019.

[26] Mohamed Elgendy. *Deep Learning for Vision Systems.* O'Reilly Media, 2020.

[27] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[28] Wael Farag. Recognition of traffic signs by convolutional neural nets for self-driving vehicles. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 22(3):205–214, November 2018.

[29] Robert Fisher. *HIPR: hypermedia image processing reference.* John Wiley, 1996.

[30] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

[31] Basilios Gatos, Ioannis Pratikakis, and Stavros J Perantonis. Adaptive degraded document image binarization. *Pattern recognition*, 39(3):317–327, 2006.

[32] Kaiser J Giri and Rumaan Bashir. Design & implementation of a novel cognitive character recognition technique. In *2013 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND COMMUNICATION (ICSC)*, pages 225–229. IEEE, 2013.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[34] Ross B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[35] Rafael Gonzalez. *Digital image processing.* Pearson, New York, NY, 2018.

[36] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[37] Ralph Grishman. Information extraction: Techniques and challenges. In *International summer school on information extraction*, pages 10–27. Springer, 1997.

[38] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. Recursive xy cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 2, pages 952–955. IEEE, 1995.

[39] Paul W Handel. Statistical machine. *General Electric Company, assignee. Patent*, 1915993:27, 1933.

[40] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[42] JS Sepp Hochreiter and J Schmidhuber. Long short-term memory (neural computation 9 (8): 1735-1780, 1997). *Fakultät für Informatik, Technische Universität München, License*, 1997.

[43] Han Hu, Chengquan Zhang, Yuxuan Luo, Yuzhuo Wang, Junyu Han, and Errui Ding. Wordsup: Exploiting word annotations for character based text detection. In *Proceedings of the IEEE international conference on computer vision*, pages 4940–4949, 2017.

[44] Jonathan Hull. *Document analysis systems II.* World Scientific, Singapore River Edge, NJ, 1998.

[45] IEEE. *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*. IEEE, 2019.

[46] Google Inc. Vision ai - google cloud platform. `https://cloud.google.com/vision`, 2021. Accessed: 2020-10-15.

[47] JaidedAI. EasyOCR:ready-to-use ocr with 80+ supported languages and all popular writing scripts including latin, chinese, arabic, devanagari, cyrillic and etc. `git://github.com/jaidedai/easyocr.git`, 2021.

[48] Bilal Jan, Haleem Farman, Murad Khan, Muhammad Imran, Ihtesham Ul Islam, Awais Ahmad, Shaukat Ali, and Gwanggil Jeon. Deep learning in big data analytics: A comparative study. *Computers Electrical Engineering*, 75:275–287, 2019.

[49] Gopal Datt Joshi, Saurabh Garg, and Jayanthi Sivaswamy. A generalised framework for script identification. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(2):55–68, 2007.

[50] Kyung-Won Kang and Jin Hyung Kim. Utilization of hierarchical, stochastic relationship modeling for hangul character recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1185–1196, 2004.

[51] Thotreingam Kasar and AG Ramakrishnan. Cococlust: Contour-based color clustering for robust binarization of colored text. *Proc. The Third CBDAR*, pages 11–17, 2009.

[52] Anthony Kay. Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2, July 2007.

[53] Woosuk Kim, Hyunwoong Cho, Jongseok Kim, Byungkwan Kim, and Seongwook Lee. Yolo-based simultaneous target detection and classification in automotive fmcw radar systems. *Sensors*, 20:2897, 05 2020.

[54] Kofax Inc. Kofax omnipage capture 18.0.

[55] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[57] Nour-Eddine Lasmar, Youssef Stitou, and Yannick Berthoumieu. Multiscale skewed heavy tailed model for texture analysis. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 2281–2284. IEEE, 2009.

[58] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: Analysis, applications, and prospects, 2020.

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[61] Aditya Lohia, Kalyani Dhananjay Kadam, Rahul Raghvendra Joshi, and Anupkumar M Bongale. Bibliometric analysis of one-stage and two-stage object detection. *Library Philosophy and Practice*, pages 1–32, 2021.

[62] Zhidong Lu, Richard Schwartz, and Christopher Raphael. Script-independent, hmm-based text line finding for ocr. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 551–554. IEEE, 2000.

[63] Raman Maini and Himanshu Aggarwal. A comprehensive review of image enhancement techniques, 2010.

