NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

# Implementation of Deep Learning models for Information Extraction on Identification Documents

Internship at Biometrid

Henrique Eduardo Espadinha Renda

Internship report presented as partial requirement for obtaining the Master's degree Program in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

**NOVA Information Management School**

**Instituto Superior de Estatística e Gestão de Informação**

Universidade Nova de Lisboa

# IMPLEMENTATION OF DEEP LEARNING MODELS FOR INFORMATION EXTRACTION ON IDENTIFICATION DOCUMENTS

by

Henrique Renda

Internship report presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

**Advisor:** *Prof Doutor Roberto Henriques*

**Co Advisor***:* Lucas Soares

# STATEMENT OF INTEGRITY

As a member of the Biometrid team and master´s student at NOVA Information Management School, I pledge to uphold the highest standards of integrity and ethical conduct in the completion of my internship report. This includes ensuring the accuracy and truthfulness of all information presented, properly citing any sources used, and refraining from plagiarism or any other form of academic dishonesty. I will also maintain confidentiality and respect any privacy or sensitive information that may be shared during the course of my internship. I understand that any violation of these principles may result in disciplinary action, and I am fully committed to conducting myself in an honest and trustworthy manner throughout my internship.

*Lisbon, February 2023*

February 2023

# DEDICATION

I dedicate this report to my family and friends, who have been a constant source of inspiration and motivation. In particular, I would like to express my deepest appreciation to my parents and girlfriend for their unwavering support, encouragement, and motivation that have been instrumental in helping me persevere through the challenges and celebrate the triumphs. Additionally, I extend my gratitude to my mentors for providing invaluable guidance and support throughout the research process. This report represents the culmination of years of hard work, and I am honored to share it with those who have played a significant role in shaping my academic and personal growth.

# ABSTRACT

The development of object detection models has revolutionized the analysis of personal information on identification cards, leading to a decrease in external human labor. Although previous strategies have been employed to address this issue without using machine learning models, they all present certain limitations, which artificial intelligence aims to overcome. This report delves into the development of a deep learning-based object detection capable of recognizing relevant information from Portuguese identification cards. All the decisions made during the project will be accompanied by a detailed background theory. Additionally, we provide an in-depth analysis of Optical Character Recognition (OCR) technology, which was utilized throughout the project to generate text from images. As the newest member of the Machine learning Team of Biometrid, I had the privilege of being involved in this project that led to the improvement of the current approach that does not leverage machine learning in the detection of relevant sections from ID cards. The findings of this project provide a foundation for further research into the use of AI in identification card analysis.

# KEYWORDS

Artificial Intelligence; Machine Learning; Artificial Neural Networks; Computer Vision; Object Detection Models; Optical Character Recognition (OCR)

# INDEX

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| **AMA** | Agência para a Modernização Administrativa |
| **ANN** | Artificial Neural Networks |
| **AP** | Average Precision |
| **AUC** | Area Under the Curve |
| **CNN** | Convolution Neural Networks |
| **CV** | Computer Vision |
| **DL** | Deep Learning |
| **FNN** | Feedforward Neural Networks |
| **FPS** | Frames per second |
| **IoU** | Interception over Union |
| **mAP** | Mean Average Precision |
| **ML** | Machine Learning |
| **OCR** | Optical Character Recognition |
| **R-CNN** | Recurrent Neural Networks |
| **RNN** | Recurrent Neural Networks |
| **ROI** | Region of Interest |
| **SVM** | Support Vector Machines |
| **TF** | TensorFlow |
| **TFOD** | TensorFlow Object Detection API |

# 1. INTRODUCTION

With the technological advances registered in recent years, it can be acknowledged that we are currently in an era where data is being produced at an unprecedented rate, compelling businesses to rely on automated methods for data analysis. Retrieving useful insights from this amount of data is crucial, so involuntarily companies in this situation notice that it is mandatory to develop advanced algorithms that can summarize, classify, extract important information, and convert them into an understandable form (V. Y., Mariano et al., 2002).

To address this issue, there has been a huge increase in the usage of Artificial Intelligence (AI) algorithms to guide business decisions. Machine learning (ML) models and Artificial Neural Networks (ANN), in particular, have demonstrated outstanding performance in resolving complex challenges across a wide range of industries, effectively revolutionizing the way companies approach problem-solving (A. Zulkhaizar, 2023).

For example, in computer vision tasks like object detection and semantic segmentation, deep learning models have been successfully used in applications such as autonomous driving. This technology is rapidly advancing, bringing systems closer to possessing a level of intelligence comparable to humans in certain situations. According to the current Nvidia CEO, Jensen Huang, "Deep Learning can train a car to drive, and ultimately perform far better, and more safely, than any human driver could do behind the wheel.". This was the goal of Huang's company when developing hardware for the upcoming autonomous cars (J. Huang, 2017).

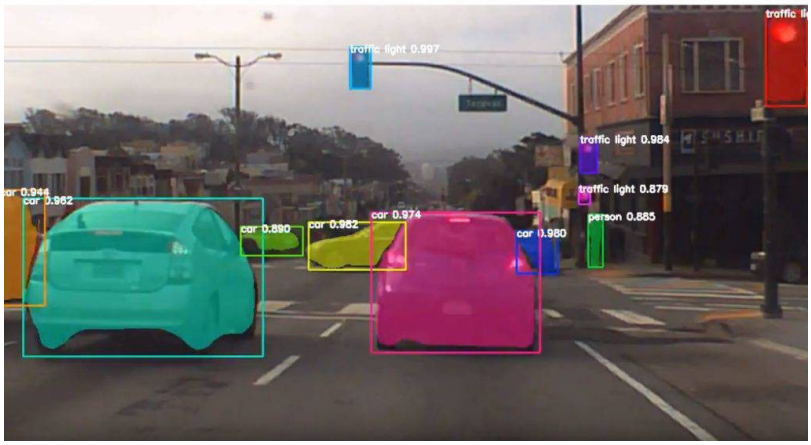


Figure 1 - Usage of Deep Learning models in autonomous cars[1]

In the example above, object detection models offer a solution for identifying relevant items in the road that allow the vehicle to drive safely. This ability renders them highly efficient for activities such as perception, decision-making, and vehicle control. Nevertheless, their usefulness extends beyond this use case scenario, as they have the capability to analyze and retrieve information from a given frame.

As another example, throughout this report, we'll highlight the experience of a Junior Machine Learning Engineer responsible for a project that led to the improvement of Biometrid's machine learning infrastructure behind its OCR solution for information retrieval for Portuguese identification

[1] **Source**: https://www.cbinsights.com/research/startups-drive-auto-industry-disruption/

cards. The responsibilities regarding the collection, processing, and analysis of information will be described in detail.

We'll start by introducing the problem addressed and clarify the challenges faced, regarding the development of an object detection model capable of recognizing relevant sections from the Portuguese identification cards.

Next, we'll do a concise explanation of how the company is composed alongside its objective, before delving into more specific and technical aspects regarding the theory behind the project. Finally, this report will end up by clearly exposing all the practical implementations of the project alongside the final results.

### 1.1. COMPANY OVERVIEW

Figure 2 - Biometrid's Logo[2]

Created in 2015 under the name Polygon, the current Biometrid emerged as an IT company with headquarters established in Porto that promised to revolutionize the way we communicate with digital systems. At a time when with one click it is possible to open a bank account, register on a social network or even subscribe to a service it is no surprise that the digital footprint of all internet users has been growing exponentially. With that in mind, Biometrid is committed to delivering a range of tools that facilitate the onboarding, authentication, and validation of personnel in companies of all sizes.

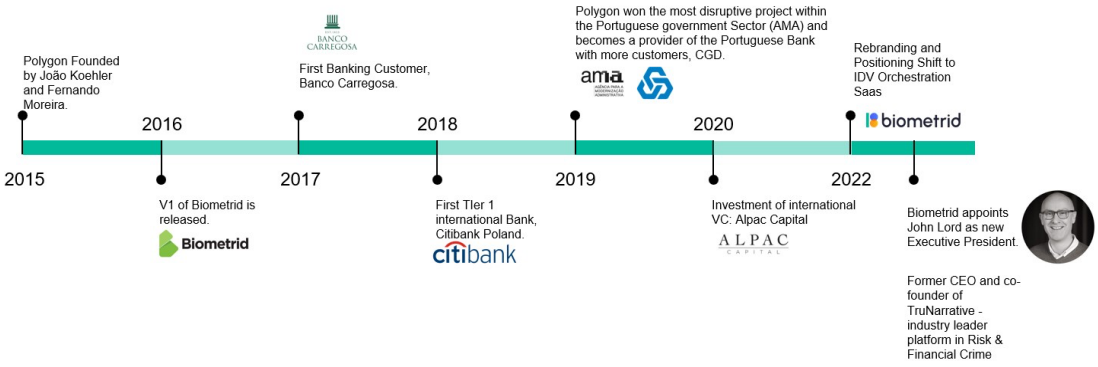


Figure 3 - Biometrid's Milestones

Biometrid offers a range of solutions to simplify the Know Your Customer (KYC) processes. The company's flagship product, which shares the same name, streamlines these time-consuming activities, which are essential to safeguard financial institutions against fraud, corruption, money laundering, and terrorism financing.

[2] **Source**: https://biometrid.com/

Compared to its main competitors, Biometrid's product takes a unique approach. It uses a Drag&Drop methodology, which provides customers with autonomy and flexibility to build personalized verification processes. The product enables users to open bank accounts remotely, verify Proof-of-Life using personal documents, and generate digital signatures for public services.

The company has established partnerships with banking institutions, insurance companies, and Portuguese Government departments. It has also expanded its operations to international markets, including Poland, Hungary, Colombia, Angola, and Mozambique. Biometrid's goal is to continue expanding its services and reach into new sectors and regions worldwide.

For the ML team the main responsibility is to create and maintain all the ML models that support its solutions, especially the Optical Character Recognition (OCR) production pipeline for retrieving information from identification cards. The focus is divided into two: on one side improving the OCR pipeline, which would receive images of ID cards or passports, and from there extract information that could identify users in the future. On the other side, researching and developing new ML tools for the OCR infrastructure.

### 1.2. Problem Definition

As mentioned before, systems equipped with artificial intelligence algorithms are now more present than ever in our daily lives. This paradigm shift means that processes that required a lot of manual work are now being replaced by models that simulate human behavior.

One of the several examples where this change is noticeable is in the registration process. In the past, regardless of the industry, to perform the onboarding of a new client this one needed to fill out a form with his personal information manually and wait for the responsible staff to transfer the information into a dedicated database. This implied a rather lengthy process with several dedicated parts and a complex infrastructure. Luckily, to streamline the process, a new method has emerged for extracting this information without requiring manual entry from either the customer or staff (M. Ryan, & N. Hanafiah., 2015).

A new system using optical character recognition (OCR) has been proposed to extract customer information from identification cards instead of manual data entry. This system, in the first instance, is powered by an object detection model responsible for processing an ID card image and retrieving the coordinates of the sections related to personal data as the result. These image coordinates are then used to crop the original image and feed those outputs into an OCR application that recognizes the text in the picture and converts it into machine-readable and editable text.

Problem Definition:

***Create a robust solution for retrieving relevant information from Portuguese Identification cards using an object detection model and OCR techniques.***

### 1.2.1. Case Study

This chapter will examine the application of the selected approach in the Biometrid OCR pipeline to automate the process of retrieving personal information from Portuguese ID cards. By leveraging cutting-edge optical character recognition technology, the pipeline aims to streamline the data input process and provide accurate and efficient results. This subsection will delve into the details of this strategy and its impact on the process of retrieving personal information from ID cards.

The growth of members of the ML team led to a new project which has the objective of improving the way the OCR data pipeline analyses images. Currently, the extraction process contains several layers of pre-processing:

1. The documents are exposed to a rotation model that orients them horizontally in an automatic way.
2. Next, this resulting image will serve as input for a segmentation model, which aims to remove all surrounding elements from the photo and generate cropped images.
3. These images are then converted from an RGB Format to Grayscale to improve the performance of the binarization that will be applied to the image lately.
4. After multiple morphological image transformations, the coordinates from the regions of interest were detected by multiple traditional image processing techniques that use relative coordinates to capture informational sections.
5. The process ends just when all of the proposed areas are analyzed by an OCR engine.

This strategy presents itself as a very good alternative compared to a full-manual process, however, it's in many ways limited as its results are only viable in favorable conditions where the document has a capture angle and favorable light conditions.

As a company that consistently embraces new technologies, Biometrid extends this philosophy to every aspect of its product. So, the main goal of this project is to improve the OCR Pipeline by creating an artificial neural network capable of detecting all sections of the front of the Portuguese identification card. This way we hope to enhance the general performance of the OCR processes and establish a strategy that delivers favorable outcomes in all image scenarios.

### 1.2.2. Constraints and Limitations

In the realm of camera-based analysis of text and documents, there exist several challenges associated with capturing images, including low resolution, uneven lighting, distorted perspectives, non-planar surfaces, wide-angle lens distortion, cluttered backgrounds, difficulties with zooming and focusing (M. Ryan, & N. Hanafiah., 2015). Throughout the report, we will explain all the strategies and techniques that we will apply to solve some of those problems.

Besides that, we also find some constraints related to the dataset containing images of Portuguese Identification Cards. This is not publicly available due to GDPR restrictions, but fortunately, Biometrid had already obtained this type of data through a partnership with the "Agência para a Modernização Administrativa" (AMA) and regularly utilized it for other projects. So, even with limited data we still

manage to organize and curate approximately 3000 of these images to construct a consistent dataset capable of successfully training the model. Important to mention that these images contain personal information, and their public share is expressly forbidden.

### 1.2.3. Proposed Solution

To address the case study, always with the limitations and constraints in consideration, it was decided by the team that the best strategy to improve the detection of the sections of the front face of the Portuguese identification card would go through the training of a deep neural network capable of improving the previous results and address situations where the image quality is not favorable.

Previous studies suggest that various methods have been employed to tackle this issue. One commonly employed approach involves identifying saliencies in the image to locate documents inside a photo or video frame, without any prior knowledge about the document (F. Attivissimo et al., 2019). However, this strategy still presents some limitations mainly when the image contains noise, or the angle of capture is not favorable. The improved solution for the Biometrid OCR that will be presented throughout this report presents itself as a more advanced approach to this problem. It starts by using a detection model to locate the document, then crops the image to isolate the document from the background. After proper segmentation, the document is classified using a deep learning model.

Throughout this document, we aim to enhance the average success rate of detecting relevant elements in identification cards, compared to the results that were achieved in the past. The models tested in the development of this solution will be thoroughly evaluated and compared to the current production models. Through this comparison, we can gain a better understanding of the progress made in our development and its potential for future advancements.

Also, in the evaluation phase, we will focus on evaluating the inference latency since we believe that this will be of utmost importance. The duration between capturing an image and producing a result has a significant impact on delivering a smooth and seamless experience for our consumers and is therefore a top priority for the company.

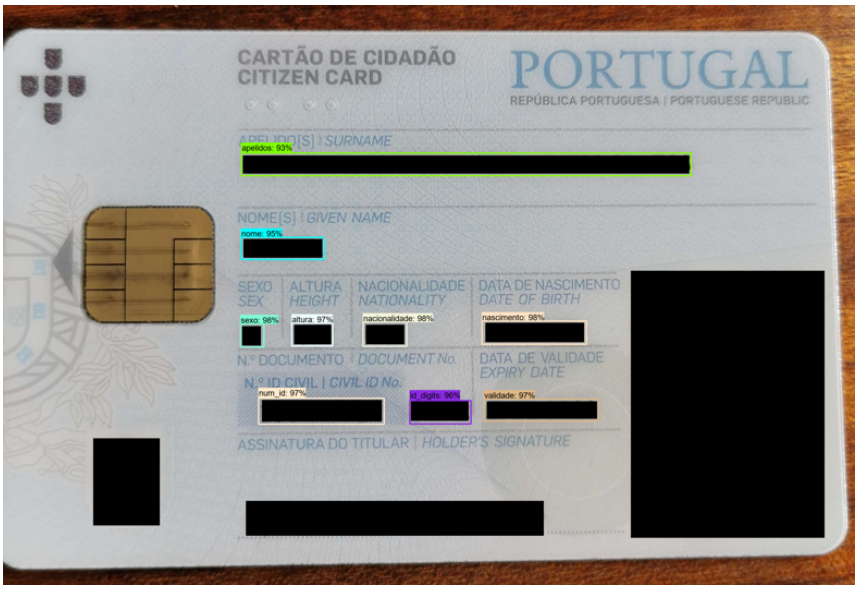


Figure 4 - Expected result

# 2. THEORETICAL FRAMEWORK

This chapter will provide a comprehensive overview of the technical knowledge that underpins the project recently completed by the Machine Learning team at Biometrid.

