1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Fábio André Constante Ferreira

INTELLIGENT SYSTEM FOR AUTOMATIC VALIDATION OF IMAGES

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. João Nuno Correia and Miguel Antunes presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Agradecimentos

Antes de mais, queria agredecer aos meus orientadores João Nuno Correia e Miguel Antunes por me terem auxiliado na aprendizagem em relação ao mundo da visão computacional e que me permitiu realizar o trabalho descrito nesta mesma tese.

Queria também agreceder o apoio constante ao longo da minha vida por parte dos meus pais, Paula e João Ferreira pois sem este apoio e a educação dada por eles, nunca teria sido capaz de chegar até aqui. O meu irmão, Diogo Ferreira, sem dúvida o meu melhor amigo e aquele que sempre me fez rir sempre que estavamos juntos independentemente das dificuldades. Obrigado puto.

Finalmente queria agradecer a todas as pessoas que passaram pela minha vida e que de múltiplas formas me ajudaram a crescer e a tornar na pessoa que sou hoje, todas as experiências pelas quais passei me ajudaram neste percurso.

Fábio André Constante Ferreira

Abstract

Brands&Ninjas is a platform based on Crowdsourcing that allows brands to monitor campaigns, evaluate the stock, monitor prices, and analyse the space on the shelves of each of its products at multiple points of sale. To obtain this information, the platform resorts to users that use the application and are paid to do missions, referred to as ninjas. In these missions, they are requested to verify some products' location, answer questions considering these products, take pictures that verify the considered products' locations and their answers to the asked questions. The data obtained from these missions is then validated. At the moment, this is done manually, where only a portion of 10 to 20%, randomly selected, is analysed to verify if a mistake occurred. From the validated images, 15 to 20% corresponded to errors indicating that the answers provided were wrong. The errors made by the ninjas, if not found, result in the creation of erroneous reports to be given to the brands who requested the missions, which indicates how critical it is the validation process for the Brands&Ninjas platform. Considering the high number of missions that are executed, the manual validation of the received images has become impracticable at this point, which resulted in a necessity to create a system capable of automatically validating the received images.

The ninjas who use the Brands&Ninjas platform can perform several different tasks, from which many of these tasks can be solved using Computer Vision (CV). So, a specific task was selected to serve as a starting point to show that it is possible to use CV to validate many of the tasks performed by the ninjas using the received images. In particular, a system was developed to validate one of the ninjas' most common and performed tasks, detecting a specific product on a shelf. Moreover, the object selected to validate the considered task was the Delta Q Qalidus package of 10 capsules, Qalidus class.

The first step corresponded to the realisation of a survey of existing competitors and solutions and a state of the art analysis of CV tasks related to the task at hand. This survey resulted in a selection of four different object detection models. Two simpler and older models that use Haar and Histogram of Oriented Gradients (HOG) features, respectively. The other two models selected were the You Only Look Once (YOLO)v4 model and an EfficientDet object detector, the current state of the art one-stage and two-stage object detectors, respectively. Therefore, an experimentation with object detectors was made to select the best object detection model. The training dataset was composed of three distinct missions and the testing dataset of one complete mission. On the training dataset, some data augmentation techniques were used focused on the lighting conditions of the supermarkets and the way a picture can be taken. Considering this, two different YOLOv4 models were trained, one with the initial training dataset and the online data augmentation implemented by the YOLOv4 model and another with the YOLOv4 data augmentation turned off, and the training dataset with the offline data augmentation implementation, both trained for 6000 iterations. As expected, the best object detector of the two YOLOv4 models trained was the YOLOv4 model trained with the offline data augmentation implementation. Having achieved an Average Precision (AP) of 75.19%, almost five per cent more than the one achieved by the YOLOv4 model with data augmentation that achieved 70.31% AP. From all the trained models, the best model selected for the implementation of the system was the YOLOv4 model trained with offline data augmentation for 8000 iterations, having achieved an AP of 92.60%.

With the best model selected, a system that uses an API endpoint to be accessed was created. Then, three complete missions were used to validate its functioning. As a result, the system correctly found 95.06% of all the objects of the class Qalidus in all the missions images. Furthermore, from the images that were validated, 99.35% were correctly validated. With these results, it was possible to validate that the construction of the objective system was successful and that the system created can, with a 95% confidence, identify the presence of objects of the class Qalidus correctly. Although the building of the system was considered successful, there are still steps that need to be done in the future to be fully considered an essential part of the Brands&Ninjas platform. Being one of these steps, the systems ability to identify more than only objects of the Qalidus class.

Keywords

Computer vision, Object Detection, Deep Learning, Automatic Validation

Resumo

A Brands&Ninjas é uma plataforma baseada em Crowdsourcing que permite às marcas monitorizar campanhas, avaliar stocks, vigiar e controlar preços e ainda analisar o espaço nas prateleiras dos seus produtos em múltiplos pontos de venda. Para obter estas informações, a plataforma recorre a utilizadores, que são pagos para realizar missões, sendo estes utilizadores denominados por ninjas. Nas missões, é pedido aos ninjas que verifiquem a localização de alguns produtos, respondam a perguntas sobre esses produtos e tirem fotos que comprovem a localização dos produtos considerados e as respostas às perguntas feitas. Os dados obtidos nestas missões são então validados. Atualmente esta validação é feita manualmente, onde apenas uma porção de 10 a 20%, selecionada aleatoriamente, é analisada para verificar se ocorreu um erro. Das imagens validadas, 15 a 20% correspondem a erros que foram identificados. Os erros cometidos pelos ninjas, se não forem encontrados, resultam na criação de relatórios errados a serem entregues às marcas que solicitaram as missões, o que indica o quão crítico é o processo de validação para a plataforma Brands&Ninjas. Considerando o elevado número de missões executadas, a validação manual das imagens recebidas tornou-se inviável neste momento, o que resultou na necessidade de criação de um sistema capaz de validar automaticamente as imagens recebidas.

Os ninjas que usam a plataforma Brands&Ninjas podem realizar várias tarefas diferentes, sendo que muitas dessas tarefas podem ser resolvidas usando visão computacional. Considerando estas múltiplas tarefas possíveis, uma única tarefa foi selecionada para servir de ponto de partida para mostrar que é possível usar visão computacional para validar muitas das tarefas realizadas pelos ninjas a partir das imagens recebidas. Em particular, foi desenvolvido sistema para validar uma das tarefas mais comuns realizadas pelos ninjas, a verificação da presença de um produto específico numa prateleira. Sendo neste caso usada a embalagem Delta Q Qalidus de 10 cápsulas para validar esta mesma tarefa escolhida, tendo sido esta embalagem associada à classe Qalidus.

A primeira etapa correspondeu à realização de uma pesquisa de soluções existentes aplicadas por concorrentes da Brands&Ninjas e a uma análise do estado da arte das tarefas de visão computacional relacionadas com a tarefa em questão. Esta pesquisa resultou na seleção de quatro modelos diferentes de deteção de objetos. Dois modelos mais simples e antigos que usam Haar e HOG como features, respetivamente. Os outros dois modelos selecionados foram: o modelo YOLOv4 e um detetor de objetos denominado EfficientDet, correspondendo estes modelos ao atual estado da arte em detetores de objetos de one-stage e two-stages, respetivamente. Portanto, foi realizada uma experimentação com estes mesmos modelos, com o objetivo de selecionar o melhor modelo de deteção de objetos. O dataset de treino usado foi composto por três missões distintas e o dataset de teste por uma missão completa. No dataset de treino, foram aplicadas algumas técnicas de data augmentation com o objetivo de melhorar este mesmo dataset, considerando as condições de iluminação dos supermercados e a forma como uma foto pode ser tirada. Foram treinados dois modelos YOLOv4 diferentes, um com o dataset de treino inicial e usando a online data augmentation implementada pelo modelo YOLOv4 e outro com esta data augmentation desligada, usando o dataset de treino com a offline data augmentation implementada, tendo sido ambos treinados 6000 iterações. Como esperado, o melhor detetor de objetos dos dois modelos YOLOv4 treinados foi o modelo YOLOv4 treinado com offline data augmentation. Tendo este modelo alcançado uma Average Precision(AP) de 75,19%, quase cinco por cento a mais que a alcançada pelo modelo YOLOv4 que usa online data augmentation que atingiu apenas 70,31% AP. De todos os modelos treinados, o melhor modelo selecionado para a implementação do sistema foi o modelo YOLOv4 treinado durante 8.000 iterações usando offline data augmentation, tendo alcançado um AP de 92,60%.