[64] Henri Maître. *From photon to pixel: the digital camera handbook*. John Wiley & Sons, 2017.

[65] J Mantas. An overview of character recognition methodologies. *Pattern recognition*, 19(6):425–430, 1986.

[66] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[67] MATLAB. *version 9.9.0 (R2020b)*. The MathWorks Inc., Natick, Massachusetts, 2020.

[68] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012.

[69] Shunji Mori, Ching Y Suen, and Kazuhiko Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029–1058, 1992.

[70] Prem Natarajan, Shirin Saleem, Rohit Prasad, Ehry MacRostie, and Krishna Subramanian. Multi-lingual offline handwriting recognition using hidden markov models: A script-independent approach. In *Summit on Arabic and Chinese Handwriting Recognition*, pages 231–250. Springer, 2006.

[71] Premkumar Natarajan, Zhidong Lu, Richard Schwartz, Issam Bazzi, and John Makhoul. Multilingual machine printed ocr. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):43–63, 2001.

[72] Nguyen, Thanh Cong, Dinh Tuan Nguyen, Tran, and Quoc Long. Information extraction from id card via computer vision techniques, 2018.

[73] Leonardo Nicolosi, A. Blug, A. Heider, Ronald Tetzlaff, and Heinz Höfler. A novel spatter detection algorithm based on typical cellular neural network operations for laser beam welding processes. *Measurement Science and Technology*, 23:015401, 12 2011.

[74] Peter Norvig. Natural language corpus data. *Beautiful data*, pages 219–242, 2009.

[75] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[76] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[78] James W. Peltier, Debra Zahay, and Donald R. Lehmann. Organizational learning and CRM success: A model for linking organizational practices, customer data quality, and performance. *Journal of Interactive Marketing*, 27(1):1–13, February 2013.

[79] Xujun Peng, Huaigu Cao, Srirangaraj Setlur, Venu Govindaraju, and Prem Natarajan. Multilingual ocr research and applications: an overview. In *Proceedings of the 4th International Workshop on Multilingual OCR*, pages 1–8, 2013.

[80] Marijana Petrović. Data quality in customer relationship management (CRM): Literature review. *Strategic Management*, 25(2):40–47, 2020.

[81] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Marco Carli, and Federica Battisti. Tid2008-a database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10(4):30–45, 2009.

[82] Simon J. D. Prince. *Computer vision: models, learning, and inference.* Cambridge University Press, 2014.

[83] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[84] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[85] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[86] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[87] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[88] Michael Ryan and Novita Hanafiah. An examination of character recognition on ID card using template matching approach. *Procedia Computer Science*, 59:520–529, 2015.

[89] Felipe Sasso, Ricardo Moraes, and Jean Martina. A proposal for a unified identity card for use in an academic federation environment. 09 2014.

[90] Karnran Sharifi and Alberto Leon-Garcia. Estimation of shape parameter for generalized gaussian distributions in subband decompositions of video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(1):52–56, 1995.

[91] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on image processing*, 15(11):3440–3451, 2006.

[92] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2550–2558, 2017.

[93] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.

[94] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), July 2019.

[95] Ronan Sicre, Ahmad Montaser Awal, and Teddy Furon. Identity documents classification as an image classification problem. Technical Report RT-0488, Inria Rennes - Bretagne Atlantique ; IRISA ; AriadNext, April 2017.

[96] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[97] Ray Smith, Daria Antonova, and Dar-Shyang Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, pages 1–8, 2009.

[98] Petru Soviany and Radu Tudor Ionescu. Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction. *CoRR*, abs/1803.08707, 2018.

[99] Carsten Steger. *Machine vision algorithms and applications.* Wiley-VCH, Weinheim, Germany, 2018.

[100] Chen Sun, Chen Tang, Xinjun Zhu, Xiaoyu Li, and Linlin Wang. An efficient method for salt-and-pepper noise removal based on shearlet transform and noise detection. *AEU - International Journal of Electronics and Communications*, 69(12):1823 – 1832, 2015.

[101] Richard Szeliski. *Computer vision : algorithms and applications.* Springer, London New York, 2011.

[102] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020.

[103] Gustav Tauschek. Reading machine. *US Pat*, 2026329, 1935.

[104] Niloofar Tavakolian, Azadeh Nazemi, and Donal Fitzpatrick. Real-time information retrieval from identity cards, 2020.