In Section 2.1, a comprehensive overview of Machine Learning will be presented. In Section 2.2, we will clarify the theoretical topics surrounding artificial neural networks, with a particular focus on convolutional neural networks (CNNs). In Section 2.3, we'll delve further into artificial neural networks by explaining the concept of Deep Learning. In Section 2.4, we will provide an overview of the various object detection model architectures that were tested over the past few months. In Section 2.5, we will discuss the evaluation stage of the project by presenting various metrics used to evaluate the performance of object detection models. Finally, in Section 2.6 of this report, we will provide an in-depth explanation of Optical Character Recognition (OCR) technology. This section will be crucial to understand the final phase of the project as it will cover the theory and technical aspects behind OCR.

## 2.1. MACHINE LEARNING

Machine Learning (ML) is one of the subject fields of artificial intelligence that makes use of statistical models so that, with the help of data, it generates reliable predictions. Systems that offer these capabilities offer solutions to previous common limitations, examples comprehend a huge space of possibilities from correctly identifying spam emails to image/video recognition (P. Ariwala, 2022). This makes it possible to find hidden insights and identify complex patterns without explicit programming, not just because of the ML development but also due to an increase in computational power registered in recent years (C. Janiesch et al., 2021).

According to a report by McKinsey, machine learning can create a potential economic impact of $2 trillion to $10 trillion per year across industries (J. Bughin et al., 2017) and is expected to grow at a compound annual growth rate of 38.8% from 2022 through 2029 and reach a value of $210 billion by the end of 2029. One of the key factors fueling this expansion is the growing acceptance of machine learning by major technology companies such as Apple, Microsoft, and many others across a wide range of industries, including healthcare, manufacturing, automotive, retail, advertising, automation, defense, financial services, and others (O. Farooq, 2022).

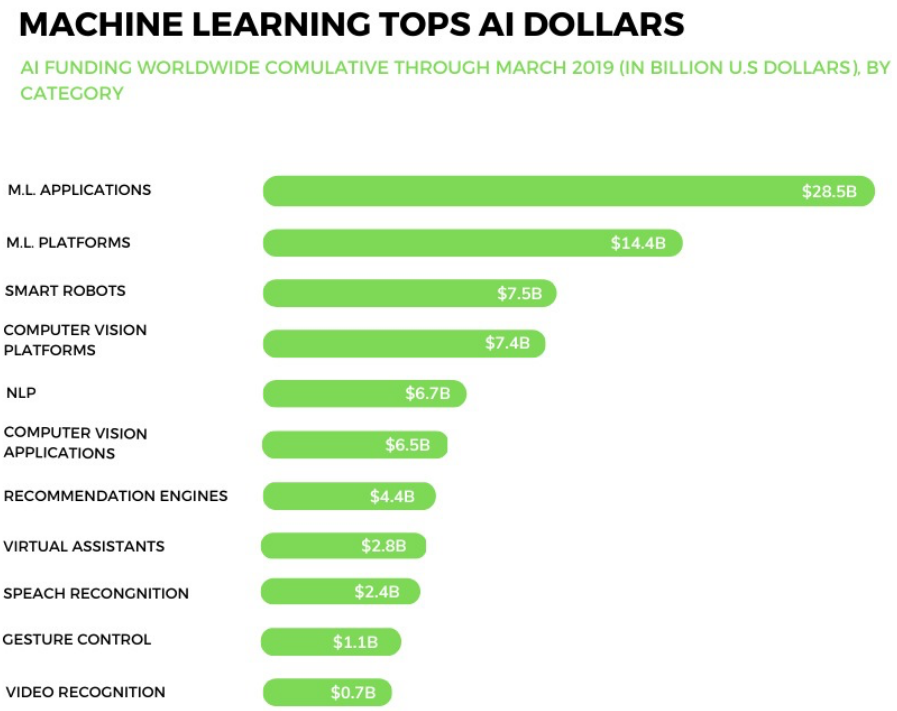


Figure 5 - Machine Learning investments by category[3]

Compared to conventional or classical programming, where systems were fed by input data, and a function with predefined rules, which generates results according to them. An intelligent mechanism, that is, a system equipped with an ML model, presents itself as an alternative to the conventional engineering approach which receives the same data, but unlike the previous one, it also receives the expected outputs and thus creates a mathematical model with parameters adjusted to the problem. (O. Simeone, 2018)

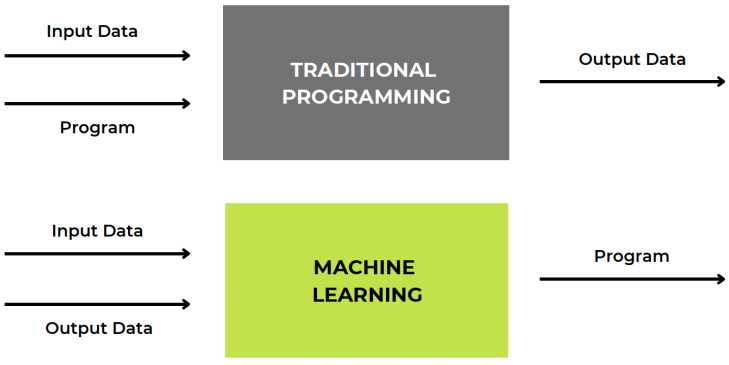


Figure 6 - Traditional Programming against Machine Learning approach[4]

The idea behind the development of any machine learning algorithm is the presence of a large number of data that present a considerable level of quality. An organized collection of data is denominated by dataset and usually present a group of features in a tabular format. These datasets

[3] **Adapted from**: https://www.statista.com/chart/17966/worldwide-artificial-intelligence-funding/

[4] **Adapted from**: Moroney, L. (2020). AI and Machine Learning for coders. O'Reilly Media.

can be used to train machine learning models for tasks such as classification, regression, clustering, and others.

On the other hand, image datasets contain information in the form of images, which are two-dimensional arrays of pixel values. These datasets are used to train machine learning models for tasks such as image classification, object detection, semantic segmentation, and others. Image datasets can be much larger in size compared to other types of datasets, as they often contain hundreds of thousands of images, each with multiple channels (e.g., red, green, and blue) and high resolution.

To evaluate the performance of a machine learning model, the data are usually divided into three groups: training, testing, and validation. Only in this way, it is possible to have a perception of the behavior of the model with data never processed in the training phase. The first group has the biggest amount of data, typically between sixty to eighty percent, and is used to train the model. The validation set is used as an evaluation at the end of each epoch to evaluate the model during training and help optimize its parameters and settings. Finally, the test set is presented as a way to evaluate the model with different data from those presented in the previous sets and have a final evaluation of the results.

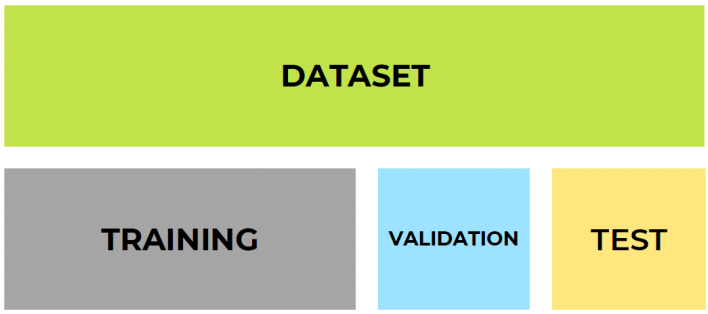


Figure 7 - Train, Validation & Test Split[5]

However, to conclude this chapter is worth mentioning that, despite the potential benefits in the industry, ML implementations have also challenges that need to be addressed. Being one of the most important examples to highlight the interpretation of accuracy. In most cases, a model that presents a high accuracy is considered a good one, and the opposite can lead to harm. For example where facial recognition systems are used in law enforcement. But highly accurate facial recognition systems can also pose risks to privacy and indicate the presence of mass surveillance (J. Fletcher, & A. Kostiainen., 2022).

Another important topic related to accuracy and the ability of AI to perform correct predictions is its usage in areas that don't have an objective ground truth and need external human judgment. The impact of an incorrect output depends on the context. For instance, predicting false occurrences of cancer can increase costs, but on the other hand, failing to predict a true result can have a much more impact on people's life and delay the treatment. Another common example of this is using ML in a judicial context, false positives may send innocents to jail, while false negatives may make it harder to convict criminals. (J. Fletcher, & A. Kostiainen., 2022).

[5] **Adapted from**: https://community.alteryx.com/t5/Data-Science/Holdouts-and-Cross-Validation-Why-the-Data-Used-to-Evaluate-your/ba-p/448982

Overall, is important to note that ML technologies are rapidly shaping a new future for the industry's panorama. Companies embracing this type of AI have the potential to gain a competitive edge and improve their bottom line. However, it is important for companies to understand the challenges and limitations of machine learning, and to invest in the necessary expertise and resources to ensure the success of their machine learning initiatives.

### 2.1.1. Supervised Learning

Based on a given problem and the available data, different types of Machine Learning algorithms can be used, each with its advantages and disadvantages. The first step is to understand which strategy to use to analyze the data. This data has great relevance in the results and can be classified as labeled data, that is, data that have one or more classes and allow their grouping or unlabeled data. In this project, we will focus on the first type of data which is characterized as data that contains input and output features.

For a full understanding of this project is crucial to understand the theoretical aspects of supervised learning. This type of machine learning model receives input data (x) and output data (y). The goal is to discover the parameters of a function that will generate the best predictions when faced with information that has never been processed before. This is an algorithm that needs to receive labeled data, acquire a deep knowledge of them based on its parameters, and create a cause-effect relationship.

We can divide the application of this type of machine learning into two categories, classification, and regression. In the case of classification, the objective is to predict a categorical result for each of the data, based on its independent variables. In regression problems, the expected output value is of the continuous type.

## 2.2. ARTIFICIAL NEURAL NETWORKS & CNNS

**Artificial Neural Networks (ANN)** are statistical models that are directly inspired by the structure and function of the neurons in the brain. The fundamental units of neural networks are also referred to as neurons, nodes, or artificial neurons. A group of connected neurons creates networks that are trained to perform a variety of tasks, such as recognizing patterns in data, making predictions, or making decisions. This can be achieved by adjusting the weights of the connections between the nodes based on example inputs and their corresponding outputs (K. O'Shea, & R. Nash., 2015).

An ANN is modeled by overlaying several layers of artificial neurons, or computational units, to receive inputs and then transfer them onto the next layer. The basic architecture consists of a network that has three types of neuron layers: input, hidden, and output layers (A. Abraham, 2005). However, there are some configuration variants, depending on the outputs and the data processing approach. Some common types of neural networks are Feedforward Neural Networks (FNN), Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN). However, throughout this report, we'll just focus on CNNs since this type will be the only one that will be used to address our problem.

**Convolutional Neural Networks (CNN)**, which are also called "ConvNets", is a type of ANN that, like the others, consists in neurons that learn through self-optimization. The main difference between this architecture and traditional ANN architecture is that this one is designed to deal with visual data.

CNN process images or videos and extracts relevant features directly from each pixel value without requiring any hand-engineered features or previous knowledge about the world.

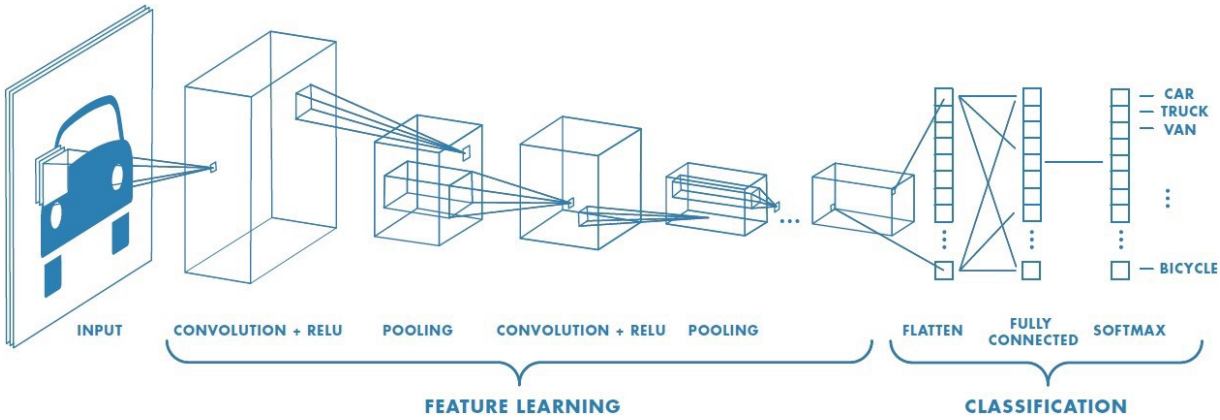


Figure 8 - Convolution Neural Network Architecture[6]

To understand CNNs is important to know how its architecture is composed and what the types of layers that take part of it are. This type of ANN includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A common structure consists of repetitions of several convolution layers and a pooling layer, followed by one or more fully connected layers.

### 2.2.1. Convolution Layers

A Convolution Layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations.

The first type of linear operation is convolution. This operation involves sliding a small matrix of weights (called a kernel or filter) over the input data (or input tensor) and computing an element-wise product of the weights and the values in the input volume at each position. The result for each pixel is then summed to obtain an output array called a feature map. This procedure is repeated by applying multiple kernels to form an arbitrary number of feature maps that extract different features from the input tensors (R. Yamashita et al., 2018).

Two key hyperparameters that define the convolution operation are size (I x J) and the number of kernels (K). The size is typically $3 \times 3$, but sometimes $5 \times 5$ or $7 \times 7$. The number of kernels that are stacked on top of each other will set the depth of the feature maps and the complexity of the detection.

[6] **Source**: Horak, K., & Sablatnig, R. (2019, August). Deep learning concepts and datasets for image recognition: overview 2019. In *Eleventh international conference on digital image processing (ICDIP 2019)* (Vol. 11179, pp. 484-491). SPIE.

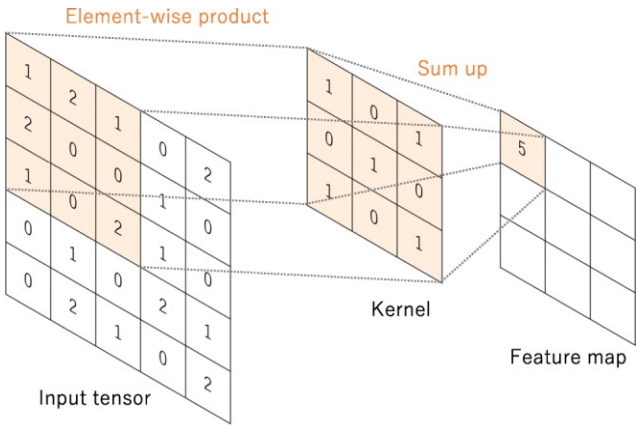


Figure 9 - Example of a convolution operation with a kernel size 3x3[7]

Important to mention that to preserve the spatial dimensions of the input data when it is convolved with a kernel, rows, and columns of extra elements are added to the edges of the input tensor in a technique called padding. Without padding, each successive feature map would get smaller after the convolution operation.

### 2.2.2. Pooling Layers

In addition to convolutional layers, CNNs also include pooling layers, which provide a down-sampling operation of the feature maps by taking the maximum or average value of a small region of the feature map and preserving important features learned by the convolutional layers. This has the effect of reducing the dimensionality of the data and making the network more robust to small translations or deformations of the input data.