Com o melhor modelo selecionado, foi criado um sistema que usa um endpoint de uma API para ser acedido. Tendo sido usadas três missões completas para validar o funcionamento do sistema. O sistema foi capaz de encontrar corretamente 95,06% de todos os objetos da classe Qalidus presentes nas imagens fornecidas. Para além disso, das imagens validadas, 99,35% foram validadas corretamente. Com estes resultados, foi possível concluir que a construção do sistema foi bem-sucedida e que o sistema criado pode, com 95% de certeza, identificar corretamente a presença de objetos da classe Qalidus. Embora a construção do sistema tenha sido considerada um sucesso, ainda existem etapas que necessitam de ser realizadas no futuro para que ele possa ser considerado uma parte essencial da plataforma Brands&Ninjas. Sendo uma dessas etapas, a capacidade do sistema para identificar mais do que um objeto.

Palavras-Chave

Visão computacional, Deteção de objectos, Aprendizagem profunda, Validação automática

Contents

Acronyms

AI Artificial Intelligence. 7, 14, 22

AP Average Precision. v, xvi, xvii, 10, 11, 46–49, 60

API Application Program Interface. 50, 51

BiFPN bi-directional feature network. 20

BoF Bag of freebies. 19, 20, 31, 42, 44

BoS Bag of specials. 19, 20, 31

CBIR content-based image retrieval. 16

CNN Convolutional neural networks. 15–18

COCO common objects in context. 11, 42

CPU Central Process Unit. 21

CV Computer Vision. v, xvi, 3, 4, 7–9, 12–14, 16, 21, 31, 59

DNN Deep Neural Networks. 16

DPM Deformable Part-based Model. 15, 16

FLOPs FLoating-point Operations Per Second. 20, 31

FPN Feature Pyramid Networks. xvi, 17–20

FPR False Positive Rate. 9

GPU Graphics Processing Unit. 19

HOG Histogram of Oriented Gradients. v, xvi, 15, 21, 31, 40, 41, 47, 48, 60

IoU Intersection over Union. xvi, 9–11, 46

JSON JavaScript Object Notation. 42

mAP mean Average Precision. 11, 46

NLP natural language processing. 16

NMS Non-maxima suppression. 18

NPC Negative Predictive Value. 9

RCNN Regions with CNN features. 16–19

ROC Receiver Operating Characteristic. 9

RoI	Region of Interest. 16, 17
SAM	Spatial Attention Module. 20
SAT	Self-Adversarial Training. 20
SE	Squeeze-and-Excitation. 20
SIFT	Scale Invariant Feature Transform. 16
SPP	Spatial Pyramid Pooling. 20
SSD	Single Shot MultiBox Detector. 18
SVM	Support vector machine. xvi, 17, 40, 41, 47, 48
TNR	True Negative Rate. 9
VOC	visual object classes. 11, 46
YOLO	You Only Look Once. v, xvi, xix, 18–21, 31, 34, 35, 38, 41–51, 54, 60

List of Figures

List of Tables

Chapter 1

Introduction

Brands&Ninjas is a platform created by RedLight Software, which uses data as a service based on Crowdsourcing that allows brands to monitor campaigns and promotions, evaluate the stock, monitor prices and analyse the space on the shelves of each of its products at the points of sale. To obtain the information requested by the brands, Brands&Ninjas uses ninjas, who are regular people who register in the application and are paid to do missions where they are requested to verify the location on stores of selected products, answer questions considering these products, take pictures that verify the products' location and validate the answers given to the asked questions. The obtained information provided by the ninjas is then processed by the Brands&Ninjas platform, analysing the ninjas' answers, and then the results are shown to the brands using a web app. Given that humans are prone to making errors, there is a need to validate if the given answers are correct using the images taken by the ninjas in each realised mission.

The validation process is critical since the ninjas may give wrong answers to the asked questions. Being These answers used to answer the brands' questions and result in an erroneous report. Therefore the problem here is not just that the answers associated with that particular image are wrong, but the resulting report will consider errors that may lead the brand to make wrong decisions. Unfortunately, despite the importance of this validation, it is currently only being done by manually to approximately 10 to 20% of all the images in each mission, randomly selected, where each mission has on average 100 or more images. In addition, many missions occur simultaneously, which makes the full manual validation impracticable at this point.

Next are showcased some example questions asked to a ninja in a possible mission requested by Delta (Delta Cafés is a company specialising in roasting and selling coffee) and its response to each question of this mission. In figure 1.2 are two versions of the image taken and sent by the ninja to validate his answers. Figure 1.2a corresponds to the original version of the image taken by the ninja to validate the images he provided, and Figure 1.2b corresponds to the same image annotated regarding the presence of the objects asked in the following questions.

- Is the Delta Q Epic package of 10 capsules(Figure 1.1a) on the shelf?

  Regarding the presence of the Delta Q Epic package of 10 capsules(Figure 1.1a) on the shelf, the ninja answer was that the Delta Q Epic package of 10 capsules was on the shelf. As shown in Figure 1.2b(in this figure are the annotated objects found to answer these questions), the Delta Q Epic package of 10 capsules is indeed on the shelf, and the brown bounding box represents it. Thus, this answer corresponds to a true positive case where the ninja's answer corresponds to the correct answer.

- Is the Delta Q Mythic package of 10 capsules(Figure 1.1b) on the shelf?

  The ninja answer regarding the presence of the Delta Q Mythic package of 10 capsules(Figure 1.1b) was just like the answer for the Delta Q Epic package of 10 capsules(Figure 1.1a) that it was on the shelf. However, after analysing Figure 1.2a, it is possible to verify that the Delta Q Mythic package of 10 capsules is not on the shelf, something proven by the resulting annotated image shown in Figure 1.2b that has no Delta Q Mythic package of 10 capsules.

- Is the Delta Q Qalidus package of 10 capsules(Figure 1.1c) on the shelf?

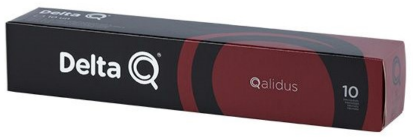
  The ninja answer relatively to the presence of the Delta Q Qalidus package of 10 capsules(Figure 1.1c) was that it was not on the shelf. However, after a careful analysis of Figure 1.2b, it is possible to see that the Delta Q Qalidus package of 10 capsules is on the shelf, and the red bounding box represents it, contrarily to the stated by the ninja, this error corresponds to a false negative case.

- Is the Delta Q Bio package of 10 capsules(Figure 1.1d) on the shelf?

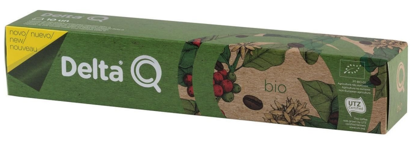
  For this last question, the ninja answer regarding the presence of the Delta Q Bio package of 10 capsules(Figure 1.1d) was that it was not on the shelf. The correct answer for this particular question is that the Delta Q Bio package of 10 capsules is not on the shelf, as shown in figure 1.2a. Therefore, this case corresponds to a true negative case.