[105] tesseract ocr. Tesseract:tesseract open source ocr engine (main repository). `https://tesseract-ocr.github.io/`, 2019.

[106] Tzutalin. Labelimg - gui image annotation tool. `https://github.com/tzutalin/labelImg`, 2015. Accessed: 2020-10-15.

[107] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M.Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[108] Pere Vilás. Classification of identity documents using a deep convolutional neural network, 2018.

[109] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *CoRR*, abs/2011.08036, 2020.

[110] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[111] Xianghui Wei, Shaoping Ma, and Yijiang Jin. Segmentation of connected chinese characters based on genetic algorithm. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 645–649. IEEE, 2005.

[112] Liu Yuliang, Jin Lianwen, Zhang Shuaitao, and Zhang Sheng. Detecting curve text in the wild: New dataset and new solution. *ArXiv*, abs/1712.02170, 2017.

[113] Wang Yutao, Qin Tingting, Tian Ruixia, and Yang Gang. Recognition of license plate character based on wavelet transform and generalized regression neural network. In *2012 24th Chinese Control and Decision Conference (CCDC)*, pages 1881–1885. IEEE, 2012.

[114] Jian-Qi Zhang, Wei Xiong, Shuo Zhang, Yong Li, and Mang Feng. Generating the schrödinger cat state in a nanomechanical resonator coupled to a charge qubit. *Annalen der Physik*, 527(1-2):180–186, October 2014.

[115] Zhong-Qiu Zhao, Peng Zheng, Shou tao Xu, and Xindong Wu. Object detection with deep learning: A review, 2019.

[116] Jian Zhu, Hanjie Ma, Jie Feng, and Leiyan Dai. ID card number detection algorithm based on convolutional neural network. In *AIP Conference Proceedings*. Author(s), 2018.

This page is intentionally left blank.

Appendices

Appendix A - Evaluation Results for the Tensorflow Environment

FASTER R-CNN					
			FASTER_RCNN_RESNET50		
SUBCLASS	**DETECTIONS**	**CONFIDENCE AVG.**	**IoU AVG.**	**AVG. INFERENCE TIME(s)**	**TOTAL TIME (s)**
CC_back	591	99,55%	0,84	1,99	1173,25
CC_front	493	99,95%	0,95	2,02	994,55
TOTAL	**1084**	**99,73%**	**0,89**	**2,00**	**2167,79**
			FASTER_RCNN_RESNET101		
SUBCLASS	**DETECTIONS**	**CONFIDENCE AVG.**	**IoU AVG.**	**AVG. INFERENCE TIME(s)**	**TOTAL TIME (s)**
CC_back	600	99,71%	0,83	2,09	1255,01
CC_front	495	99,94%	0,95	2,13	1054,81
TOTAL	**1095**	**99,81%**	**0,88**	**2,11**	**2309,82**
			FASTER_RCNN_RESNET152		
SUBCLASS	**DETECTIONS**	**CONFIDENCE AVG.**	**IoU AVG.**	**AVG. INFERENCE TIME(s)**	**TOTAL TIME (s)**
CC_back	640	99,63%	0,78	7,02	4495,33
CC_front	532	99,83%	0,89	7,07	3760,11
TOTAL	**1172**	**99,72%**	**0,83**	**7,04**	**8255,44**

Table 1: Evaluation Results for the Faster-Region Based Convolutional Neural Network (R-CNN) Models

CENTERNET					
			CENTERNET_MOBILENET_V2_FPN		
SUBCLASS	**DETECTIONS**	**AVG. CONFIDENCE**	**AVG. IoU**	**AVG. INFERENCE TIME(s)**	**Soma de TIME (s)**
CC_back	489	96,52%	0,94	0,32	154,51
CC_front	459	95,92%	0,94	0,34	155,32
TOTAL	**948**	**96,23%**	**0,94**	**0,33**	**309,83**
			CENTERNET_RESNET50_V1_FPN		
SUBCLASS	**DETECTIONS**	**AVG. CONFIDENCE**	**AVG. IoU**	**AVG. INFERENCE TIME(s)**	**TOTAL TIME(s)**
CC_back	489	96,26%	0,95	0,66	324,26
CC_front	467	95,97%	0,95	0,68	317,96
TOTAL	**956**	**96,12%**	**0,95**	**0,67**	**642,21**
			CENTERNET_RESNET101_V1_FPN		
SUBCLASS	**DETECTIONS**	**AVG. CONFIDENCE**	**AVG. IoU**	**AVG. INFERENCE TIME(s)**	**TOTAL TIME(s)**
CC_back	494	96,32%	0,95	0,97	480,25
CC_front	475	96,13%	0,95	0,99	468,33
TOTAL	**969**	**96,23%**	**0,95**	**0,98**	**948,58**