There are several different ways to perform pooling, like Max, Sum, or Average, however the most common and preferred one is Max pooling, which extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values.

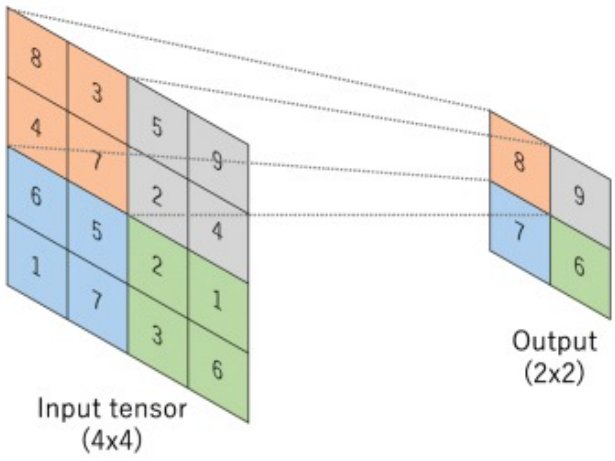


Figure 10 - Example of max pooling operation with a filter size of 2 × 2[8]

[7,8] **Source**: Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*, 611-629.

### 2.2.3. Fully Connected Layers

Finally, after applying several convolution and pooling layers, the output of the feature maps is generally transformed into a one-dimensional array of numbers (vector) and fed to one or more Fully Connected Layers, also known as dense layers.

The main idea behind these layers is the same that is implemented on feed-forward neural networks. They are used to retrieve the features from the last pooling or convolutional layer and use them to classify the input image into various classes based on the training dataset (D. Bhatt et al., 2021). Generally, in the more advanced ANN architectures, the last few layers are fully connected layers with the same number of nodes as the number of possible outcomes.

First is important to understand that a flattened vector will be the input layer that will be coupled to a first Fully Connected Layer meaning that each input neuron is connected to all the neurons of that layer (A. Alexandari, M. Shrikumar, & A. Kundaje., 2017). The same will happen between the first layer and the second one until the end of the network creating a Fully Connected Network. In this network, the connections between the layers are called weights and these are trainable parameters that our model needs to learn to associate features to a particular category.

The training process is done by applying a non-linear function called the activation function. This function decides whether a neuron should be triggered or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. A bias parameter will always be present in the network and is added to the weighted sum of the inputs to each node. It is a constant value that shifts the activation function of the node to help the network to better model the underlying data by allowing the neurons to have a non-zero output even when all of the inputs to the neuron are zero. This can be useful for modeling certain types of patterns in the data, such as trends or seasonal effects. The bias can also help to improve the convergence of the network during training, by providing a non-zero starting point for the optimization algorithm.

Fully connected layers are easy to implement and can learn a wide range of functions. However, they are also the second most computationally expensive, behind convolution layers, because they require a large number of parameters to be learned (J. Janke, M. Castelli, & A. Popovič.,2019) and can be susceptible to overfitting if the network is not sufficiently regularized. As a result, they are often used in combination with other types of layers, such as dropout layers, to improve the performance and generalization ability of the network.

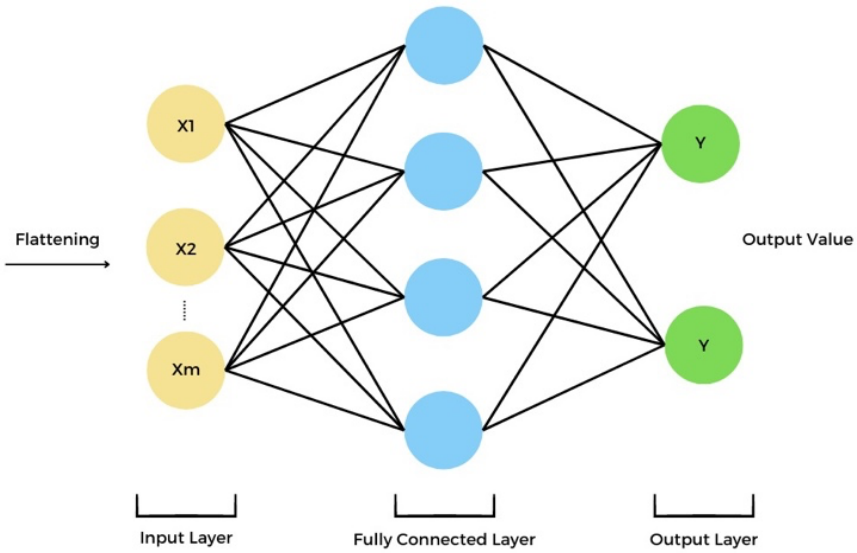


Figure 11 - Example of a neural network with fully connected layers[9]

The performance of an artificial neural network is determined by its architecture, the training algorithm used, and the quality and quantity of the training data. By adjusting these factors, it is possible to create neural networks that are highly effective for a wide range of tasks. However, training neural networks can be a complex and computationally intensive process, requiring specialized knowledge and expertise.

Nevertheless, as datasets become larger and more complex, there is a growing need for more sophisticated learning processes to effectively analyze them. This has led to increased research into developing more advanced neural network architectures. In the upcoming chapter, we will explore this area of deep learning that focuses on creating deep neural networks. We will also focus on how these models can be applied to our specific use case, highlighting their potential to improve performance.

## 2.3. DEEP LEARNING

**Deep Learning** is another subfield of machine learning, that only focuses on designing artificial neural networks with three or more layers making it possible to perform accurate decisions based on large and complex amounts of data. As an approximation of how the human brain behaves, this technology becomes possible to carry out recognition tasks and analysis of complex patterns. This offers useful possibilities, not just for computer vision tasks, but also for other types of tasks without the need for extensive programming or manual feature engineering.

By using a network of layers as architecture, where each layer can be thought of as the state of the computer's memory after executing another set of instructions in parallel (I. Goodfellow et al., 2016), systems powered by deep learning algorithms can process multiple iterations of data processing, resulting in a continuously improve their accuracy and performance.

[9] **Adapted from**: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection

The algorithms are typically trained using large amounts of data and powerful computational resources, such as GPUs. This means that training deep learning models can be time-consuming and computationally intensive and requires specialized knowledge and expertise.

Important to mention, for the full understanding of this project, that there is a field of AI that relies very much on deep learning to develop this kind of model to understand the contents of digital data, such as images or videos. This field is known as **Computer Vision** (CV), and the goal is to create deep artificial neural networks capable of process and understanding information in a way that is similar to humans and performing tasks like object recognition, image, and video analysis, medical imaging, etc.

CV image-related models can be seen as a highly versatile solution, however, it's worth noting that these models can only perform tasks that they were trained to execute. This introduces a major limitation when exposed to new tasks that require a different set of data. Meaning that the algorithms need huge amounts of dedicated data specifically for each project to produce good results. Our days, this is not a deal breaker since images are available online in bigger quantities than ever, but the fact that these images need to be labeled for many occasions can reveal expensive labeling work that can only be done by humans (S. Mavrikis et al., 2021).

In the next section, we will focus on the theoretical knowledge of object detection algorithms. This is one of the most popular computer vision use cases and represents the foundation of the solution for this work. This will be important to understand how the Machine Learning team in Biometrid developed the object detection algorithm that is capable of retrieving sections of the Portuguese identification card.

## 2.4. Object Detection

Object detection is a computer vision type of task that combines the tasks of identifying the location of one or more objects in a frame and classifying them. It is a crucial component for many applications and is required for unattended monitoring systems that need the ability to detect, identify, and, in some cases, track moving objects. Examples such as self-driving cars, image search engines, and robotics can be cases where this type of detection needs to present a higher monitoring performance even in complex environments (H. Zhu et al., 2020).

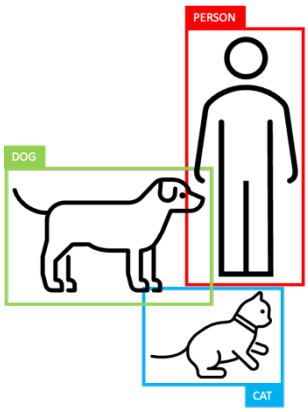


Figure 12 - Example of the expected output of an object detection model trained to detect dogs, cats and humans.

It's important to understand the difference between object detection and image recognition tasks. While both tasks involve analyzing images to identify objects, the key difference lies in the output they generate. Object detection tasks aim to predict bounding box coordinates and assign a label to each detected object in an image. This involves drawing one or more boxes around each object and providing information about its location, scale, and pose. On the other hand, image recognition tasks only focus on identifying the presence of an object in an image or video, without providing information about its location. In other words, image recognition is a simpler task that involves identifying objects without the need to localize them (Fritz Labs, 2018). While both tasks are important in computer vision, object detection is more complex and requires more advanced techniques to accurately identify and locate objects within an image.

In recent years, the fast development of artificial neural networks has gained a lot of interest in the context of object detection tasks providing powerful learning abilities. By learning parameters themselves, ANN can now deliver a high degree of accuracy compared to previous computer vision strategies. Following the publication of the original paper in 2012, which achieved huge attention from the community due to their results on the ImageNet classification benchmark (Krizhevsky A. et al., 2012.), many ANN architectures have emerged and proposed different strategies to help improve previous results (H. Hakim, A. Fadhil., 2021).

The following paragraphs will present and compare variants that we tested throughout the past months and are important for the full understatement of this project. These variants give us different views to solve the problem of detecting and correctly classifying objects in an image or video.

### 2.4.1. Region-based Convolution Neural Networks (R-CNN)

In 2014, Girshick et al. proposed a new method called Region-based Convolution Neural Networks (R-CNN) that achieved a high level of accuracy regarding a variety of applications, including image classification, object detection, and facial recognition. The basic idea behind R-CNNs is to use CNNs to learn features from an input image, and then use those features to classify and localize objects within the image (Girshick R. et al., 2014).

To do this, R-CNNs follow a two-step process:

1) Region Proposal: In the first step the algorithm proposes a set of potential bounding boxes or regions of interest (ROIs) that may contain an object of interest. These bounding boxes are typically generated using a sliding window approach, where the algorithm slides a fixed-size window across the image and proposes a bounding box at each location.

2) Feature Extraction: Next, R-CNN takes use of a CNN to extract features from the image within each bounding box. These features are then fed into a Support Vector Machines (SVM) classifier trained to predict the class of the object and the location of the object within the bounding box.

Despite producing good results, R-CNNs still presented some disadvantages mainly regarding the high time consumption to train and make inferences caused by the multiple-stage pipeline.

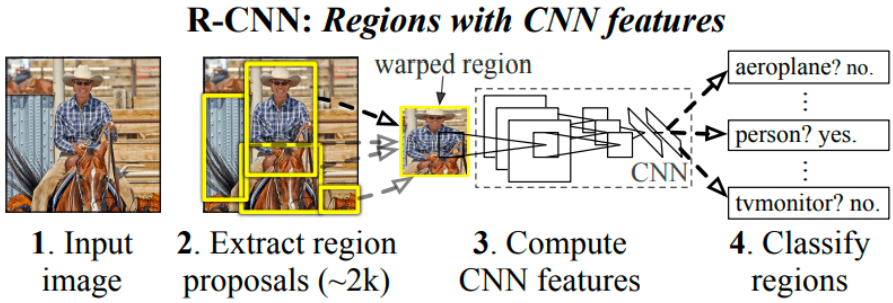


Figure 13 - R-CNN Stages[10]

### 2.4.2. Faster-RCNN

In 2015, Microsoft Research (Ren S. et al., 2016) introduced **Faster R-CNN** as an improvement over the original R-CNN algorithm and since then has become a popular choice for object detection due to its speed and accuracy.

With shared philosophy with R-CNNs, Faster R-CNNs aim to classify and localize objects within an input image using a combination of convolutional neural networks (CNNs) and region proposal algorithms. However, Faster R-CNNs introduce Region Proposal Networks (RPN) as a more efficient approach compared to the previous ones since it only uses one neural network to perform both the region proposal and the feature extraction steps, rather than using separate networks for each step.

In other words, the architecture of a Faster R-CNN algorithm begins with several convolution layers responsible for the feature extraction of the input image. Before the last pooling layer, a convolution RPN layer of size 3x3 is implemented and comes with the responsibility of receiving the feature maps and generating outputs to both a classifier, that determines the probability of a pixel to host a target object (proposals) and a regressor, that regresses the coordinates of the regions of interest. (S. Ren et al., 2015).

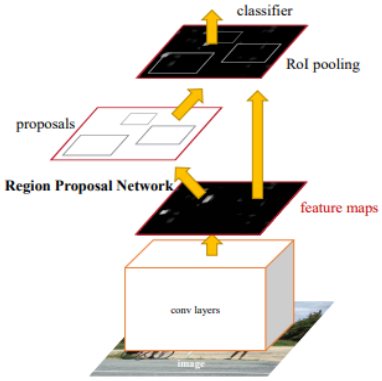


Figure 14 - Faster R-CNN's Architecture[11]

[10] **Source**: Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[11] **Source**: Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

At each sliding-window location, several proposals are predicted, and knowing that the regressor has 4*k coordinate outputs and the classification layer generates 2*k binary results the authors introduce the definition of “anchors” to define the central point of each sliding window. For any image, scale, and aspect ratio play a decisive role as important parameters to calculate the number of anchors (k). The developers chose 3 scales and 3 aspect-ratio as default parameters meaning that the maximum number of proposals per pixel is equal to 9, concluding that for the whole image, k = W*H*9.

According to the original paper, the loss function of the RPN is minimized by the binary class label (of being an object or not) of each anchor. To assign a positive label to an anchor this one has to have the highest Intersection-over-Union (IoU) overlap with a ground-truth box or has to have an IoU overlap higher than 0.7 (S. Ren et al., 2015).

### 2.4.3. Single Shot MultiBox Detector (SSD)

To address the high computational problems caused by the Faster R-CNN, other object detection systems, proposed different techniques that reduce the inference time while maintaining the same level of accuracy. So, by the end of 2016, C. Szegedy presented a paper on the implementation of Single Shot MultiBox Detector (SSD) reaching immediate breakthrough records in terms of performance and precision for object detection tasks compared to the previous models.

This model approach is based on a feed-forward convolutional network that produces a fixed number of bounding boxes and creates a scoring mechanism that evaluates the presence of objects in those boxes. The architecture of the SSD uses VGG-16 architecture as its base network inheriting in this way its strong performance in high-quality image classification tasks. From there, the main difference stands in the removal of the fully connected layers by a set of convolutional layers that decrease in size progressively and enable predictions at multiple scales. These layers are then followed by a non-maximum suppression step to produce the final detections.

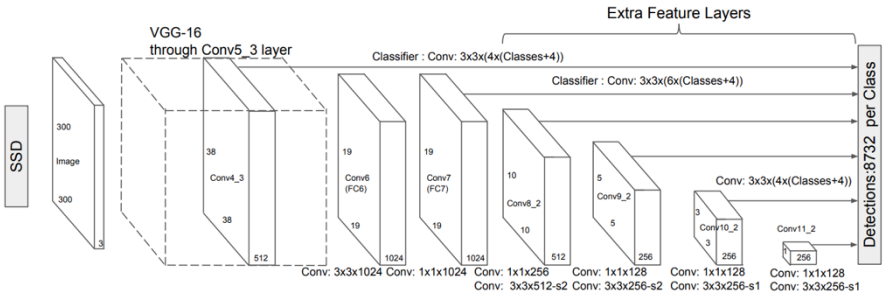


Figure 15 - SSD Architecture[12]

The SSD Model works as follows, the base network processes an input image that is divided into grids of various sizes, and, for each grid, detection is performed for different classes and different aspect ratios. After that, each of these grids is evaluated by a score that says how well the detected object fits in that particular grid. The non-maximum suppression step is as stated before, responsible for choosing the best final detection from the set of overlapping detections (W. Liu et al., 2016).

[12] **Source**: Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21-37). Springer International Publishing.