(a) Delta Q Epic package of 10 capsules

(b) Delta Q Mythic package of 10 capsules



(c) Delta Q Qalidus package of 10 capsules



(d) Delta Q Bio package of 10 capsules

Figure 1.1: Images of the products that were asked to the ninja to verify their presence on the supermarket shelves.

The answers given by the ninja to the questions made in the Delta mission relatively to the products in Figure 1.1 show the necessity to validate the data inserted by the ninjas, to solve the occurrence of both errors of type one(false positives), the error made by the ninja while answering the second question, and type two errors(false negatives), the error made by the ninja while answering the third question. The validation process is currently being done by hand, only on 10 to 20% of the received images, with an error percentage in these validated images between 15 to 20%. The results obtained with the validation done by hand show the necessity of creating a system for automatic validation of all the images taken by the ninjas to find and solve errors of both types one and two.

Considering the need for automatic validation of the images provided by the ninjas, the best option is to use a CV task since it is needed that the computer analyses the provided images. There are several CV tasks involved in the process of creating a system for automatic validation of the images taken by the ninjas. The first solution corresponds to the creation of a system capable of detecting objects on the shelves, validating the presence of these objects and comparing it with the answers given by the ninjas to validate the questions made to them automatically. This system needs to detect objects in multiple different positions and multiple sizes under multiple lighting conditions. Moreover, this system for automatic validation of all the images will need to be able to receive the images to be analysed remotely. For each one, identify the objects present in it, and how many of each it has identified, so that with these answers compared with answers given by the ninjas, it can validate them and help correct the errors that they might have done that can be solved with the provided images. Considering that this system has the objective of being used to complement the existing Brands&Ninjas platform, it needs to be accessed by the Brands&Ninjas platform integrating it and providing the platform with a way of validating the received images.

(a) Image taken by the ninja to validate the answers given by him.

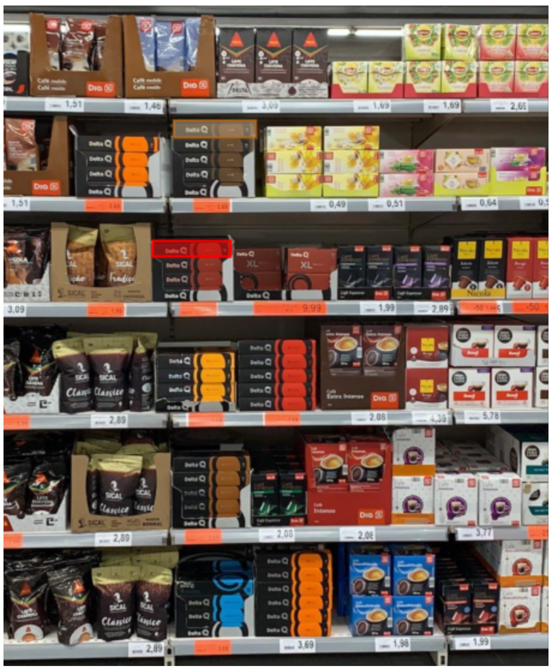

(b) Annotated image taken by the ninja to validate his answers, the bounding box in red is a Delta Q Qalidus package of 10 capsules, and the bounding box in brown is a Delta Q Epic package of 10 capsules.

Figure 1.2: Two versions of the image taken by the ninja that was used to validate the answers to the questions asked in the mission, Figure 1.2a corresponds to the image taken as it was sent, and the Figure 1.2b corresponds to the same image annotated with the objects found from the requested mission.

1.1 Objectives

So far, it has been described the necessity that led to the need of creating a system for automatic validation of all the images. This section explicitly describes the task that the objective model attempts to solve, the reason that led to its choice among all the tasks performed by the ninjas, and the objectives that need to be fulfilled for the system to help validate this task.

The ninjas that use the Brands&Ninjas platform need to perform several different tasks, and many of these tasks can be solved using CV. For this particular first system, the specific task that will be attempted to be solved is one of the tasks most performed by the ninjas, which is to verify the presence of a specific product on a shelf, utilising the images received from the ninjas to verify their answers. By analysing the presence of a product on an image, it is possible to automatically compare the objects found with the ninjas' answers, validating them. Focusing on this task makes it possible to take the first step to solve the remaining tasks. Furthermore, after creating this system in future work, it becomes easier to continue the development to include and validate more of the tasks currently being performed by the ninjas.

Considering this specific task that encompasses the creation of a system for the automatic validation of images, a set of small objectives were created to clarify what does the system need to fulfil for it to be considered finalised. Described next are all the created objectives:

1. Problem analysis.
2. State of the art analysis on CV tasks related with the problem.
3. Concurrent approaches analysis of systems' created by other companies.
4. State of the art analysis of the object detection models.
5. Selection of the object detection model best suited for creating an automatic system for image validation, using data from missions of the Brands&Ninjas platform and the same metrics for all compared models obtained, being these metrics obtained under the same conditions.
6. Creation of a system using the object detection model best suited for automatic validation of images.
7. Creation of an API for integration of the system for automatic validation.
8. Use the created API to validate at least an entire Brands&Ninjas mission, allowing it to be integrated by the Brands&Ninjas platform.

All the described objectives are sequential, meaning that in order to do the second objective, the first objective needs to be finalised, and the objective goal needs to be clearly achieved. The same happens for all the other objectives. The fulfilment of all these goals indicates that the system is finalised, successfully done and can be considered a feature of the existing Brands&Ninjas platform.

1.2 Document Structure

In this chapter was described the reason why there is a need for a system to automatically validate the images sent by the ninjas. The following chapters describe the steps to create such a system regarding validating the presence of only one object. **Chapter 2**gives some context to the diverse CV tasks while describing the best ways to solve the problem at hand. It also describes other solutions implemented by rival companies of RedLight and others, proceeding to introduce the current state of the art for object detection. **Chapter 3** describes the used approach and how the experimentation was structured. **Chapter 4** specifies the intended audience as well as the use of the solution created and, with that, the specific product requirements and risk analysis. **Chapter 5** corresponds to the description of the experimentation steps made for each of the selected object detection models and the respective results, which led to choosing the one that was considered the best suited. **Chapter 6** describes how the deployment was done after the selection of the best model. After selecting the best model in the experimentation chapter, it is described how the deployment of the final model was done and how the platform performed in action, analysing the results of validating three different missions. Finally, **Chapter 7** corresponds to the conclusions taken regarding building a system for automatic validation of images and the work that can be done in the future to improve the created system.

Chapter 2

State of the art

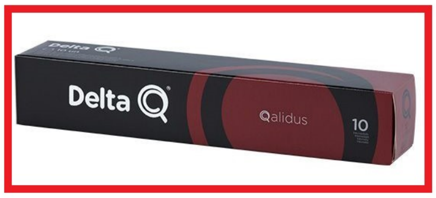
Till this point, an introduction has been made as to why there is a necessity for creating a system for automatic validation of all the images received by the ninjas, which has become impracticable if only done manually. This necessity led to the definition of the objectives that needed to be performed and solved to create the best possible automatic validation system of all images. This chapter describes the Computer Vision (CV) tasks best associated with the problem at hand, other systems created by companies considered rivals to RedLight and other companies' choices regarding similar problems in other areas. The last section of this chapter describes the current state of the art analysis regarding the object detection models for unconstrained environments since an object detection model is essential to create a system that can detect the presence of specific products in an image. Furthermore, using an object detection model allows us to solve one of the most common types of questions, which is if a specific object is on a shelf, by comparing the ninjas' answers with the responses obtained using an object detection model. Notably, in the context of the Brands&Ninjas, if there is a system that can detect the objects present in an image, it is possible to validate and automatically answer with some certainty if a specific object is indeed on the shelf.