Table 2: Evaluation Results for the CenterNet Models

EFFICIENTNET

EFFICIENT_DET0

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	500	98,90%	0,91	0,44	218,46
CC_front	475	99,03%	0,92	0,44	210,12
TOTAL	**975**	**98,96%**	**0,92**	**0,44**	**428,58**

EFFICIENT_DET1

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	495	97,68%	0,91	0,71	350,76
CC_front	456	97,20%	0,90	0,72	326,16
TOTAL	**951**	**97,45%**	**0,91**	**0,71**	**676,92**

EFFICIENT_DET2

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	501	98,53%	0,91	1,03	515,27
CC_front	474	98,26%	0,92	1,06	500,21
TOTAL	**975**	**98,40%**	**0,92**	**1,04**	**1015,48**

Table 3: Evaluation Results for the EfficientDet Models

SSD

SSD_MOBILENET_V1_FPN

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	494	98,68%	0,93	0,83	407,89
CC_front	482	99,09%	0,94	0,83	398,64
TOTAL	**976**	**98,88%**	**0,94**	**0,83**	**806,52**

SSD_MOBILENET_V2_FPNLite

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	466	96,31%	0,93	0,36	169,04
CC_front	471	96,61%	0,94	0,36	170,27
TOTAL	**937**	**96,46%**	**0,93**	**0,36**	**339,31**

SSD_RESNET50_V1

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	502	98,14%	0,93	1,21	606,80
CC_front	476	98,27%	0,94	1,22	581,26
TOTAL	**978**	**98,21%**	**0,94**	**1,21**	**1188,06**

SSD_RESNET101_V1

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	492	98,21%	0,94	3,12	1537,02
CC_front	481	98,24%	0,94	3,14	1508,77
TOTAL	**973**	**98,22%**	**0,94**	**3,13**	**3045,79**

SSD_RESNET152_V1

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	495	98,10%	0,93	4,58	2268,96
CC_front	484	97,99%	0,93	4,61	2228,96
TOTAL	**979**	**98,05%**	**0,93**	**4,59**	**4497,92**

Table 4: Evaluation Results for the SSD Models

This page is intentionally left blank.

Appendix B - Evaluation Results for the Darknet Environment

YOLO

YOLOv3-tiny

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	372	97,21%	0,86	0,16	60,37
CC_front	309	96,47%	0,82	0,16	48,82
TOTAL	**681**	**96,87%**	**0,84**	**0,16**	**109,19**

YOLOv4-tiny

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	451	98,67%	0,88	0,17	78,29
CC_front	460	98,50%	0,87	0,17	79,12
TOTAL	**911**	**98,58%**	**0,87**	**0,17**	**157,41**

YOLOv3

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	499	99,24%	0,88	1,63	812,24
CC_front	470	99,31%	0,86	1,63	764,46
TOTAL	**969**	**99,27%**	**0,87**	**1,63**	**1576,70**

YOLOv4

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	501	99,63%	0,89	0,93	467,90
CC_front	488	99,66%	0,88	0,93	455,95
TOTAL	**989**	**99,64%**	**0,89**	**0,93**	**923,85**

YOLOv3-SPP

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	486	98,88%	0,84	0,84	409,76
CC_front	453	99,08%	0,85	0,84	382,36
TOTAL	**939**	**98,98%**	**0,85**	**0,84**	**792,12**

CSRESNEXT50-SPP-PANET

SUBCLASS	DETECTIONS	AVG. CONFIDENCE	AVG. IoU	AVG. INFERENCE TIME(s)	TOTAL TIME(s)
CC_back	508	99,06%	0,88	0,74	373,49
CC_front	485	98,86%	0,88	0,74	357,45
TOTAL	**993**	**98,96%**	**0,88**	**0,74**	**730,94**

Table 5: Evaluation Results for the YOLO family and other architecture in Darknet Environment.