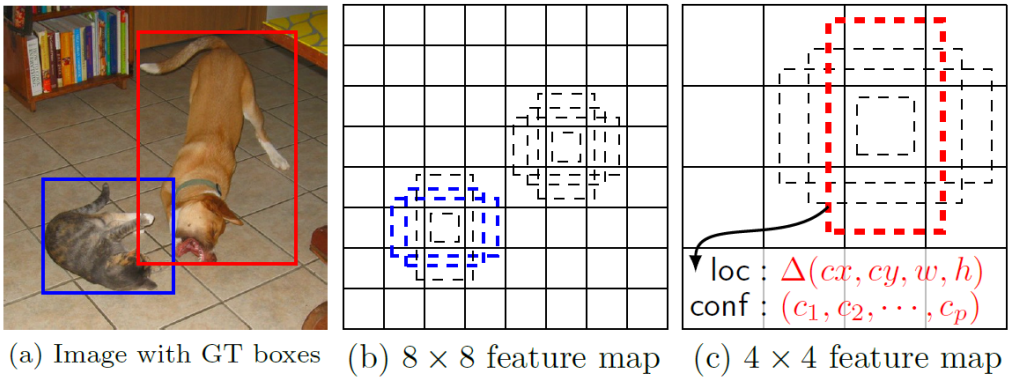


Figure 16 - SSD Framework[13]

In the year of its launch, the SSD model registered better results than the previous state-of-the-art detection algorithms, but in recent years, new variants of YOLO architecture and SSD keep competing for the award of best object detection model in the PASCAL VOC 2007 Challenge.

### 2.4.4. CenterNet

To reduce the steps performed by two-stage object detection architectures, like R-CNN or Faster R-CNN models, and to deal with the limitations of anchor-based models, CenterNet architecture was proposed in 2019 as an anchorless object detection algorithm that proves to be an effective lightweight option (K. Duan et al., 2019).

Taking as the baseline the CornerNet architecture, CenterNet is a one-stage detector that aims to improve to performance of its predecessor by using a triplet of keypoints, rather than a pair. This architecture reduces the analysis cost by only paying attention to the central information using a keypoint detection process.

In the given figure, the model represents each object using a center keypoint and a pair of corners. To generate a heatmap, CenterNet uses CNNs as a model backbone to generate bounding boxes, then count the number of bounding boxes that contain the center keypoint and uses this information to predict the likelihood of a central region in the image containing the same class of center keypoints. Each pixel in the heatmap represents the probability of an object center being present at the corresponding spatial location in the image. A high probability score indicates a high likelihood of an object center being present at that location, while a low probability score indicates a low likelihood of an object center is present. This approach of predicting the likelihood of object centers using a heatmap can improve the accuracy of object detection, particularly for small or occluded objects.

[13] **Source**: Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21-37). Springer International Publishing.

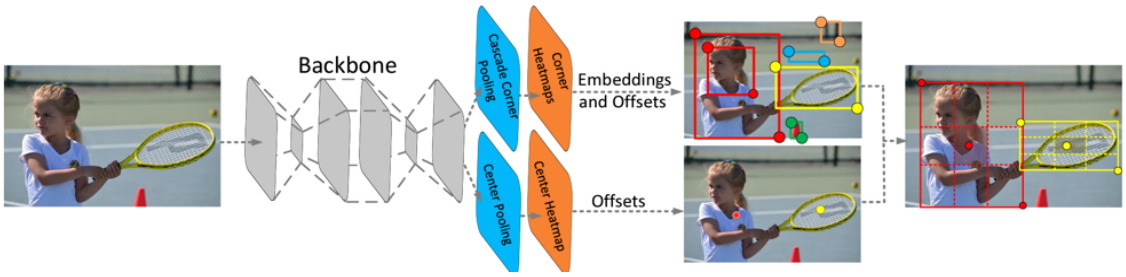


Figure 17 - CenterNet Architecture[14]

The authors presented an architecture that employs two key strategies to predict the geometric centers and corners of objects. The first strategy is called **Center Pooling** and starts by predicting the center keypoint by finding the maximum value in both horizontal and vertical directions and adding these two values together. This helps to obtain more recognizable visual patterns within objects, making it easier to perceive the central part of the proposal.

The second strategy is denominated **Cascade Corner Pooling**, which aims to find the corner points similarly to the previous one, but now by obtaining the maximum summed response in both the boundary and internal directions of objects on a feature map for corner prediction. Empirically, the authors claim that the results are more stable and robust to feature-level noises, leading to improved precision and recall.

In summary, the CenterNet model is a state-of-the-art object detection architecture that achieves high accuracy with low computational costs. The model predicts object centers directly using a heatmap, which reduces the complexity of the model and improves detection accuracy for small and occluded objects. The model has achieved state-of-the-art results on several benchmark datasets and is widely used in both research and industry for various computer vision applications.

## 2.5. Evaluation Metrics

Evaluation metrics play a crucial role in the development and deployment of machine learning models. Building a machine learning model can be seen as a recursive pipeline where the responsible team trains a model, then evaluate it using evaluation metrics, finetune specific model hyperparameters, and repeat this procedure until the desired performance is achieved. So, it is extremely important to carefully choose and track the right evaluation metrics to ensure that the model is performing well, to perform a model comparison, and to meet the desired objectives.

In this chapter, we will discuss the theoretical background of the evaluation metrics used throughout the project and explore how to choose and interpret these metrics in a production environment. We will also discuss the importance of considering the specific characteristics and goals of the task when selecting an evaluation metric, and the role of evaluation metrics in model selection, optimization, and monitoring.

[14] **Source**: Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569-6578).

Since we are dealing with an object detection algorithm, is expected to evaluate the model by comparing its output with ground-truth bounding boxes generated by a human operator responsible for manually labeling the objects and defining their boundaries (V. Y. Mariano et al., 2002). From this analysis, it's common to use evaluation metrics like accuracy, precision, recall, and many others to understand how well the model can distinguish between different class labels and the trade-offs between false positive and false negative predictions.

Since 2010 challenges like VOC PASCAL Challenge, COCO, ImageNet Object Detection Challenge, and Google Open Images Challenge have emerged as competitions that aim to evaluate new implementations and are now seen as valuable benchmarks to test object detection models in specific scenarios by using real-world annotated datasets (M. Everingham et al., 2009). These competitions contribute to the definition of standard evaluation procedures within the scientific community by using popular metrics like **Average Precision (AP)**, **Intersection over Union (IoU)**, or creating their variants to rank the models.

In the next sections, we'll try to clarify the metrics previously mentioned in a simplified and organized strategy where the reader is guided by a sequential process illustrated in the figure below. It should be noted that each step can be considered as a standalone method of evaluation, but they are interdependent and build upon one another.

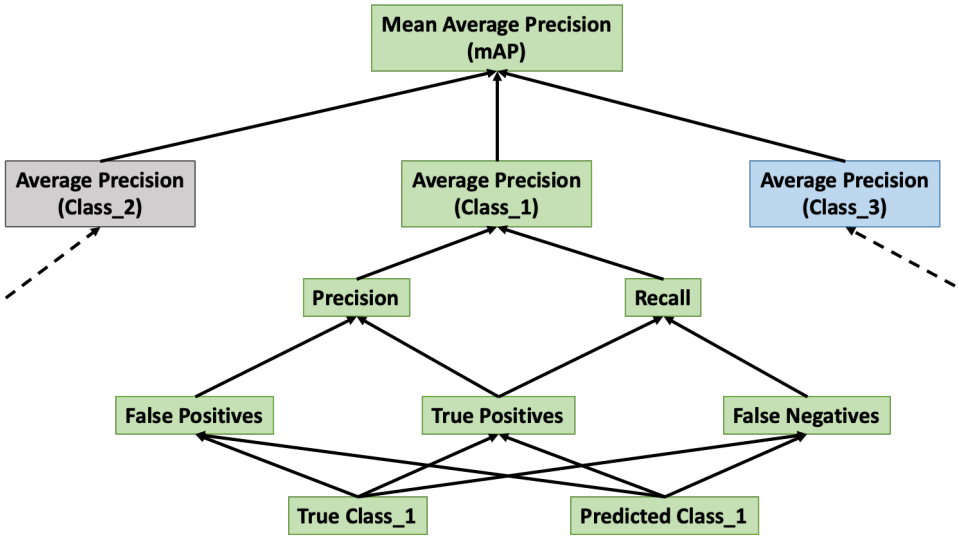


Figure 18 - Object Detection Evaluation Process

### 2.5.1. Confusion Matrix

This table comes up as a useful tool for an initial understanding of the types of errors made by a model and for comparing the performance of different models. It's constructed by comparing the predicted class labels with the true class labels of the data.

| | | True Class | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted Class | Positive | TP | FP |
| | Negative | FN | TN |

Table 1 - Confusion Matrix[15]

Looking at the table above, a confusion matrix will be composed by four main areas that help to understand the strengths and weaknesses of a model and for identifying areas for improvement. More precisely, each prediction can be classified as:

- **True positive (TP)** if the prediction contains a correct detection of a ground-truth bounding box.
- **False positive (FP)** if the algorithm performs an incorrect detection of a nonexistent object or a misplaced detection of an existing object.
- **False negative (FN)** if the ground-truth object is not detected.

Bear in mind that, within the realm of object detection, there is no such thing as a **False Positive (FP)** as there are an infinite number of bounding boxes that should not be identified within a given image.

### 2.5.2. Intersection over Union (IoU)

Based on the Jaccard Index, this metric individually evaluates the overlap between a pre-annotated ground truth bounding box (*gt*) and the one predicted by the model (*pd*) having in consideration a specific threshold value.

IoU score ranges between 0 and 1 where the closer the two boxes the higher the score, meaning that detection output with maximum IoU value is considered to have a perfect overlap with the ground truth bounding box.

$$IoU = \frac{area(gt \cap pd)}{area(gt \cup pd)} = \frac{Area\ of\ Overlap}{Area\ of\ Union} =$$

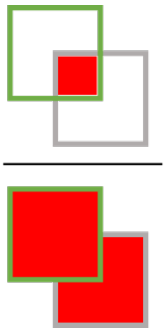

Figure 19 - Formula and Representation of the IoU calculation[16]

[15] **Adapted from**: Jeppesen, J. H., Jacobsen, R. H., Inceoglu, F., & Toftegaard, T. S. (2019). A cloud detection algorithm for satelliteimagery based on deep learning. Remote sensing of environment, 229, 247-259.

[16] **Adapted from**: Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.

By computing the IoU score for each detection, it's important to define a threshold ($\alpha$) to convert real-valued scores into classifications.

### 2.5.3. Precision and Recall

This performance benchmark act as a measure of how closely a predicted value agrees with the true value of a quantity. It is defined as the ratio of the number of correctly predicted positive cases (positive predictions containing an IoU greater than the threshold) to the total number of predicted positive cases (correctly identified objects + false positive predictions). In other words, it is a measure of the proportion of positive cases that are actually positive.

$$Precision = \frac{Number\ of\ True\ Positive\ Predictions}{Number\ of\ True\ Positive\ Predictions + Number\ of\ False\ Positive\ Predictions}$$

Figure 20 - Precision Formula

Precision is often used in conjunction with another metric called **Recall (or Sensivity)** to evaluate the performance of a model. Precision is typically more relevant when the goal is to limit the number of false positive predictions, whereas recall is more relevant when the goal is to identify as many positive cases as possible. This way we can define Recall as the metric that measures the ratio of the number of correctly predicted positive cases to the total number of actual positive cases. In other words, it is a measure of the proportion of actual positive cases that are correctly identified by the model.

$$Recall = \frac{Number\ of\ True\ Positive\ Predictions}{Number\ of\ True\ Positive\ Predictions + Number\ of\ False\ Negative\ Predictions}$$

Figure 21 - Recall Formula

Important to mention that, in the presence of imbalanced datasets, where one class is rare, it's important to focus on the recall of the model.

### 2.5.4. F1-Score:

This is the harmonic mean of precision and recall, with a higher score indicating a better balance between the two. The F1 score is defined as:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Figure 22 - F1-Score Formula

The F1 score is often used in imbalanced classification tasks, where it is important to achieve a balance between precision and recall. It is also useful when the cost of false positive and false negative predictions is not the same, as it allows for the weighting of these costs to be incorporated into the evaluation metric.

### 2.5.5. Average Precision

In object detection algorithms, it is essential to strike a balance between precision and recall. The traditional evaluation metric, which is the F1 score, can only provide a single score for a given

threshold, making it less informative in situations where the threshold varies. To address this issue, the Average Precision (AP) metric was introduced, which has become a widely used evaluation metric for object detection algorithms. AP considers the number of true positives, false positives, and false negatives, making it particularly useful when the data is imbalanced. By computing the area under the precision-recall curve (AUC), AP provides a comprehensive summary of the trade-off between precision and recall across different thresholds, thus providing a more informative evaluation metric for object detection algorithms.

To calculate the AP, the precision-recall curve is first computed by varying the threshold values and plotting the precision on the y-axis and recall on the x-axis. The AUC is then computed, which ranges from 0 to 1, with higher values indicating better performance. AP provides a more precise evaluation metric compared to traditional metrics, making it a valuable tool for object detection algorithms. As such, it has become a standard metric used in many object detection benchmarks and competitions.

### 2.5.6. Mean Average Precision (mAP)

Multiclass Object Detection models face the challenge of detecting multiple object classes in an image. As a result, a metric called Mean Average Precision (mAP) was developed as an extension of the AP metric to calculate the average precision for each class. mAP calculates the AP for each class in the dataset and then averages the AP values for all the classes to obtain the final mAP score. The resulting value ranges from 0 to 1, with 1 indicating the best possible performance. mAP is a useful metric as it provides an overall view of the model's performance and helps to identify which classes the model is performing well on and which classes it is struggling with, providing insight into where the model needs improvement.

The simplicity of mAP has made it an essential benchmark for evaluating the performance of object detection models in competitions and challenges. The mAP allows for easy comparison of different models and provides a fair evaluation metric for all models, regardless of the number of classes in the dataset. This makes it an effective tool for researchers and developers to assess the performance of their models and identify areas for improvement.

## 2.6. Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a technology that provides the capacity of converting handwritten, typewritten, or printed text into machine-readable images. This offers numerous applications that can be used to improve workflow efficiency. Common examples are industries like legal, banking, and healthcare that are currently leveraging OCR technology to simplify their operations and streamline their processes (A. Singh, K. Bacchuwar, & A. Bhasin., 2012).

Unlike humans, machines do not have the capability to recognize text or characters easily from an image, which is why significant research efforts have been put into developing OCR techniques. OCR is a complex problem due to the numerous languages, fonts, and styles in which text can be written, as well as the complex rules of languages. As a result, OCR requires the integration of various computer science disciplines, such as image processing, pattern classification, and natural language processing, to overcome these challenges (N. Islam, Z. Islam, & N. Noor., 2017).

The process of OCR involves a series of distinct phases. The first one, denominated by image acquisition, entails obtaining an image from an external source such as a camera or scanner, and

transforming it into a format that is compatible with computer processing. This is a crucial step in the OCR process as it sets the foundation for accurate and efficient optical character recognition.

Once the image has been acquired, the subsequent step in the OCR process is known as preprocessing. During this phase, a variety of techniques can be utilized to enhance the quality of the image. These techniques may involve removing noise, setting thresholds, and extracting the baseline of the image. By improving the image quality, the OCR algorithm can achieve a higher level of accuracy when it comes to recognizing the text or characters contained within the image. This phase is critical to ensuring the overall effectiveness of the OCR process.

Moving to the next phase, the next step in the OCR process will focus on character segmentation. The goal of this phase is to isolate individual characters within the image so that they can be properly identified by the recognition engine. While simple techniques such as connected component analysis and projection profiles may be sufficient in certain cases, more advanced segmentation techniques are necessary for complex situations where characters may be overlapping, broken, or obscured by noise within the image. By effectively separating the characters, the OCR system can accurately recognize the text and produce an output that is faithful to the original document.

The segmented characters are then processed to extract different features. Based on these features, the characters are recognized. Different types of features that can be used extracted from images are moments etc. The extracted features should be efficiently computable, minimize intra-class variations, and maximizes inter-class variations.

The segmented characters are then processed in order to extract different features that will be used for character recognition. Based on these features, the characters are recognized. The features extracted from the characters should be efficiently computable and should minimize intra-class variations (variations within the same class of characters) while maximizing inter-class variations (variations between different classes of characters). Various types of features can be extracted from the characters, such as moments and other image-based features. After that, the features of segmented images are assigned to different categories or classes using different types of character classification techniques.