This chapter is composed of three sections. The first section, called context and definitions, gives some context to the field of CV, describing some definitions essential to understand the following sections. It also has the objective of showing why object detection is the most appropriate methodology to create the first approach to solve the problem of creating a system for automatic validation of images, regarding the specific problem of knowing if a product is indeed on an image of a shelf. The second section, called related solutions, describes the similar solutions made by direct or indirect Brands&Ninjas competitors and other work essential to construct the objective system. The third and final section of this chapter, called state of the art on object detection, highlights the state of the art models and breakthroughs in object detection.

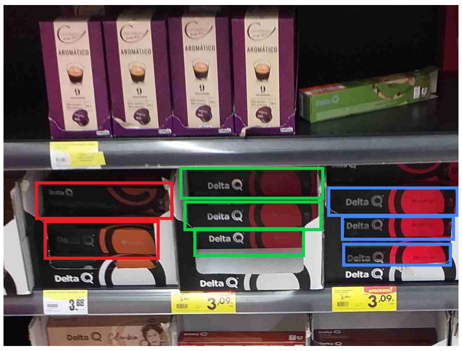
2.1 Context and definitions

CV corresponds to an Artificial Intelligence (AI) field that studies how can artificial systems extract information and gain a high-level understanding from digital images and videos, understanding what is seen and extract complex information in a way that can be used in other processes. CV is a field composed of several tasks, including methods for acquiring, processing, analysing and understanding digital images. In this context, understanding corresponds to the transformation of the input image into descriptions of the world that make sense to the artificial system. The CV tasks that are more related to building the considered objective are image classification, image classification with object localization, object detection, and instance segmentation.

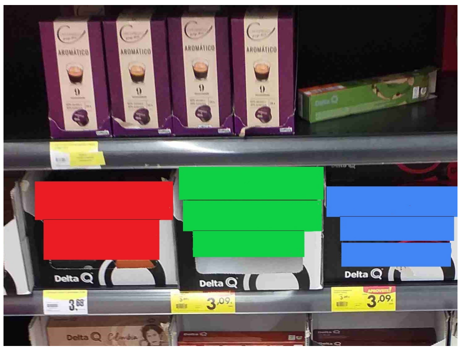
(a) Example of image classification: In this example, the image would be classified as belonging to the Qalidus class.



(b) Example of image classification with object localization: In this example, the image would be classified as belonging to class Qalidus with the red bounding box identifying the object used to label the image.



(c) Example of object detection: In this example, the detector found objects of three different classes, aQtivus, Qalidus and Qharacter, from left to right, respectively.



(d) Example of instance segmentation: In this example, the detector detected the objects and their shape, having found objects of three different classes and their respective shapes, aQtivus, Qalidus and Qharacter, from left to right, respectively.

Figure 2.1: Examples of the use of the CV tasks considered.

Image classification corresponds to a CV task that can help identify the content of a given image by associating each image with a respective class, labelling it. A class in CV corresponds to a set of objects that one intends to identify and that can be aggregated under the same designation. So, for example, all products of coffee from the Delta Q Qalidus package of 10 capsules can be associated with the class name Qalidus. Figure 2.1a corresponds to an image classification example on an image that associates the class with the name Qalidus to it, labelling it as belonging to the Qalidus class.

Image classification with object localization is a CV task that allows to classify a given image's content and localizes the found object, classifying the respective image and associating it with a class. For example, for the image in Figure 2.1b, the image is associated with the class with the name Qalidus, and the object that helped associate the image with that class is inside the drawn bounding box in red.

Object detection corresponds to a CV task to classify several objects in the same image. Object detection allows contrarily to what happens with both image classification and image classification with object localization, where each image is only associated with a class, to detect multiple objects from multiple different classes in the same image. For example, in Figure 2.1c, eight objects were found using an object detection model from three different classes. In this example, the bounding boxes in red correspond to the objects found of the class aQtivus, the Delta Q aQtivus package of 10 capsules, the bounding boxes in green correspond to the found objects of the class Qalidus, and the bounding boxes in blue correspond to the found objects of the class Qharacter, the Delta Q Qharacter package of 10 capsules.

Instance segmentation corresponds to a CV task that, besides classifying and detecting multiple objects in an image, can find the shape of the detected objects. For example, for the image in Figure 2.1d, eight objects were found from three different classes and their respective shapes using an instance segmentation model. The shapes in red correspond to the objects found of the class aQtivus, the shapes in green correspond to the objects found of the class Qalidus, and the shapes in blue correspond to the objects found of the class Qharacter.

Considering all the CV tasks described above and the objective of creating a system for automatic validation of images, wherein each of these images to validate taken by the ninjas are multiple objects that can be of multiple different classes. Both the Image classification and Image classification with object localization cannot be used to find multiple objects, meaning that these two are not suitable to create the considered system and need to be discarded, so it remains the object detection and instance segmentation CV tasks. Both can detect and localize multiple different objects from multiple different classes, being the difference between these two CV tasks the ability of the instance segmentation task to identify the object's shape at the cost of extra calculations. So image segmentation for this particular problem would not be the best option since the object's shape is unnecessary and would require extra calculations. Considering this, the best-suited CV task for constructing the considered system is object detection.

Above, it was described why object detection is the best CV task to construct the proposed system. The next step is to analyse some of the possible metrics commonly used for comparing object detection models, and for that, it was made an analysis of the possible metrics, being this analysis described next. Finally, for a more complete overview of all the metrics described next and how they compare each against each other, a review of R. Padilla et al.[18] article is recommended.

At a low level, measuring the performance of an object detector involves determining if a detection is valid or not. To do so, the use of the Intersection over Union (IoU) is essential. Thus, resulting in the appliance of the precision, recall, and F-score metrics.

The IoU evaluates the overlap between the ground-truth mask and the predicted mask, and it is calculated as the area of intersection between the ground-truth mask and the predicted mask divided by the area of the union of the two, as shown in figure 2.2. IoU ranges from 0 to 1, in which 0 means no overlap whatsoever and 1 corresponds to the perfect overlap between the annotated example and the prediction made by the object detection model. Therefore, using the IoU metric always needs a threshold to distinguish a valid detection from a non-valid detection.

The analysis of true negatives is not used in object detection problems because there are infinitely many instances, and these instances should not be detected as an object. Therefore, the metrics from the confusion matrix such as True Negative Rate (TNR), False Positive Rate (FPR), Negative Predictive Value (NPC) and the Receiver Operating Characteristic (ROC) curve are usually avoided. Instead, the evaluation of object detection models is based on the precision and recall metrics together with the use of the IoU

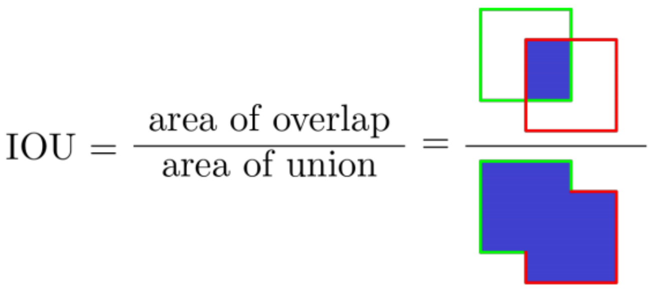


Figure 2.2: Illustration of the IoU

regarding the validity of a detection.

The precision(represented in equation 2.1) corresponds to how accurate the considered model is in relation to the total predicted observations. The recall(represented in equation 2.2) corresponds to how sensitive the model is in relation to the total correctly predicted positive observations in the responses considered positive. Finally, the F-score(represented in equation 2.3) corresponds to the precision and recall weighted average. A good model corresponds to a high recall model, capable of identifying most ground-truth objects while only finding the relevant objects, high precision. A perfect model is one with both recall and precision equal to 1.

Although the precision, recall and F-score metrics are suitable for analysing an object detection model, these metrics do not consider different confidence levels, only considering a fixed confidence threshold. Thus, the confidence value corresponds to a value associated with a bounding box between 0 and 1, showing how confident the detector is in regard to that particular prediction.

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.1}$$

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.2}$$

$$\text{F-score} = 2 * \frac{\text{Precision } ^{*} \text{ Recall}}{\text{Precision} + \text{Recall}} \tag{2.3}$$

A solution for the fact that the precision and recall can only be calculated while considering one fixed confidence threshold is implementing the precision-recall curve that corresponds to a plot of the precision as a function of the recall. Thus, showing the trade-off between the two metrics for varying confidence values for the model detections. For a good model, both the precision and the recall should remain high even if the confidence threshold varies. An example of a precision-recall curve can be seen in figure 2.3.

Another important metric that results from the analysis of the precision-recall curve is the Average Precision (AP), which, ideally, corresponds to the area under the precision-recall curve and needs an IoU threshold to be selected for it to be used. Usually, the AP is commonly represented as AP@.50 or AP@.75, which means the AP is considering an IoU threshold of 50% and 75%, respectively. Thus, mathematically AP is defined as shown in

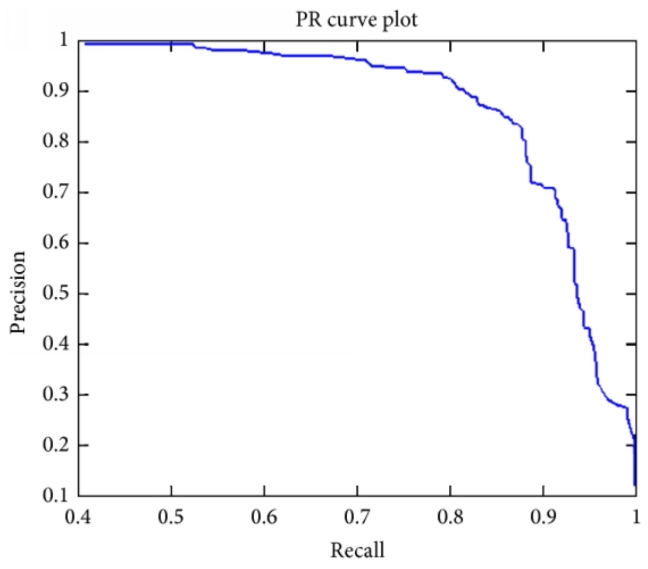


Figure 2.3: Example of the precision-recall curve.

equation 2.4. This equation can be used considering different methods, such as the 11-point interpolation and the all-point interpolation methods. The 11-point interpolation method uses the precision-recall plot of interpolated precision scores for model results at 11 equally spaced standard recall levels, namely, 0.0, 0.1, 0.2, .. 1.0. The all-point interpolation method interpolates through all the positions of the precision-recall plot.

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.4}$$

The AP is only calculated individually for each class, meaning that there are as many AP values as the number of classes. Therefore, it is possible to obtain a different metric to analyse an object detection model's capabilities to identify multiple classes by averaging these values. This metric is called the mean Average Precision (mAP) and corresponds to the average of the AP values over all classes.

Popular competitions such as the common objects in context (COCO)[15], PASCAL visual object classes (VOC)[3], and Open Images Dataset[12] all use different metrics to analyse the submitted object detection models. Being the Microsoft COCO's AP@[.5:.05:.95](AP calculated at 10 different IoU ranging from 50% to 95% at 5% step-size) and the PASCAL mAP metrics the most popular ones used as benchmarks. Considering that the COCO's AP@ [.5:.05:.95] is affected by different IoUs, it is impossible to evaluate the detector's effectiveness with a more or less restrictive IoU with this metric, something that the PASCAL VOC mAP allows. For a more strict evaluation regarding the likeness of the ground truth and detection bounding boxes, the choice usually falls back to the AP@.75.

Considering all the metrics exposed above when comparing object detection models, the choice to evaluate and compare the candidate models falls on the VOC mAP since it is one of the most applied with success for comparing object detection models. Also, it suits the way the system for automatic validation of images should work. Considering all these, the implementation to this metric that will be used is the one at the R. Padilla et al. GitHub repository[1].

[1]`https://github.com/rafaelpadilla/Object-Detection-Metrics`

In this section, were introduced several CV tasks and the reasons that led to the choice of object detection among them to construct the proposed system, and it was given some context to the metrics that are commonly used when analysing an object detection model. The following section describes some of the work done by competing companies to Brands&Ninjas for them to build intelligent systems that can be related to the system to be created, and other solutions not about the same problem or by competing companies but that can be related to the creation of the objective system.

2.2 Related solutions

Brands&Ninjas is a platform for remote audit of brands at the point of sale and in real-time. Brands&Ninjas auditors are called ninjas and use the Brands&Ninjas platform to have an extra source of income. These ninjas only have to go to specific stores, answer a set of questions, and take pictures that validate their answers. Brands&Ninjas way of using Crowdsourcing to audit the presence of the brands product's is not unique. Several other companies use a similar strategy, having developed some systems to validate the images taken by the auditors to have an extra dose of confidence in the answers given by them. Some of these companies strategies are described in this section. The first part describes the solutions taken by rival companies regarding the problem that made the creation of a system for automatic validation of images needed for the Brands&Ninjas. The second part describes other solutions from companies not in the same field but related to creating such a system.

Some of the companies that have created a system and use it to validate the images received are BeMyEye, VIDI, Trax, Roamler and Native.

BeMyEye[2] is a company based in the United Kingdom that has already expanded to multiple countries, including Portugal. Although their way of auditing the product's brands at the point of sale is similar to the method used by Brands&Ninjas, they also use Crowdsourcing to do this audit, where the auditors have to answer questions and take pictures to answer those questions. To validate the answers provided, BeMyEye bought in 2019 a Russian company called Streetbee, which allowed them to implement object detection models to validate the answers given by the auditors. Besides validating the received images, BeMyEye provides stores with what they call the "Perfect Store". That allows stores to track on-shelf availability and visibility(the presence and visibility of products for each shelf), promotional compliance(the appliance and results of promotions) and brand recommendation(ways to recommend the best brand for each customer needs).

VIDI[3] is a Brazilian company that, besides using Crowdsourcing, also uses internal auditors to evaluate the field team's execution work. Like BeMyEye and Brand&Ninjas, VIDI uses Crowdsourcing to audit the presence of the brand's products in-store and uses image recognition, in specific, object detection models to validate the answers provided by the auditors. In addition, VIDI also uses image recognition like BeMyEye to help conceive what they see as the "Perfect Store". Finally, they use internal auditors and external auditors to evaluate the sales of their products from a store's perspective and monitor shelf availability and visibility.

Trax[4] is a company based in Singapore that has expanded worldwide with a presence on all continents. Trax introduced in 2010 a new and revolutionising technology for image recog-

[2] `https://bemyeye.com/`
[3] `https://vidibr.com/`
[4] `https://traxretail.com/`

nition in the industry of consumer goods, with the objectives of sensitising for the meaning and value of image recognition and boosting a more significant efficiency to the customer goods companies. With the introduction of image recognition, Trax was the first company to allow brands to receive detailed information about products and categories, including products out of place, stock of products on the shelves and analyse the effectiveness of promotions, all using image recognition. Trax is the world leader in CV solutions for retail stores, convenience stores and supply retailers. Trax work and solutions are worldwide recognised and recreated by other companies.