Finally, post-processing techniques can be performed to improve the accuracy of OCR systems. These techniques utilize natural language processing, and geometric and linguistic context to correct errors in OCR results.

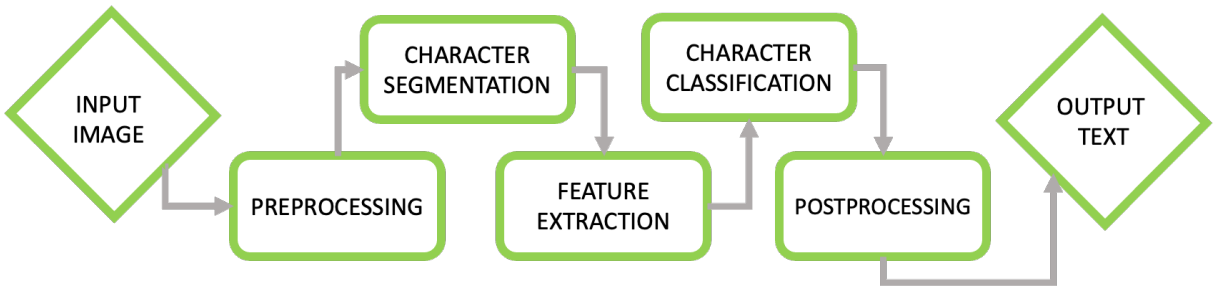


Figure 23 – General OCR Workflow[17]

Over the years, OCR technology has undergone tremendous improvements, and modern systems can now recognize various fonts, sizes, and styles of text. Despite this, the accuracy of OCR systems still depends on factors such as image quality, text complexity, and language. Fortunately, advancements in artificial intelligence and machine learning have resulted in substantial improvements in OCR accuracy, leading to more reliable and precise results (A. Ranjan, V.N.J. Behera, & M. Reza., 2021).

Nowadays, there are numerous OCR tools available that have made their usage more common in different environments. Some of the most popular OCR tools include Tesseract, developed by Google, Microsoft Cognitive Services by Microsoft, and Amazon Textract. For our project, we have decided to utilize the Pytesseract Python library, which provides an easy-to-use interface to leverage all the features and capabilities of Tesseract. The Pytesseract library offers a straightforward and efficient way of integrating OCR functionality into Python-based applications, making it a great choice for our project needs.

[17] **Adapted from**: Mudiarta, I. M. D. R., Atmaja, I. M. D. S., Suharsana, I. K., Antara, I. W. G. S., Bharaditya, I. W. P., Suandirat, G. A., & Indrawan, G. (2020, April). Balinese character recognition on mobile application based on tesseract open-source OCR engine. In Journal of Physics: Conference Series (Vol. 1516, No. 1, p. 012017). IOP Publishing.

# 3. METHODOLOGY

## 3.1. TOOLS AND TECHNOLOGIES

In this chapter, all the tools and technologies used throughout this project will be presented. We will explore the reasons for their selection, enumerate alternatives available in the market, and understand the importance of these tools and technologies for the success of the project. The chapter will cover the various libraries, frameworks, and platforms that have been employed to facilitate the process of building an object detection model, from data preparation to model training and deployment. By the end of this chapter, readers will have a comprehensive understanding of the tools and technologies used in the project, and how they contribute to the overall objective of the project.

### 3.1.1. TensorFlow

TensorFlow (TF) is a powerful open-source software library for machine learning and deep learning, developed by researchers and engineers working on the Google Brain Team. At its core, TF uses data flow graphs to represent any computation. A data flow graph is a directed acyclic graph (DAG) where the edges represent the flow of data, and the nodes represent operations. This makes it easy to implement machine learning algorithms, such as neural networks (M. Abadi et al., 2016).

TensorFlow also provides a wide range of tools for building, training, and deploying machine learning models using a variety of platforms including CPUs, GPUs, and TPUs. However, the development team maintains a collection of pre-made models on a platform called TensorFlow Hub, which can be used for tasks like object detection, image classification, and text generation.

For visualization purposes, TensorFlow has a visualization tool called TensorBoard, which allows developers to easily visualize and understand the behavior of their models during training, evaluation, and inference.

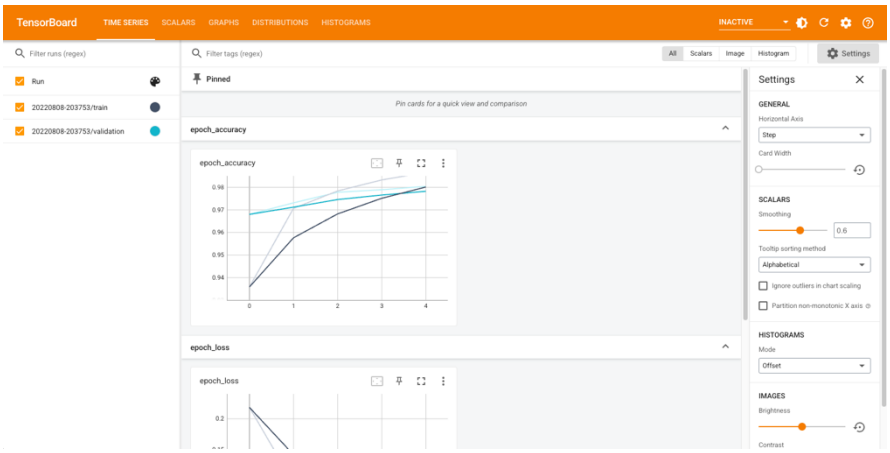


Figure 24 - TensorBoard Interface[18]

The main alternative to TensorFlow is PyTorch. This python library was developed by Facebook and defines itself as an easier-to-use tool easy for tasks such as image classification, natural language

[18] **Source**: https://www.tensorflow.org/tensorboard

processing, and generative models. The optimized integration with the most popular python libraries like NumPy, Pandas, and Matplotlib makes it a great choice for data scientists who want to leverage their existing knowledge and tools.

Biometrid uses this specific framework as its primary tool for training machine learning models in a production setting. However, the team is open to utilizing different tools and encourages the exploration of new options that may offer improved performance.

TF presents itself as the main tool for almost all the tasks that were performed during this project, and the full understanding of this library was central to being able to train, evaluate and deploy new algorithms for the extraction of elements of the identification card. In the next subsection, we'll explain in more detail a specific TensorFlow API for the creation of Object Detection models.

#### 3.1.1.1. TensorFlow Object Detection API

TensorFlow Object Detection API (TFOD API) presents as a framework for a specific build and deployment of object detection models using TensorFlow. It provides a collection of pre-trained detection models that are good baselines for out-of-the-box inferences.

The API also provides tools to fine-tune the model on a new dataset, or to train a new model from scratch. Additionally, TFOD API facilitates this workflow by including a collection of utilities for converting existing object detection datasets to the "TFRecord" file format, which is the input format for TensorFlow training and evaluation.

Once the model is trained, TensorBoard can also be used as a visualization tool to understand and debug your model during training, evaluation, and inference. In terms of the deployment of the model, this can be done in a variety of ways, including as a command-line tool, as a library that can be integrated into other Python code, or as a web service using TensorFlow Serving.

Overall, TensorFlow Object Detection API is a powerful tool for building and deploying object detection models, making it easy to train and deploy models using TensorFlow, and providing a collection of pre-trained models that include architectures like Faster-RCNN, SSD, and CenterNet.

### 3.1.2. Label Studio

Label Studio is a software platform for creating and managing data annotation projects. It is designed to help machine learning engineers, data scientists, and other professionals easily create and manage annotation projects.

With its user-friendly interface users can define the data that needs to be labeled, create annotation tasks, and invite other users to collaborate on the project. The platform supports a wide range of annotation tasks, including text classification, object detection, image segmentation, and more. Users can also customize the annotation interface to suit their needs, by creating custom forms, adding instructions, and setting up validation rules.

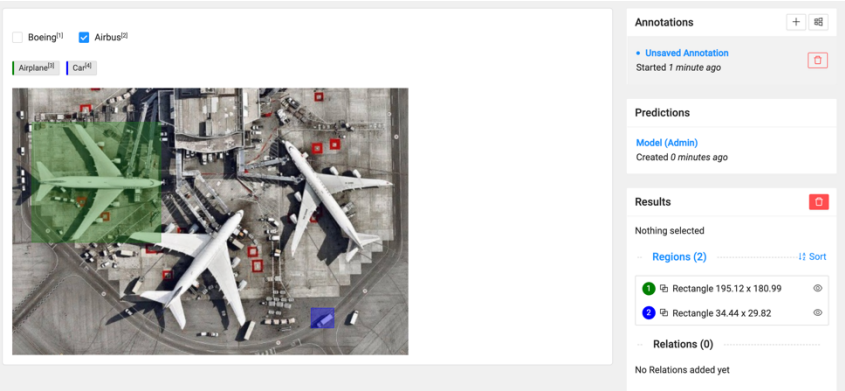


Figure 25 - Label Studio Interface[19]

Once the data is labeled, it can be exported in various formats, including JSON, CSV, and TFRecord, which makes it easy to use the labeled data in machine learning pipelines and other applications.

This was the tool used to perform labeling activities and build the dataset used since the beginning of the project. All of its advantages made it the preferred choice over its competitors like LabelImg and Labelme.

### 3.1.3. OpenCV

Open-Source Computer Vision (OpenCV) is a library of programming functions mainly aimed at real-time computer vision for multiple languages, including Python, Java, and C#. It was presented by Intel in 2000 and is now maintained by a non-profit organization called Willow Garage.

OpenCV provides a wide range of features for image and video processing, including image filtering, image transformation, object detection, and machine learning. Currently is widely used in industry and academia, and it is supported by a large and active community. It is also supported by many platforms, including Windows, Linux, and macOS, and it can be used on both desktops and mobile devices.

Throughout the project, this was used to manipulate the images and understand the best image conditions for the OCR engine.

### 3.1.4. Streamlit

Streamlit is an open-source framework used to build and share interactive data-driven applications. This relatively new Python-based library simplifies the development of machine learning and data science web applications. Streamlit has gained popularity among data scientists due to its intuitive interface, ease of use, and ability to develop applications quickly. With it, everyone that knows how to code in Python can easily create interactive dashboards and data visualizations without any web development experience.

Another feature of this library is its ability to render real-time data. Streamlit automatically updates the data and visualizations whenever there is a change in the input. This means that data scientists can work with their data in real time and get instant feedback on the changes made to the data.

[19] **Source**: https://labelstud.io

A variety of built-in widgets is provided allowing developers to create interactive user interfaces quickly. These widgets include sliders, buttons, and drop-down menus, which can be used to interact with the data and visualize the results in real time.

Streamlit has support for a wide range of data science and machine learning libraries, including NumPy, Pandas, Matplotlib, and Scikit-learn. Developers can easily integrate these libraries into their applications and leverage their functionalities to analyze and visualize data.

## 3.2. ID ELEMENTS PT PROJECT

Since starting at Biometrid, the objective was to expand the understanding of the subjects and technologies mentioned above. An initial experience with the business world and the field of machine learning outside of an academic setting has proven to be a valuable step in pursuing a career that aligns with a personal desire to continue learning and, in particular as a ML Engineer to comprehend the inner workings of systems utilizing artificial intelligence.

With the growth in terms of personnel inside the company and, in particular with the growth of the Machine Learning team, an improved version of the OCR Pipeline was proposed enabling a more accurate analysis of a high volume of identification card images from different countries daily. This data extraction, transformation, and loading process is an essential component of the Biometrid SDK as it plays a crucial role in enrolling new users and validating existing ones on the platform where it is integrated.

The procedure starts by acquiring images taken by a mobile device or webcam, and a classification algorithm is then employed to determine the type of document being analyzed, before proceeding to the next phase (card type model). The next step involves applying pre-processing methods to properly segment the images (segmentation model), isolating the relevant portion from the background, and adjusting the orientation of the document to a horizontal layout (rotation model).

The entire process described above is accomplished with the aid of multiple machine learning models that were trained and implemented previously. These models serve as a foundation for this project, and it’s important to enhance that every point mentioned in this chapter will have that attention.

Throughout the next chapters, we will go into detail on the various steps that compose this project consisting of finding the best object detection model architecture and training it to the point that is capable of successfully locating and identifying all the sections from the front face of the Portuguese identification card. It is extremely important to mention that reading the previous chapters regarding the theoretical background of this project is mandatory to fully understand the subsequent sections.

### 3.2.1. Data Processing

#### 3.2.1.1. Data Collection

After one week of gaining a comprehensive understanding of the company's operations and the specific project assigned to me, it became apparent that a dataset containing a significant quantity of images captured in various settings and featuring diverse arrangements was necessary to ensure that the model would be as robust as possible and avoid any overfitting issues.

To replicate the OCR pipeline in a local setting, it was necessary to properly segment and align the images in a horizontal orientation before they were processed by the detection model, to avoid any potential conflicts with the other components of the pipeline. To accomplish this, we initiated interactions with existing segmentation and rotation models that were already in use in the production environment.

In essence, we are examining object detection models with distinct objectives for the first time. Specifically, the segmentation model's goal is to partition an image into regions of interest (in this case, a single region) that correspond to an object or a class of objects. The output of the model will be a segmentation mask, which is effectively a binary image that categorizes each pixel to indicate the locations where objects of interest are present.

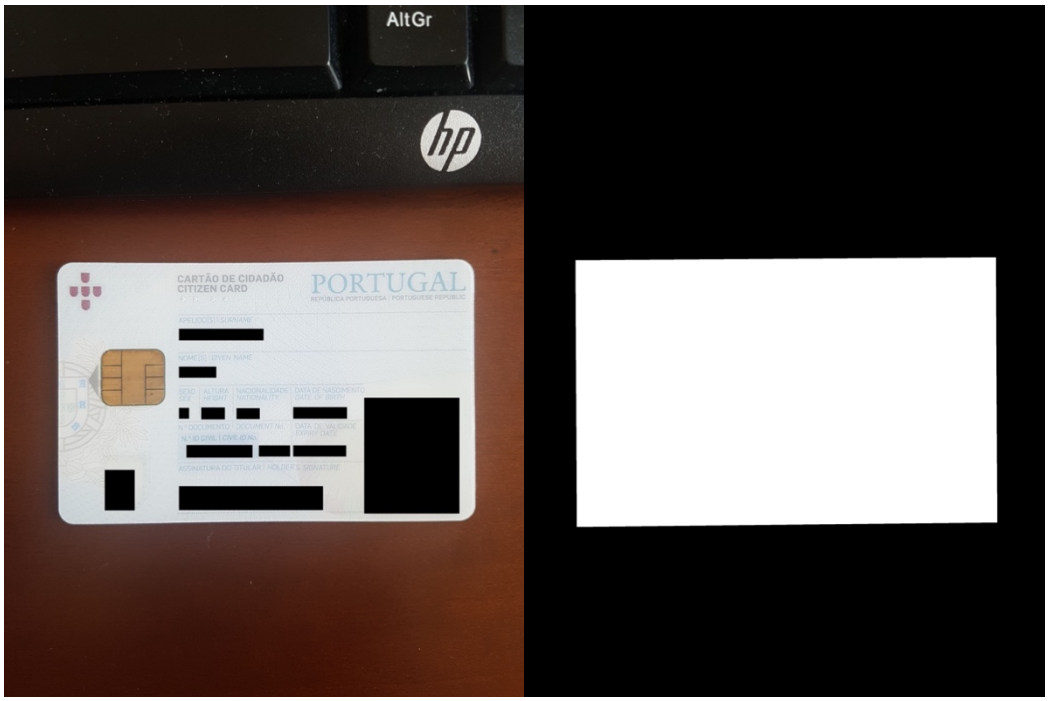


Figure 26 - Segmentation Model output

As for the rotation model, we utilize an automated method to rotate images based on the model's output. This returns a list of probabilities corresponding to each rotation operation, allowing us to apply the technique with the highest probability to achieve a successful outcome and align the image in a horizontal orientation.