Roamler[5] is a company based in the Netherlands that has expanded across multiple European countries. Using a similar model to the one used by Brands&Ninjas to audit for numerous brands at the point of sale, although similarly to Brands&Ninjas so far and contrarily to the companies described above, the validation of the submission of the auditors is manually verified, having workers responsible for doing so 24 hours per day, seven days a week.

Native[6] is a company based in the United States of America that works worldwide. Native uses Crowdsourcing to allow brands to monitor product availability and product ageing, down to the store level, missing products or misplaced products in stores, track promos, prices and see how the brand is presented in-store. All these possible tasks are sent to the auditors that answer the questions specified for each task, where Native uses image recognition to verify all the answers where an image was provided. In particular, it uses object detection to analyse the present objects in the provided images.

As seen in the competitors described above, all except for Roamler, which verifies their received images by hand, use an intelligent system that uses image detection and, more specifically, object detection. Thus, allowing them to verify the provided pictures and create reports to each brand from each mission with optimal accuracy. The use of object detection to create their intelligent validation system by rival companies to Brands&Ninjas demonstrates that the choice of object detection models to solve the considered problem is the best option. Moreover, companies that are direct or indirect competitors with Brands&Ninjas have developed different but simultaneously similar solutions for the same problem. Furthermore, many other companies have created solutions that can be used to prove that using object detection is indeed the best approach to tackle this problem. Three of those companies are Landing Ai, Emberion and Argutec, and their solutions to three different issues are described below.

Landing Ai[7] is a company based in the United States of America that uses object recognition technologies and powerful camera equipment to ensure the quality of finished goods, facilitating the quality checks of the various production processes that a company can have. Their goal is to gradually teach the system to use cameras to consistently detect defects on the end product.

Emberion[8] is a spin-off from Nokia's research and development unit that employs object detection models to help sort plastic in waste management industries and goods for agricultural industries. The need for Emberion to use these types of models came from the traditionally required intensive labour costs and almost always included the possibility of a human error factor. The use of CV allows to eliminate this and automate the process of product sorting. Automated product sorting is especially relevant for the agriculture and waste management sectors.

[5]`https://www.roamler.com/en/main`
[6]`https://www.native.io/`
[7]`https://landing.ai/`
[8]`https://www.emberion.com/applications/`

Argutec[9] is a company based in the Czech Republic that understood that the development made in the manufacturing industries to allow for mass production made their assembly lines incredibly complex. Until recently, the assembly process for high-tech industries, such as aerospace and automotive, relied on various scripts and scenarios pre-programmed into the machines despite being automated. AI powered CV makes the assembly line more flexible, allowing the machines to make decisions based on objects that it detects at all stages of the assembly. Considering all this, Argutec developed machine vision solutions that use, among others, object tracking technology to enhance the assembly process and automate production even further. For example, in a car assembly line, it can determine what part of a car is at what particular stage, whether it has all of the components to be transferred to the next assembly step or not.

This section showed some of the solutions taken by other companies that are Brands&Ninjas competitors and how they also rely on object detection to validate the images provided by the auditors. Additionally, it highlighted the important role that object detection models are having when it comes to creating better solutions for other problems in different fields, allowing to decrease the possible errors made by the human error factor. The following section will describe the current state of the art models on object detection.

2.3 State of the art on object detection

As seen so far, object detection is the most suited solution for creating an intelligent system for automatic validation of the images taken by the ninjas. This section describes the current state of the art models on object detection and the models that will be used to create the objective system.

Over the years, there have been multiple approaches to object detection. For a complete overview of its evolution, review Zhengxia et al. survey[31]. Usually, the history of object detection models is divided into two different periods. The first is designated as the "traditional object detection period" and is associated with models made before 2014. The second period defined as the "deep learning-based detection period", is related to models after 2014. This distinction in two different periods is made accordingly to the one made at the Zhengxia et al. survey[31].

2.3.1 Traditional object detection period

In the early 2000s, due to the lack of effective image representation and the limited computing resources at the time, most of the early object detection algorithms were built on handcrafted features. A framework of object detectors created at the beginning of this period that uses handcrafted features is the so-called Viola Jones detectors[29]. This framework, developed in 2001 by Paul Viola and Michael Jones, allows real-time detection of human faces. The Viola-Jones detectors use a sliding window that searches in all the image for Haar-like features to see if any part of the image of the size of the sliding window contains a human face. The Haar-like features are an adaptation made by Paul Viola and Michael Jones of the haar wavelets that were named after Alfred Haar, who proposed in 1909 the concept of haar wavelets and can be used to analyse digital signals and compress digital images[25]. These haar-like features consider adjacent rectangular regions in the detection window at a specific location, adding the pixel intensities in each region and calculating the difference between these sums. The resulting difference is used to categorise

[9]`https://www.argutec.eu/`

subsections of an image. An example of haar-like features and the rectangular considered areas can be seen in figure 2.4.

A common haar-like feature for the face detection problem is a set of two adjacent rectangles above the eye and the cheek region. In the detection phase, a sliding window of fixed size is moved over the input image, and for each subsection of the image, the Haar-like feature is calculated. If the resulting value is above the defined threshold is considered a found face. If not, it is regarded as a non-face for the problem of face detection. Considering that using only a Haar-like feature results in a weak classifier, whose quality is slightly better than random guessing, many Haar-like features are necessary to create a more robust object detector. The Viola–Jones object detection framework organises these features in a cascade classifier to form a robust classifier. A cascade classifier corresponds to a particular case of ensemble learning based on the concatenation of several classifiers, using the information collected from the classifier's output as additional information for the next classifier. It is using the responses given by all classifiers that the cascade classifier can answer more correctly. To improve detection speed, the Viola-Jones detectors implement three essential techniques. First, the integral image technique that corresponds to the creation of a Haar-like feature representation of the image to detect while using the sliding window, passing these representations as an integral image allows to make the computational complexity of each sliding window independent of its window size since cropping multiple pictures and passing them through the Convolutional neural networks (CNN) is computationally very expensive. The second improvement made to improve the detection speed was the use of the Adaboost algorithm that selected a small set of features that were the most helpful for face detection. Besides that, a multi-stage detection paradigm called Detection Cascades was also used to reduce the model computational overhead by spending fewer computations on background windows and more on face targets. Although the Viola-Jones family of detectors introduced significant improvements at the time, they could not detect human faces if any constraints were used, like skin colour segmentation. Furthermore, since the sliding window is of a fixed size, it produces inaccurate bounding boxes.

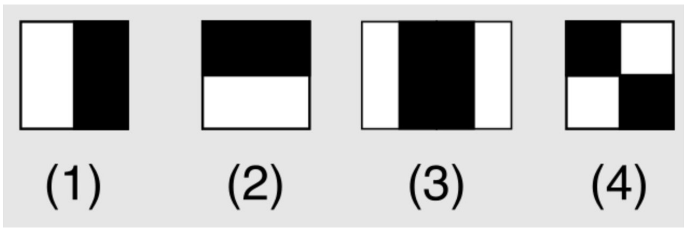


Figure 2.4: Example of Haar-like Features, Two rectangle features are shown in (1) and (2), three rectangle features in (3) and four rectangle features in (4)

Another object detector that was a breakthrough in this era was the Histogram of Oriented Gradients (HOG) that was originally proposed in 2005 by N. Dalal and B. Triggs[2]. It was an improvement of the scale-invariant feature transform and shaped contexts of its time. The HOG uses a sliding window similarly as it occurs with Viola Jones detectors, which it calls blocks, that correspond to a pixel grid in which the gradients are constituted from the magnitude and direction of change in the intensities of the pixels within the block. HOG classifiers are widely known for their use in pedestrian detection. There are also other known implementations like the one Schmidt et al.[23] proposed in 2007 and applied to face and eyes detection. The HOG detector rescales the input image multiple times to detect objects of different sizes while keeping the detection window size unchanged.