With that in mind, the objective for each image was to utilize the segmentation model to automatically crop out most of the background. To achieve this, we wrote a Python script that, after properly initiating the model, would iterate through the entire directory of images and create a copy of each image, but with the background segmented out. This approach ensured that all data analysis and manipulation would be performed on the copy of the images, thus preserving the integrity of the original dataset.

However, upon collecting data, it became apparent that the rotation model would not need to be run locally as the majority of images were already in a horizontal layout. Despite this, it is crucial to

note that the alignment of the images is of paramount importance as a successful OCR outcome is highly dependent on it.

In summary, the majority of the two initial weeks at Biometrid were dedicated to research tasks, where the primary focus was to gain a comprehensive understanding of how the product functions, its components, the core objectives of the project assigned to me, and the methods for utilizing the models locally. Only after gaining this understanding, we were able to commence organizing a study dataset and begin testing the manipulations applied in the OCR pipeline locally.

#### 3.2.1.2. Data Annotation

The process of training an object detection model involves several steps, in our case, the second one is data labeling. After collecting a sufficient number of images, the goal is to manually identify and label all the objects of interest in each image. This manual process requires close attention to detail and the ability to identify the objects in the image. The desired result of the object detection model must be defined before starting the data labeling process, as any changes in the desired outcome, later on, will require re-annotating all the images.

For our specific task, we selected bounding boxes as the method for identifying the objects in the images. Bounding boxes are widely used in computer vision and consist of rectangular boxes defined by their edge coordinates. This type of annotation is used to identify objects in an image by surrounding them with these boxes. By using bounding boxes, we created a custom dataset that consists of images paired with the accurate coordinates of the objects, which will assist in both training and evaluating the model. These coordinates are saved in a specific XML file alongside other information about each image.

This stage was important to become familiar with various data labeling tools and dedicate a few days to evaluating and selecting the best one for the company. The goal was to establish a standard for data labeling. Biometrid aims to utilize open-source technologies with a strong developer community, to leverage the community in case any issues arise. Keeping this in mind, we surveyed the tools commonly used by machine learning teams at major technology companies and concluded that Label Studio would be the ideal option for this phase.

The tool presents itself as a data labeling platform known for its user-friendly design and customization options. Upon first use, you are presented with a comprehensive menu that allows you to customize and configure various aspects of the annotation process, such as the type of annotation, the classes for the bounding boxes, and more. With its robust customization capabilities and intuitive design, Label Studio is revealed to be an excellent choice for the general majority of data labeling tasks in the computer vision field.

This week's efforts culminated in a preliminary understanding of the project outcome and an introduction to previously unfamiliar tools. Although the days were relatively routine and involved repetitive tasks, they also served as opportunities for meetings with the machine learning team to plan future steps.

#### 3.2.1.3. Preparing the Workspace

After organizing the training dataset and performing the necessary annotations, we enter a phase of the project where we need to organize the workspace so that it is possible to test our data in several model architectures and run several experiments with different combinations of parameters.

With that in mind, the TensorFlow API dedicated to object detection tasks presents us with very well-written documentation where, through a few steps, it is possible to train models out-of-the-box or to train custom ones from scratch.

The first step involves dividing the dataset with a script provided in the documentation, and creating subsets as discussed at the end of chapter 2.1. This is a common practice where, in our case, we opt to split the data in a way that 90% is utilized for training and the remaining is reserved for evaluation (test set). Once the script has finished, two new folders were created in our working folder. To avoid the loss of any files, the script will not delete the original data that will be present where it previously was.

Next, another important component in the process of preparing the train of a detection model was the creation of a label map. It provides the mapping between class names and class IDs in the training dataset. This file has a .pbtxt extension used by the model to recognize objects in images, and also to decode the predictions made by the model during inference.

During training, TensorFlow requires the ground-truth object labels to be associated with each image in the training set. The label map is used to encode this information in a format that the model can understand. This allows the model to learn the relationships between the object classes and their corresponding bounding boxes in the training images.

During inference, the label map is used to decode the class IDs predicted by the model back into human-readable class names. Without the label map, the model's predictions would be difficult to interpret and would not provide meaningful results.

Then we move to the next step where, according to TensorFlow Object Detection API, we need to convert the entire dataset into a proprietary binary file format called TensorFlow Records (TFRecords). This is an efficient and convenient format used to receive large amounts of data during the training and evaluation of models and provides an optimized way to manipulate data inside TensorFlow’s data processing pipelines.

This also offers several advantages like portability and speed, since TFRecords can be used on different platforms and devices much faster than other formats, making it easy and lightweight to share datasets between different systems. To apply this conversion a python script was also provided in the official documentation.

### 3.2.2. Model Training

In order to obtain better results, we use transfer learning as a starting point to retrain a previously designed model. In this way, we take advantage of the complex structure of a model that was previously trained on a large volume of data and adjust some parameters according to our purpose. By fine-tuning the model for a new task, the model is able to leverage its prior knowledge, making training faster and more efficient. This approach is especially useful in situations where the amount

of data for a new task is limited, as transfer learning reduces the need for extensive data collection and labeling.

Luckily, TensorFlow offers a repository that contains several model architectures previously trained and accompanied by their evaluations when used in the COCO dataset. This way we started to research which ones met our needs seeking a model with consistent performance but at the same time not presenting a high inference time as this would negatively affect the processing time of the OCR pipeline.

To organize the project and observe improvements over time, we opt to divide it into versions. Each version would present a different group of models alongside data modifications that we believed that made sense according to the outputs retained. In this way, it was possible to evolve our work while delivering reports to the Biometrid team at the end of each month. Below is displayed a table illustrating all the models tested per version.

| | **Models** | **Observations** |
|---|---|---|
| **Version 1** | EfficientDet D1 | |
| | SSD ResNet50 | |
| | SSD ResNet101 | |
| | SSD ResNet152 | |
| **Version 2** | SSD ResNet152 1024x1024 | |
| | SSD ResNet50 v2 | Improved version with different hyperparameters |
| | SSD ResNet101 v2 | Improved version with different hyperparameters |
| **Version 3** | Faster R-CNN Inception | |
| | EfficientDet D2 | |
| | CenterNet HourGlass104 | |
| **Version 4** | Faster R-CNN Inception v2 | Improved version with different hyperparameters |
| | CenterNet Resnet50 | |
| | CenterNet Resnet101 | |

Table 2 - Model's architecture used per version.

It's crucial to emphasize that the model's key aspects have to be altered to adapt to the appropriate data format for our project. This involved modifying each model's configuration file to enhance the detection of objects based on the quantity and shape of the bounding boxes.

To measure the performance of our models, we focused on evaluating them using metrics explained previously such as mAP, Recall, and IoU. For this purpose, the evaluation metrics group was set to “coco_detection_metrics” in the eval_config parameter. Although the library provides other evaluation metrics, we found this to be the most appropriate for our project.

The model training procedure was thoroughly documented, making it possible to initiate the first training iteration after inserting the data and label maps. This gave us a preliminary understanding of the model's capabilities. The training process was monitored by not only the console logs but also through the use of TensorBoard. This allowed for comprehensive tracking of the training evolution.

### 3.2.3. Model Fine-Tuning

In the initial iteration of the models, we solely relied on the training and test data converted to TFRecords format without making any modifications to the model architecture. Nonetheless, we conducted research to identify the optimal parameters suited for our scenario. This led to the next phase of the solution's development known as finetuning the model, where during each training session, small modifications are made to the model's architecture to enhance its efficacy.

In this section, the main objective is to discuss the various modifications that the team made to the model's configuration file to obtain the optimal architecture for our particular problem. The focus will be on the dataset composition, and a detailed overview will be provided for each of the themes contained within the configuration file. Throughout this process, we will explain the reasoning behind all the choices and provide insight into the various options that are available through this library.

#### 3.2.3.1. Image Preprocessing

In the context of any computer vision project, it is absolutely crucial to have a full understanding of the input data that is being fed into the model. Without this understanding, it is impossible to design and train a model that is truly effective at the task at hand. Deep learning algorithms that power this kind of project are heavily dependent on the input data, and even small variations or inconsistencies in the data can greatly impact the performance of the model. It is therefore essential to carefully preprocess the data, ensuring that it is in a format that is compatible with the model's architecture.

Fortunately, the TFOD API provides a range of preprocessing tools that can be accessed right out of the box. In the initial section of the configuration file, users have the ability to exercise control over the image resizing process, including the shape of the resulting resized image. The library offers a considerable amount of flexibility concerning the various parameters that can be modified to achieve the desired results. However, it is important to note that before any modifications are applied, TensorFlow automatically handles the normalization of the images. This is an important feature that ensures consistency in the data and helps to optimize the performance of the model.

When configuring the strategy chosen to process the input image for our model, the API offers two distinct options: "**keep_aspect_ratio_resizer**" and "**fixed_shape_resizer**". The former option allows users to specify a minimum size while maintaining the aspect ratio of the original image. However, it is important to note that this option can sometimes result in extensively padded images, especially in cases where the original image is rectangular. In such cases, it may be more suitable to use the "fixed_shape_resizer" option, which resizes the image to a specified rectangle size defined by the

"height" and "width" parameters. For our specific use case, we have determined that the "fixed_shape_resizer" option is the optimal choice, as it allows us to maintain consistency in the size of the input images, which is critical for optimizing the performance of the model.

#### 3.2.3.2. Image Augmentation

Image augmentation is an essential technique that can greatly enhance the accuracy and robustness of the model. By applying various transformations and manipulations to the input images, augmentation procedures can help to address issues such as overfitting, insufficient training data, and class imbalance. For example, flipping, rotating, and cropping images can increase the variability and diversity of the training data while adjusting brightness, contrast, and color can help to account for variations in lighting conditions. Additionally, image augmentation can also help to mitigate the effects of occlusions, viewpoint changes, and other real-world factors that may impact the model's ability to accurately detect objects in new and unseen images.

Throughout the course of this project, we explored various of these methods to augment our dataset with synthetic images generated through the application of image manipulation techniques. Knowing that our images were accurately segmented and correctly oriented from the segmentation and rotation models, we made a deliberate effort to avoid modifying their orientation. Instead, we focused on applying techniques that could alter the color system of the data as you can check in the image below. This decision proved to be highly effective, as adjustments to factors such as hue, contrast, saturation, and brightness allowed us to minimize the model's sensitivity to color. As a result, the model was able to detect objects more accurately across a wider range of colors, leading to improved overall performance. The use of these techniques demonstrates the importance of careful and strategic data augmentation in the context of object detection and underscores the critical role that it plays in optimizing model performance.

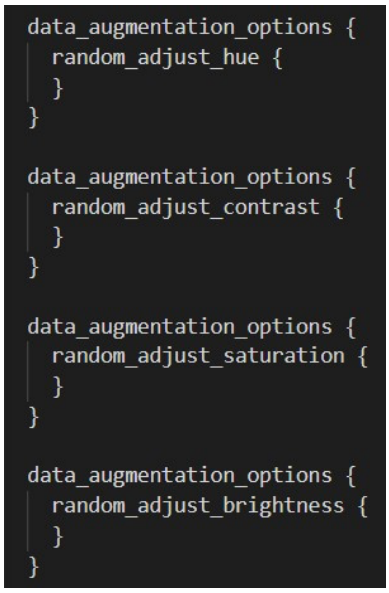


Figure 27 - Image Augmentation techniques used for the CenterNet configuration file

While the techniques described above can be highly effective for augmenting image data in our context, it's worth noting that the TensorFlow Object Detection API offers a much broader range of augmentation options. As you can see in the image below, there are a wide variety of different

techniques that can be used to manipulate and enhance image data, ranging from simple color adjustments to more complex transformations like rotation, flipping, and distortion. By carefully considering the unique characteristics and requirements of your specific use case, you can leverage these powerful tools to create highly customized and effective data augmentation strategies that can significantly improve the accuracy and performance of your object detection models.

```
message PreprocessingStep {
  oneof preprocessing_step {
    NormalizeImage normalize_image = 1;
    RandomHorizontalFlip random_horizontal_flip = 2;
    RandomPixelValueScale random_pixel_value_scale = 3;
    RandomImageScale random_image_scale = 4;
    RandomRGBtoGray random_rgb_to_gray = 5;
    RandomAdjustBrightness random_adjust_brightness = 6;
    RandomAdjustContrast random_adjust_contrast = 7;
    RandomAdjustHue random_adjust_hue = 8;
    RandomAdjustSaturation random_adjust_saturation = 9;
    RandomDistortColor random_distort_color = 10;
    RandomJitterBoxes random_jitter_boxes = 11;
    RandomCropImage random_crop_image = 12;
    RandomPadImage random_pad_image = 13;
    RandomCropPadImage random_crop_pad_image = 14;
    RandomCropToAspectRatio random_crop_to_aspect_ratio = 15;
    RandomBlackPatches random_black_patches = 16;
    RandomResizeMethod random_resize_method = 17;
    ScaleBoxesToPixelCoordinates scale_boxes_to_pixel_coordinates = 18;
    ResizeImage resize_image = 19;
    SubtractChannelMean subtract_channel_mean = 20;
    SSDRandomCrop ssd_random_crop = 21;
    SSDRandomCropPad ssd_random_crop_pad = 22;
    SSDRandomCropFixedAspectRatio ssd_random_crop_fixed_aspect_ratio = 23;
    SSDRandomCropPadFixedAspectRatio ssd_random_crop_pad_fixed_aspect_ratio = 24;
    RandomVerticalFlip random_vertical_flip = 25;
    RandomRotation90 random_rotation90 = 26;
    RGBtoGray rgb_to_gray = 27;
    ConvertClassLogitsToSoftmax convert_class_logits_to_softmax = 28;
    RandomAbsolutePadImage random_absolute_pad_image = 29;
    RandomSelfConcatImage random_self_concat_image = 30;
    AutoAugmentImage autoaugment_image = 31;
    DropLabelProbabilistically drop_label_probabilistically = 32;
    RemapLabels remap_labels = 33;
    RandomJpegQuality random_jpeg_quality = 34;
    RandomDownscaleToTargetPixels random_downscale_to_target_pixels = 35;
    RandomPatchGaussian random_patch_gaussian = 36;
    RandomSquareCropByScale random_square_crop_by_scale = 37;
    RandomScaleCropAndPadToSquare random_scale_crop_and_pad_to_square = 38;
  }
}
```

Figure 28 - Image Augmentation techniques provided by Tensorflow Object Detection API[20]

### 3.2.3.3. Post-processing

Post-processing is another indispensable step in any object detection project, as it enables the model to generate accurate and meaningful results from the raw output of the detection algorithm. In the context of object detection, post-processing typically involves analyzing the output of the model and applying various techniques to refine and filter the results. By carefully tailoring these post-processing techniques to the specific requirements of the project, it's possible to significantly improve the accuracy and precision of the object detection model, while also reducing the risk of false positives or other errors.

As we are facing an anchorless architecture, our primary concern at this stage was preventing overfitting and improving model performance. To achieve this, we focused on advanced approaches to learning rate control over time. Learning rate plays a critical role in determining the rate at which the model's internal parameters and weights are updated during training. A carefully chosen learning rate can significantly impact the model's convergence and its ability to find the optimal set of parameters for the given task. As such, we employed various strategies to tune the learning rate,

[20] **Source**: Tensorflow Model Garden Repository

including learning rate schedules and optimization algorithms. Through these techniques, we aimed to strike a balance between model convergence and generalization, ultimately leading to improved performance.

In our case, we employed the cosine learning rate decay schedule for controlling the learning rate over time. This approach enables the learning rate value to alternate between increasing and decreasing throughout the training process. Specifically, the learning rate is gradually decreased towards zero as the training progresses, which helps the model converge toward an optimal solution while reducing the risk of overfitting.

To correctly configure the learning rate schedule, it is important to focus on the following parameters and understand their impact of them on the performance of the model:

- **learning_rate_base**: parameter that defines the initial learning rate that will be used to train your model.
- **total_steps**: parameter that defines the number of total steps your model is going to train. Important to note that in the last steps of your training job, the learning rate scheduler will drive the learning rate value to be close to zero.
- **warmup_learning_rate**: the maximum value that the learning rate will reach before starting to decrease.
- **warmup_steps**: defines the number of steps that will be taken to increase the learning rate from learning_rate_base to warmup_learning_rate

### 3.2.4. Image Pipeline

Once we had properly trained the model on our custom dataset and fine-tuned its parameters to enhance its performance, the subsequent phase involved comprehending the outcomes produced by the model. For every examined image, the model delivers numerous results, with the first ones being the labels assigned to each section. This is crucial for linking the section's name with the coordinates of the bounding boxes that will eventually be showcased.

The bounding box coordinates generated by the model during image inference play a vital role in determining the location of the regions of interest. Utilizing these coordinates makes it feasible to isolate specific subsections of the images that only contain the desired information. In our scenario, we aim to extract sections of the image that exclusively contain personal data presented on the front face of the Portuguese identification card.

To achieve this objective, we designed a data pipeline that involves conducting an image inference and subsequently utilizing the model's outcomes to crop the original image, leading to several sub-images, each corresponding to a particular section. Our workflow deliberately involves minimal image manipulation techniques since the model is adept at processing horizontal segmented images and can accommodate color variations owing to the pre-applied image augmentation techniques. This represents one of the key benefits of our implementation over the current approach, as we can simply resize the image and feed it into the model, reducing the reliance on extensive image

processing that can lead to latency issues. As a result, our ML-powered solution operates independently of any time-consuming image processing, enhancing its overall efficiency.

In the following paragraphs, we will discuss each function that constitutes our pipeline. Our pipeline was entirely coded in Python and created entirely by the Machine Learning team, which modified the existing functions to accommodate the changes introduced by the object detection algorithm.

The initial function in our pipeline is called "**detect_sectionsML**" and only needs a parameter that refers to the image that will be analyzed. For the function to operate correctly, it must receive an image of a Portuguese identification card, which will be fed into our model to generate a prediction. Although the model generates several pieces of information, the function is designed to solely return the values corresponding to the bounding box coordinates and their respective classes/labels.

```python
def detect_sectionsML(image):
    """
    Gets the prediction from the Id Elements model and returns the predicted
    classes and correspopnding bounding boxes.
    """
    try:
        image = np.array(image)
        prediction = id_elements_pt.predict(image)
        print("Prediction: " + str(prediction), flush=True)
        box_coords = prediction["detection_boxes"]
        pred_classes = prediction["detection_classes"]
    except Exception as e:
        print("Error in detect_sectionsML: " + str(e), flush=True)

    return pred_classes, box_coords
```

Figure 29 – Python function used to make model inferences

With a clear understanding of the preceding function's outcomes, we will now proceed to crop the original image utilizing the "**crop_sections_ML**" function, resulting in a list of images, with each element corresponding to a particular section.

```python
def crop_sections_ML(img, pred_classes, box_coordinates, show=False):
    """
    Gets the cropped images for each bounding box predicted, and returns
    them in a list.
    """

    # Create an empty list to store the crops
    section_img_class = {}
    # Iterate over the boxes and crop the image
    im_width = img.shape[1]
    im_height = img.shape[0]
    for cls,box in zip(pred_classes,box_coordinates):
        ymin, xmin, ymax, xmax = box
        [left, right, top, bottom] = [int(xmin * im_width), int(xmax * im_width), int(ymin * im_height), int(ymax * im_height)]
        crop = img[top:bottom, left:right]
        section_img_class[cls] = crop
        if show:
            crop = Image.fromarray(crop)
            crop.show()

    return section_img_class
```

Figure 30 – Python function used to crop card sections by given coordinates

Concluding our pipeline is the "**extract_sections_data_ML**" function, which plays a fundamental role in processing the images of each section. This function employs the Pytesseract library to convert images into text. However, as each section can display data in various formats, such as numbers and punctuation marks (as in the case of height) or solely letters (as in fields like first or last name), each

section requires a distinct Tesseract configuration. As a result, this function merges the multiple configurations with their respective sections and creates a Python dictionary to save the results.

```python
def extract_sections_data_ML(section_img_class, resize = False, resize_factor = 0.5):
    """
    Takes as input a list of cropped images, and returns the OCR output
    for each.
    """

    config_surname = "--psm 6 --oem 1 -l por -c tessedit_char_blacklist=abcdefghijklmnopqrstuvwxyz1
    config_name = "--psm 6 --oem 1 -l por -c tessedit_char_blacklist=abcdefghijklmnopqrstuvwxyz1234
    config_gender = "--psm 10 --oem 1 -c tessedit_char_whitelist=FMX -c tessedit_char_blacklist=123
    config_height = "--psm 6 --oem 1 -c tessedit_char_whitelist=X0123456789\\,"
    config_nation = "--psm 6 --oem 1 -l por -c tessedit_char_blacklist=abcdefghijklmnopqrstuvwxyz12
    config_date = "--psm 6 --oem 1 -c tessedit_char_whitelist=0123456789\\ "
    config_civilid = "--psm 6 --oem 1 -c tessedit_char_whitelist=0123456789"
    config_civilid_digits = "--psm 6 --oem 1 -c tessedit_char_blacklist=abcdefghijklmnopqrstuvwxyz\
    config_expiration = "--psm 6 --oem 1 -c tessedit_char_whitelist=0123456789\\ "

    config_list = [config_surname,config_name,config_gender,config_height,config_nation,config_date
    section_names = ['surnames','names','gender','height','nationality',
                     'birth_date','id_number','doc_number',
                     'expiry_date']
    ocr_results_dict = {}
    for cls,row_img in section_img_class.items():
        if resize:
            row_img = row_img.resize((int(row_img.size[0] * resize_factor),
                                      int(row_img.size[1] * resize_factor)),
                                     Image.Resampling.LANCZOS)

        row_img_cv = cv.cvtColor(row_img, cv.COLOR_RGB2GRAY)
        row_img_cv = cv.threshold(row_img_cv, 0, 255, cv.THRESH_BINARY | cv.THRESH_OTSU)[1]
        ocr_output = pytesseract.image_to_string(row_img_cv, config=config_list[cls-1])
        # The -1 is because the classes are from 1-9 instead of 0-8
        ocr_results_dict[section_names[cls-1]]=ocr_output.replace("\n","")

    return ocr_results_dict
```

Figure 31 – Python function used to apply OCR in given sections

# 4. EXPERIMENTAL STUDY

In this chapter, we will present the outcomes of our experimentation with various model architectures and determine the optimal one for deployment in a production setting. We will start by outlining the evaluation metrics used, and their relationship to the concepts introduced in chapter 2.5. Following that, we will end by providing a detailed analysis of the comparative performance of our best model against the existing production strategy.

## 4.1. EVALUATION PROTOCOL

Throughout our object detection model training, TensorFlow generates real-time training process checkpoint log files which allow us to assess the performance of our models. These logs contain COCO evaluation metrics, which include essential metrics like mean average precision, recall, and Intersection over Union. These metrics aid us in quantifying the accuracy and precision of our models and enable us to identify areas that require improvement. By using COCO evaluation metrics, we can ensure that our models are optimized to detect and classify objects with high accuracy and efficiency, allowing us to achieve our research goals.

As we have previously mentioned, we split our dataset into two parts - a training set and a test set. The test set plays a crucial role in evaluating the performance of our model, as it contains data that the model has never seen before. Important to understand that the evaluation process utilizes the checkpoint files generated during the training process to assess the model's ability to detect objects in the test dataset. The evaluation generates a set of metrics that provide a summary of the model's performance, enabling us to track its accuracy and precision over time. These metrics offer valuable insights into the strengths and weaknesses of our model, allowing us to fine-tune it for optimal performance. By regularly monitoring the evaluation metrics, we can ensure that our models are continuously improving and delivering reliable results (L. Vladimirov, 2020).

All these results can also be visualized with the help of TensorBoard. This visualization tool converts the evaluation results obtained from the checkpoint files into intuitive and informative dashboards. These dashboards provide a comprehensive overview of the model's performance, enabling us to identify areas that require improvement. In addition to the evaluation metrics, TensorBoard also allows us to visualize the detection results in the test images, providing a clear understanding of how well the bounding boxes are detecting the areas of interest. This feature is particularly useful in identifying false positives or false negatives, which can be further investigated and corrected to improve the model's accuracy. By utilizing TensorBoard to analyze the evaluation results and visualize the detection outputs, we can gain a deeper understanding of our model's performance and make informed decisions on how to optimize it further.

Once we establish that a model produces favorable outcomes, it becomes crucial to assess the latency it introduces while making an inference. This aspect holds immense significance since it can adversely impact the pipeline's performance if the response time increases significantly. Therefore, it is essential to carefully scrutinize the model's inference time to ensure that it meets the requirements of the intended application.

Finally, upon analyzing the model's performance metrics over the test set, checking the bounding boxes display and the latency, the final evaluation phase now shifts focus to assessing the outputs

generated by the OCR engine. In light of this, the project team decided to build a web application using Streamlit to compare the results of our approach against the current production environment's output. The application provides an interface that you can check below, that allows us to browse through a folder of images depicting identification cards and manually check which sections Tesseract accurately converted to text using tick boxes. This evaluation phase is undoubtedly the most time-consuming since there is no record of all the ground truth values of the documents, and the performance of the OCR engine must be assessed by manually inspecting each section to determine if the output matches the expected values.

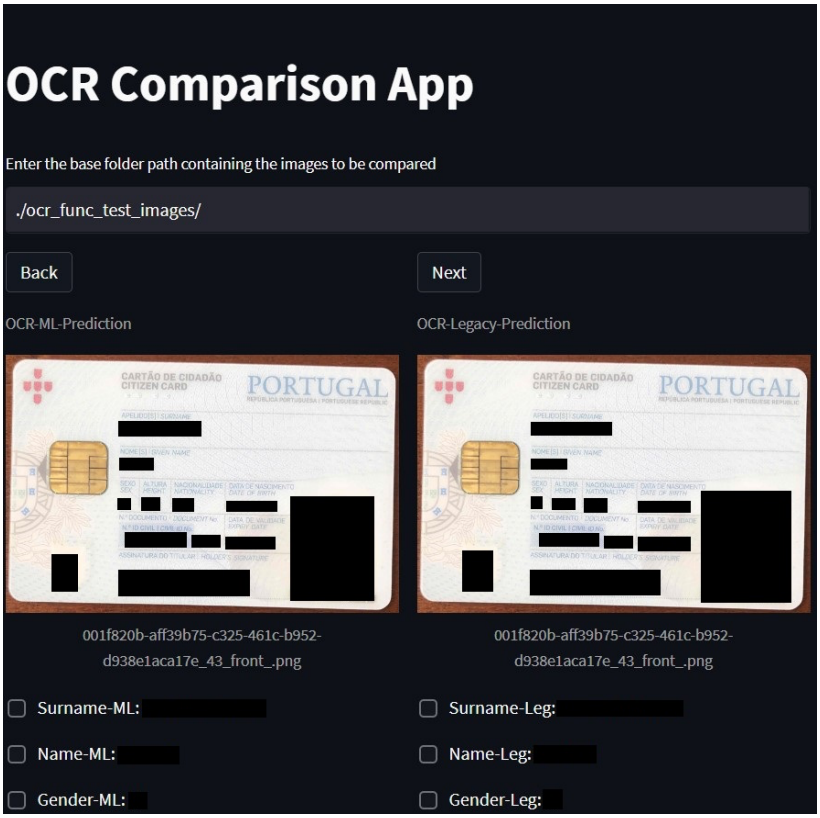


Figure 32 - Streamlit Output Comparison App

Upon evaluating both methods' results on our 3000-image dataset, the application generates a CSV file, which we use to produce a line plot that will be used to compare visually the two implementations. We will present this plot below when analyzing the results.

The evaluation process for this project is comprehensively outlined, highlighting each distinct phase. The next chapter will provide a detailed overview of the outcomes obtained in each of these phases and explicate the rationale behind selecting the final model.

## 4.2. Experimental Results and Discussion

This chapter focuses on presenting the outcomes achieved with our artificial intelligence algorithm's implementation for detecting Portuguese ID card sections. We have divided the results into three phases, as discussed in the previous chapter, to assess our approach's effectiveness in different ways.

### 4.2.1. TensorBoard Results

Throughout each phase presented in Table 2, we started checking the performance of each model via TensorBoard by analyzing its evaluation metrics on the train and test set. This approach allowed us to monitor the training progress of a specific model in real time and compare it to previously trained models.

Our initial emphasis is on the evaluation metric known as loss, which is a value assigned during the training of a model that aims to reflect the disparity between the model's predicted output and the real output. The fundamental objective of the training process is to decrease the loss metric, indicating that the predicted output should closely resemble the actual output.

Presented in the following table are the loss values of the most optimized versions for each of the trained models. It is evident that all the loss values are quite low, but the CenterNet Resnet101 model outperformed the rest, indicating that its predictions are the most accurate. This initial evaluation positions it as a strong contender for the final model, although we still need to analyze numerous other metrics to determine if it is the optimal choice.

| Model | Loss value |
|---|---|
| CenterNet HourGlass104 | 0.01957 |
| EfficientDet1 | 0.02583 |
| EfficientDet2 | 0.02957 |
| Faster R-CNN Inception | 0.01372 |
| SSD ResNet101 | 0.04287 |
| SSD ResNet50 | 0.0577 |
| CenterNet Resnet50 | 0.01219 |
| CenterNet Resnet101 | **0.0106** |

Table 3 - Loss Comparison

Regarding the metrics of mean average precision and recall values, the results were found to be dissimilar. Among the evaluated model architectures, only three achieved values higher than 80% for mean average precision, namely CenterNet HourGlass104, Faster R-CNN Inception, and CenterNet Resnet50. These models also performed exceptionally well in terms of recall values. Thus, these same three models were considered the most effective ones for the given task. A comparative view of the recorded values for both these evaluation metrics is presented in the following table.

| Model | mAP value | Recall value |
|---|---|---|
| CenterNet HourGlass104 | **0.8175** | **0.8563** |
| EfficientDet1 | 0.7683 | 0.8093 |
| EfficientDet2 | 0.7637 | 0.8072 |
| Faster R-CNN Inception | 0.8159 | 0.8538 |
| SSD ResNet101 | 0.7844 | 0.8254 |
| SSD ResNet50 | 0.7591 | 0.7971 |
| CenterNet Resnet50 | 0.8109 | 0.8480 |
| CenterNet Resnet101 | 0.7988 | 0.8347 |

Table 4 - Mean Average Precision/Recall Results per model architecture.

After analyzing the results obtained, it was evident that four models had performed remarkably well on the test set. Despite being able to comprehend the functioning of the models visually through the TensorBoard, an additional group of images was created. These images were never before analyzed by the models, and visualizations with bounding boxes were generated for them. By doing so, an extra evaluation layer was established to gauge the robustness of the model's results. This helped in obtaining a more comprehensive understanding of the effectiveness of the models and their capacity to generalize to unseen data.

The test results revealed that the CenterNet Resnet50 model had difficulty in detecting certain sections, even in favorable conditions, whereas the other models performed better in this aspect. Taking this into account, the Machine Learning team decided to focus solely on the other three models that consistently and similarly detected all sections of all images. Hence, the size and latency tests were continued only for these models to ensure that they met the desired performance criteria. This approach helped in streamlining the evaluation process and optimizing the selection of the best-performing model for the project.

### 4.2.2. Size & Latency Tests

The second testing phase was initiated to identify the architecture that could introduce the least amount of latency to the OCR pipeline. Latency, in this context, refers to the time taken by the model to process a single inference, i.e., the time between the model receiving an image and returning the results. This test assumes great significance as it is not sufficient for a model to have exemplary performance alone if it consumes an extended period for processing. If such an instance occurs within the pipeline that is currently in production, it could significantly slow down the entire OCR solution and worsen the user experience.

The two bar plots below provide an overview of the results obtained in the latency testing phase. The first one was made by measuring the processing time taken by the models on the images present in our dataset. To achieve this, we developed a python script that iterated through the entire dataset and calculated the average time taken by each of the models to return results compared to the

current OCR strategy (OCR Legacy), in seconds. The second one provides us with the cumulative latency, which offers an understanding of the total time it would take to perform 500 inferences. This information assumes significance while evaluating the effectiveness of implementing this strategy in production. This helped us to compare and contrast the performance of the different models in terms of their latency and select the most efficient one for integration into the OCR pipeline.