The peak of traditional object detection models came with the Deformable Part-based Model (DPM) originally proposed by P. Felzenszwalb[4] in 2008 as an extension of the

HOG detector. Later a variety of improvements were made by R. Girshick[5, 8]. The DPM uses a divide and conquer strategy. For example, the problem of detecting a car can be solved by detecting its window, body and wheels. Considering this, the training process of this detector involves learning a proper way of decomposing an object, and its inference involves ensembling detections of different object parts. A DPM consists of a root filter and several part-filters. A weakly supervised learning method is developed, where all the configurations(size, location, etc.) of part filters can be automatically learned as latent variables. The improvements made by R. Girshic had the objective of improving the detection accuracy. For that, he used a particular case of Multi-instance learning and other essential techniques, such as hard negative mining, bounding box regression and context priming. To improve the detection time, R. Girshic also implemented a cascade architecture that allowed to achieve up to 10 times improved detection time without losing accuracy.

A method to represent the features of an image using handcrafted features is the Bag of features, also known as Bag of visual words. The Bag of features is inspired by the Bag of words model that is usually used in natural language processing (NLP). The feature extraction by this method is divided into three different steps. The Feature extraction step is the first step, where all the raw features are extracted from all the images in the training dataset. This feature extractor is usually done using Scale Invariant Feature Transform (SIFT). For reference on feature extraction in CV, please refer to David et al. paper[17]. The second step corresponds to the codebook generation, where all descriptors are clustered using a clustering algorithm. The resulting clusters represent the main features present in the whole training dataset, often known as the codewords or the visual words. The third and final step corresponds to the feature vector generation. This step uses a test image to obtain a vector that contains the frequency of each codeword. This method for extracting handcrafted features is usually done to solve problems like content-based image retrieval (CBIR), which corresponds to finding an image in a database that is the closest to a query image. For a complete understanding of the functioning and use of the Bag of features, check the Chih-Fong Tsai review[28].

In the period designated as the traditional object detection period and considering the limited resources available, all the features used for object detection resulted from handcrafted features like the Bag of features, Haar-like features or through creating a histogram of oriented gradients. And with this, from 2008 to 2012, object detection reached a plateau as the performance of handcrafted features became saturated, as occurred minor gains from building complicated ensemble systems, showing the limitations of traditional object detectors.

2.3.2 Deep learning based object detection

In 2012, a rebirth of the CNN happened as the use of Deep Neural Networks (DNN) successfully learned robust and high-level feature representations for image classification. The deadlocks of object detection were broken in 2014 by the proposal of R. Girshick et al.[7], the Regions with CNN features (RCNN) for object detection that corresponds to a region-based CNN detector. Compared to traditionally handcrafted feature descriptors, DNN generates hierarchical features and capture different scale information in different layers, producing robust and discriminative features for classification, utilising the power of transfer learning. Object detectors are grouped into two genres in this deep learning era: the two-stage detectors and the one-stage detectors. The two-stage detectors are constituted by a pose estimation step and a detection step separated by a Region of Interest (RoI). The one-stage detectors, on the other hand, only have a pose estimation step.

Usually, one-stage detectors are associated with achieving better efficiency at the cost of sacrificing accuracy.

The RCNN detector is a two-stage detector that starts by extracting a set of object proposals or object candidate boxes using selective search. Then each proposal is rescaled to a fixed size and fed into a pre-trained CNN model to extract its features. Finally, linear Support vector machine (SVM) classifiers are used to predict the presence of an object within each region and recognise the object class. Although RCNN does much better than traditional detection methods, it has several drawbacks. First, the selective search algorithm is a fixed algorithm, causing no learning to happen at that stage, leading to the generation of bad candidate region proposals. Second, it still takes a considerable amount of time to train the network as it is needed to classify up to 2000 region proposals per image. Leading to extremely slow detection speed, meaning that it cannot be implemented in real-time as it takes more than 40 seconds for each test image.

A Year later, intending to tackle RCNN's problems, the same author of RCNN R. Girshick proposed a faster version of the RCNN detector called Fast RCNN[6]. This detector's approach is similar to the RCNN detector algorithm. A significant change is that instead of feeding the region proposals to the CNN, the input image is fed to the CNN to generate a convolutional feature map. The region proposals are identified using the convolutional feature map and wrapped into squares. With the use of the RoI pooling layer, the region proposals are reshaped into a fixed size so that they can be fed into a fully connected layer. Using a softmax layer from the RoI feature vector, it is possible to predict the class of the proposed region and the offset values for the bounding box. The reason that makes Fast RCNN faster than the RCNN detector is the fact that it is not needed to feed the 200 region proposals to the CNN every time. Instead, the convolution operation is done only once per image, and a feature map is generated from it. With this new version, the training time decreased from over 80 hours to about 8.75 hours. In addition, the testing time per image decreased from more than 40 seconds to about 2.3 seconds, including the region proposal time and 0.3 seconds without including the region proposal time. This difference verified with and without the region proposal's time shows how the region proposals are a limitation in the Fast RCNN algorithm affecting its detection performance.

Shortly after the Fast RCNN, S. Ren et al. proposed the Faster RCNN[22] detector, the first near real-time deep learning object detector. All of the above two-stage deep learning detectors use selective search to find out the region proposals. However, it is considering that using selective search is a slow and time-consuming process that affects the network's performance. Faster RCNN eliminates the use of the selective search algorithm allowing the network to learn the region proposals. Similarly to the Fast RCNN, the image is provided as an input to a CNN, which provides a convolution feature map. To replace the selective search algorithm, a separated network is used to predict the region proposals. The predicted region proposals are then reshaped using an RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. By being part of the Faster RCNN model, the region proposal network is trained with the rest of the model. With this, the region proposal network can learn how to generate high-quality proposed regions, which reduces the number of proposed regions while maintaining the precision of the object detector, allowing Faster RCNN to achieve a testing detection time up to 10 times faster than Fast RCNN.

One of the Faster RCNN problems is the detector's inability to detect small objects in an image. To solve this problem, it was proposed by T. Y. Lin et al. a Feature Pyramid Networks (FPN)[13] architecture. A visual representation of the FPN can be seen in figure 2.5. This architecture creates a simple image pyramid that can scale the image to different sizes and send it to the network. Once the detection occurs on each scale, all the predictions can be combined using other methods. Before the FPN architecture, most deep learning detectors executed detections only on the network's top layer. Although the features in deeper layers of a CNN benefit category recognition, it is not conducive to localising objects. Since a CNN naturally forms a feature pyramid through its forward propagation, the FPN architecture shows significant advances for detecting objects with a wide variety of scales. FPN has now become an essential building block of many of the latest detectors.

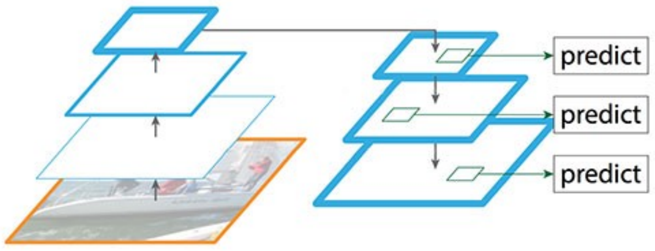


Figure 2.5: Representation of the FPN architecture

The detectors described so far are two-stage detectors that use regions to localise the object within the image. With that, the network does not look at the entire image. Instead, it only looks at the parts of the image with high probabilities of containing the object. The You Only Look Once (YOLO) object detector proposed by Redmon et al.[19] is a one-stage detector that trains on full images and directly optimises detection performance. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and the class probabilities for those boxes. To do so, YOLO divides the input image into a S x S grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for these boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks its predicted box is. The anchor boxes are chosen by exploring the training data to select reasonable height/width ratios representing the different classes. To get rid of many incorrect bounding boxes with a low probability score it is used a threshold. However, just with the use of a threshold for the scoring probability, the detector still predicts a lot of bounding boxes. So it is used Non-maxima suppression (NMS) in which only one bounding box is selected when several boxes overlap with each other and detect the same object. Despite its significant improvement in detection speed, YOLO has a drop in the localisation accuracy compared with the two-stage detectors above, especially for small objects.