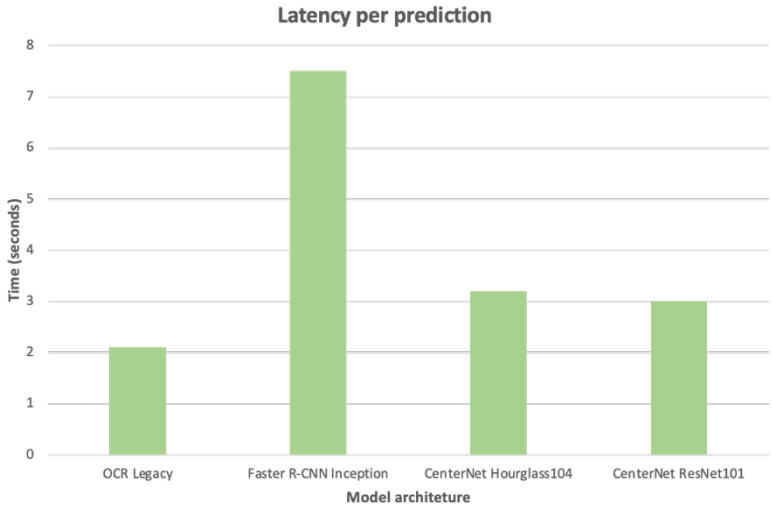


Figure 33 - Latency per prediction

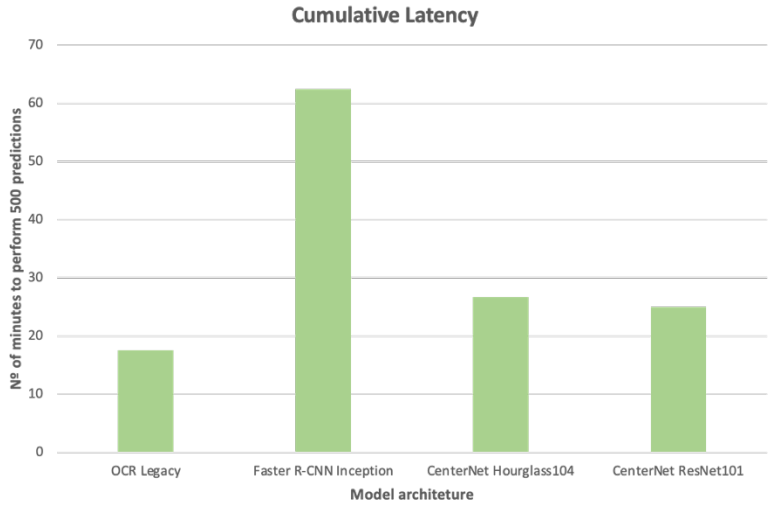


Figure 34 - Cumulative Latency

Despite presenting impressive results the Faster R-CNN Inception model, as you can check above, comes with a major drawback - its unacceptable latency time. In comparison to the current OCR implementation, the model takes approximately 3.5x longer to execute a substantial number of inferences. Consequently, our focus shifts to evaluating the remaining models to determine the most suitable replacement candidate for the OCR Legacy. Fortunately, these alternative models offer processing times compared to the current implementation, and although their integration with ML technology may marginally increase latency time, the benefits they provide far outweigh the slight increase in processing time.

We initiated efforts to differentiate between the remaining models based on their respective storage sizes. Typically, models that occupy more storage space tend to demand more computational resources and memory to function optimally. This can pose challenges for resource-constrained systems, rendering such models unsuitable for online or real-time applications where low latency is of paramount importance.

Once we became aware of the vast difference in the storage space occupied by the two models, the decision became clear. We would opt for the CenterNet Resnet101 model as a viable replacement for the OCR Legacy. However, to substantiate this decision, we needed to ascertain whether this alternative could deliver better outcomes than the previous model. In the next section, we'll present the results that we aim to obtain using a Streamlit app that compares the OCR powered by ML and the OCR Legacy.

### 4.2.3. OCR Results Comparison

In the final evaluation phase, we aim to compare the performance of the pipeline presented in chapter 3.2.4 with the current results produced by Biometrid daily. To achieve this, we use a Python framework called Streamlit to create a CSV file containing the success rate of each card section that was used to generate the plot that you can observe below.

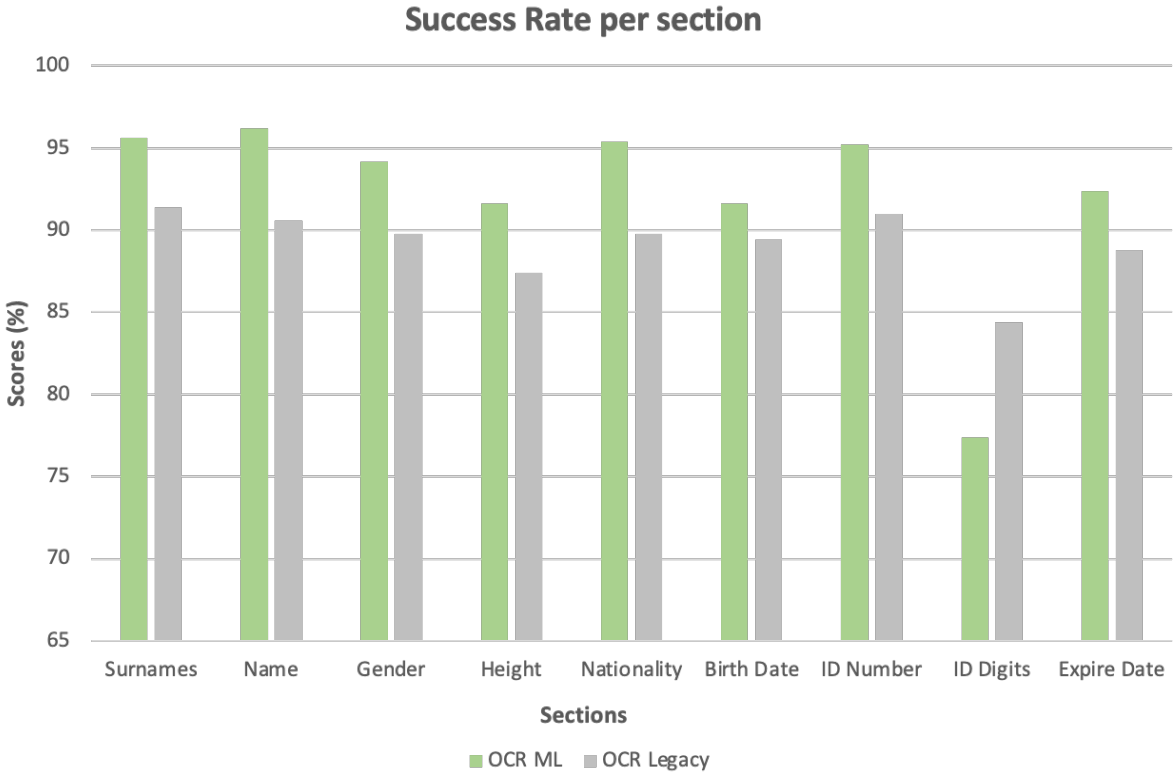


Figure 35 - Success rate per section for each OCR Strategy

Based on the information presented, it seems reasonable to conclude that the new OCR pipeline powered by machine learning performed better than the current implementation in most sections, except for the identification digits section where the current implementation returned better results.

Even though, as we stated earlier, there was an increase in latency by the new implementation, we were able to improve the average success rate by approximately 3%, indicating that the new model is set ahead over the current implementation.

After these results, we conclude that the new OCR model shows promise for improving accuracy in most sections of the OCR process, though further testing and optimization may be necessary to address the identified issues regarding the identification digits.

# 5. CONCLUSIONS

In this internship report, the challenge was to train and deploy an object detection model that was able to automate the process of retrieving all the front sections from a Portuguese identification card using artificial neural networks to support an OCR-based solution for information retrieval of identification cards. This new implementation had as a baseline the current production environment, which only uses image manipulation techniques to detect card sections, an approach that has some limitations regarding inferences on low-quality images.

The applied research in this report was based on a case study using a custom anonymized real-world dataset composed of images provided by AMA.

The theoretical framework section of this report aimed to provide a comprehensive understanding of object detection algorithms, from basic concepts to the model architectures experimented with over the past few months. We started by providing a comprehensive explanation of Machine Learning and how to approach a project utilizing it. Moving forward, we delved into the realm of Deep Learning and outlined the process of evaluating models to determine whether they perform as expected.

Within the methodology chapter of this report, we presented the various tools that were utilized throughout the project. Additionally, we provided a detailed, step-by-step explanation of how we trained an artificial neural network (ANN) utilizing the TensorFlow Object Detection API. Finally, we explained the image pipeline construction process, which is capable of receiving model-generated results and extracting information from them via an Optical Character Recognition mechanism, ultimately allowing for the conversion of images to text.

As we conclude this report, we presented the various evaluation phases that were implemented in the selection of the optimal model. Through this process, we ultimately compare our strategy with the currently implemented approach and publish the final results.

## 5.1. LIMITATIONS

The major limitations that emerged during the course of this project were mainly regarding the data and insufficient work equipment. This includes the limited availability of images for developing our custom dataset, the unavailability of cloud services for expediting machine learning model training, and the confidentiality of personal information that impeded result sharing.

The primary constraint of this project was the shortage of images available for building the dataset necessary for training machine learning models. Despite strenuous efforts to collect relevant images, the dataset employed in this study remained comparatively small compared to the datasets commonly employed in similar investigations. This limitation was considered, and the team use techniques like data augmentation to prevent negative impacts on the model's accuracy, generalizability, and an increased likelihood of overfitting.

Another limitation experienced during this project was the lack of cloud services for training the model and storing all the experiments that were made. We were aware that because of that the training process was slower and required more computational resources than would have been

necessary with cloud services. This limitation resulted in extended wait times and reduced the number of experiments conducted.

Finally, the need to restrict the sharing of results due to the personal information contained within the dataset constituted a significant limitation of this project. Although measures were taken to ensure the privacy and confidentiality of the individuals in the dataset, this limitation prevented the broad dissemination of the results, which could have limited the impact of the research.

## 5.2. Future Work

Moving forward, our focus will be directed toward improving our model by utilizing additional data and enhancing the efficiency of our data pipeline. Following the model's deployment, it will undergo multiple stages of testing to ensure its effectiveness in a production environment. Only through this rigorous testing can we determine the viability of its implementation.

If the implementation of our model proves to be successful, we anticipate the emergence of new projects focused on training models capable of identifying elements from identification cards and passports from various countries. These models would then be integrated into Biometrid's Optical Character Recognition (OCR) pipeline to further enhance its capabilities.

# 6. BIBLIOGRAPHY


[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016, November). Tensorflow: a system for large-scale machine learning. In *Osdi* (Vol. 16, No. 2016, pp. 265-283).

[2] Abraham, A. (2005). Artificial neural networks. *Handbook of measuring system design*.

[3] Alexandari, A. M., Shrikumar, A., & Kundaje, A. (2017). Separable fully connected layers improve deep learning models for genomics. *BioRxiv*, 146431.

[4] Ariwala. (2022, October). 9 Real-World Problems that can be Solved by Machine Learning. Maruti Techlabs. https://marutitech.com/problems-solved-machine-learning/

[5] Attivissimo, F., Giaquinto, N., Scarpetta, M., & Spadavecchia, M. (2019, October). An automatic reader of identity documents. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 3525-3530). IEEE.

[6] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., ... & Ghayvat, H. (2021). CNN variants for computer vision: history, architecture, application, challenges and future scope. *Electronics*, *10*(20), 2470.

[7] Bughin, J., Hazan, E., Ramaswamy, S., Chui, M., Allas, T., Dahlstrom, P., ... & Trench, M. (2017). Artificial intelligence: The next digital frontier?

[8] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569-6578).

[9] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2009). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*, 303-308.

[10] Farooq, O. (2022, October 3). [10 Best Machine Learning Stocks to Invest In]. Yahoo Finance. https://finance.yahoo.com/news/10-best-machine-learning-stocks-165306920.html

[11] Fletcher, J., & Kostiainen, A. (2022, November). Ethical Principles for Web Machine Learning. W3C. https://www.w3.org/TR/webmachinelearning-ethics

[12] Fritz Labs. (2018). Object Detection Guide. Fritz AI. https://www.fritz.ai/object-detection/

[13] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[14] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[15] H. Zhu, X. Yan, H. Tang, Y. Chang, B. Li and X. Yuan, "Moving Object Detection with Deep CNNs," in IEEE Access, vol. 8, pp. 29729-29741, 2020, doi: 10.1109/ACCESS.2020.2972562.

[16] Hakim, H., & Fadhil, A. (2021, February). Survey: Convolution neural networks in object detection. In Journal of Physics: Conference Series (Vol. 1804, No. 1, p. 012095). IOP Publishing.

[17] Horak, K., & Sablatnig, R. (2019, August). Deep learning concepts and datasets for image recognition: overview 2019. In *Eleventh international conference on digital image processing (ICDIP 2019)* (Vol. 11179, pp. 484-491). SPIE.

[18] Huang, J. (2017, August 14). *"Our goal: Every single car will be autonomous."* Bosch Global. https://www.bosch.com/stories/thought-leader-jensen-huang/

[19] Islam, N., Islam, Z., & Noor, N. (2017). A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*.

[20] Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685-695.

[21] Janke, J., Castelli, M., & Popovič, A. (2019). Analysis of the proficiency of fully connected neural networks in the process of classifying digital images. Benchmark of different classification algorithms on high-level image features from convolutional layers. *Expert Systems with Applications*, *135*, 12-38.

[22] Krizhevsky A., Sutskever I. & Hinton G.E., 2012. Imagenet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS). :1097–1105.

[23] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[24] Mariano, V. Y., Min, J., Park, J. H., Kasturi, R., Mihalcik, D., Li, H., ... & Drayer, T. (2002, August). Performance evaluation of object detection algorithms. In *2002 International Conference on Pattern Recognition* (Vol. 3, pp. 965-969). IEEE.

[25] Mavrikis, S. T., Antonopoulos, C. P., Voros, N. S., & Keramidas, G. (2021, September). Comparative evaluation of computer vision technologies, targeting object identification and localization scenarios. In *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)* (pp. 1-8). IEEE.

[26] Moroney, L. (2020). AI and Machine Learning for coders. O'Reilly Media.

[27] Mudiarta, I. M. D. R., Atmaja, I. M. D. S., Suharsana, I. K., Antara, I. W. G. S., Bharaditya, I. W. P., Suandirat, G. A., & Indrawan, G. (2020, April). Balinese character recognition on mobile application based on tesseract open-source OCR engine. In Journal of Physics: Conference Series (Vol. 1516, No. 1, p. 012017). IOP Publishing.

[28] Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.

[29] Ranjan, A., Behera, V. N. J., & Reza, M. (2021). Ocr using computer vision and machine learning. *Machine Learning Algorithms for Industrial Applications*, 83-105.

[30] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[31] Ryan, M., & Hanafiah, N. (2015). An examination of character recognition on ID card using template matching approach. Procedia Computer Science, 59, 520-529.

[32] Simeone, O. (2018). A very brief introduction to machine learning with applications to communication systems. IEEE Transactions on Cognitive Communications and Networking, 4(4), 648-664.

[33] Singh, A., Bacchuwar, K., & Bhasin, A. (2012). A survey of OCR applications. *International Journal of Machine Learning and Computing*, *2*(3), 314.

[34] Vladimirov, L. (2020). TensorFlow 2 Object Detection API Tutorial. TensorFlow 2 Object Detection API Documentation. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html

[35] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

[36] Zulkhaizar, A. (2023, January 4). *The Future of AI: How Artificial Intelligence is Revolutionizing the Way We Work - Digital First Magazine*. Digital First Magazine. https://www.digitalfirstmagazine.com/the-future-of-ai-how-artificial-intelligence-is-revolutionizing-the-way-we-work/