Released after YOLO and RCNN, the proposed Single Shot MultiBox Detector (SSD) detector by W. Liu et al.[16] reached new records regarding performance and precision for object detection tasks. The SSD detector is a one-stage object detector, just like YOLO. SSD's main contribution is introducing multi-reference and multi-resolution detection techniques, which significantly improves the detection accuracy of a one-stage detector, especially for small objects. The SSD detector has two versions, one that uses 300*300 input size images called SSD300 and another version called SSD512 that uses 512*512 input size images able to surpass the Faster RCNN results. This detector uses Multi-scale feature layers that are a series of convolution filters added after the base network. This base network corresponds to a standard pre-trained network, and some of the networks that achieved good results when added to the SSD detector were the VGG16 network[24] and the ResNet network[10]. The Multi-scale feature layers decrease in size progressively to allow the prediction of detections at multiple scales.

Usually, as described at the beginning of section 2.3.2, one-stage detectors are associated with achieving better efficiency at the cost of sacrificing accuracy. Considering this and after discovering that there is a severe class imbalance problem during the training of dense one-stage object detectors, that is considered to be the main cause of one-stage detectors' inferior performance concerning two-stage detectors. T. Lin et al. proposed a new loss function called focal loss and introduced it in the RetinaNet detector[14]. Lower loss is contributed by easy classified negative samples, allowing the detector to focus on misclassified examples during training. The use of focal loss enables one-stage detectors to achieve comparable accuracy with two-stage detectors while maintaining their high detection speed. RetinaNet uses the Resnet network together with the FPN network as the backbone for feature extraction, plus two task-specific sub-networks for classification and bounding box regression. With this, RetinaNet is able to achieve a state of the art performance and outperform well-known two-stage detectors like Faster RCNN.

The YOLO family of detectors continued their development with the update made by Joseph Redmon and Ali Farhadi called YOLOv2[20] to further improve model performance. Although this new version is usually referred to as YOLOv2, an instance of this model is considered to be capable of predicting 9000 different object classes, to what was given the name YOLO9000. Multiple training and architectural changes have been made to this new YOLO version, such as batch normalization and the use of high-resolution input images. Besides these improvements, it was also implemented, like in the Faster RCNN, the use of anchor boxes, that are pre-defined bounding boxes with useful shapes and sizes adapted during training. The choice of the bounding box is pre-processed using a k-means analysis on the training dataset. Furthermore, to create a more stable detection model, YOLOv2 allows small changes to have a less dramatic effect on the predictions by changing the predicted representation of the bounding boxes. All these new alterations to the YOLO family introduced a new backbone called Darknet-19 that improved YOLO's speed and precision.

YOLOv2 could not use multi-label classification. Multi-label classification allows multiple labels to be assigned to each instance. There is no constraint on how many labels can be assigned to each instance in the multi-label problem. A new version called YOLOv3[21] was proposed a year later with the objective of introducing the use of multi-label classification (independent logistic classifiers) to adapt to more complex datasets containing many overlapping labels without sacrificing accuracy and efficiency. The improvements to this new YOLO version were minor. Besides using multi-label classification, it was also included in this new version a deeper feature detector network and some minor representational changes.

Although Joseph Redmon left the YOLO development project after the YOLOv3 detector because of its military uses, a new version called YOLOv4 was proposed by A. Bochkovskiy et al.[1]. The new YOLOv4 model introduced a detector that applied many of the previous developments to create a more complete one-stage detector that could be used in any Graphics Processing Unit (GPU) for both training and testing and still achieving real-time, high quality and convincing object detection results. To do so, the YOLOv4 detector uses as a backbone the ResNet network and the VGG16 network, among others, as feature extractors. These networks are pre-trained on image classification datasets like ImageNet[10] and then fine-tuned on the detection dataset. Furthermore, the YOLOv4 neck uses just like YOLOv3, an FPN and other pyramid networks to extract features of different scales from the backbone. Besides this, the YOLOv4 implements two categories of methods used to improve the object detector's accuracy, the Bag of freebies (BoF) and the Bag of specials

[10]`https://www.image-net.org`

(BoS).

The BoF corresponds to a set of methods that only change the training strategy or increase the training cost. These methods aim to make the resulting object detector achieve better accuracy without increasing the inference cost. Some of these methods are data augmentation techniques such as CutOut(randomly mask out square regions of the input image during training)and Random Erasing(selects rectangle regions in an image and erases its pixels). Other data augmentation techniques to prevent overfitting are DropOut, DropConnect and DropBlock. Besides the data augmentation techniques described, so far were also introduced two new data augmentation techniques called Mosaic and Self-Adversarial Training (SAT). The Mosaic process mixes four different training images into one image. The SAT operates in two forward-backwards stages. In the first stage, the neural network alters the original image instead of the network weights. This way, the neural network executes an adversarial attack on itself, transforming the original image to create the deception that there is no desired object on the image. In the second stage, the neural network is trained to detect an object on this modified image in the usual way. The BoS corresponds to plugin and post-processing methods that increase the inference cost by a small amount but significantly improve the object's detector accuracy. For that, YOLOv4 implements three different models the Spatial Pyramid Pooling (SPP)[9], the Squeeze-and-Excitation (SE)[11] and a modified version of the Spatial Attention Module (SAM)[30]. All these improvements make the current YOLOv4 the state of the art one-stage detector.

Later, after the YOLOv4 object detector launch, a so-called YOLOv5 detector was launched by Ultralytics[11] without any paper about its specifications, neither a comparison between this new YOLO model nor the YOLOv4 version. Then, a while after the release of the YOLOv5 version, Ultralytics came out. They said that the released model was neither static nor complete. It is still in development, and they intended to create a new YOLO model from the YOLOv3 model and that the given name was the internal name of the development model and not a final version.

The current state of the art for two-stage object detectors is the EfficientDet object detector proposed by Mingxing Tan et al.[27], since the EfficientDet object detector is the one that was able to achieve the best performance among peer models relative to model size on the COCO dataset for image detection. The EfficientDet model introduced a new family of scalable and efficient object detectors and implemented further model architecture optimisations. It starts by using as backbone an EfficientNet[26] that allowed the detector to achieve much better efficiency. Another implemented optimisation in the EfficientDet detector is the implementation of a new bi-directional feature network (BiFPN) that incorporates the multi-level feature fusion idea from the FPN, allowing to solve the limited one-way information flow. By combining the EfficientNet backbone with the BiFPN, Mingxing Tan et al. developed a small size EfficientDet-D0 considered the baseline. After applying a compound scaling, the EfficientDet detector can obtain new models from EfficientDet-D1 to D7. Each consecutive model has a higher compute cost ranging from 3 billion FLoating-point Operations Per Second (FLOPs)) to 300 billion FLOPs and provides higher accuracy

2.4 Overall Conclusions

This second chapter, particularly in the first section, gave some context regarding the possible choices to build the objective system and then described the reasons that led to the

[11]`https://ultralytics.com/`

choice of object detection as the CV task most suited to create the objective system. In the first section were also introduced the metrics usually associated with the comparison of object detection models. The second section described the related solution made by Brands&Ninjas rival companies for implementing similar systems, and then some other companies not directly related to Brands&Ninjas, which have created systems to solve problems that needed to identify the objects involved to solve the issues considered. Finally, the third section described the current state of the art object detectors in object detection, from the object detectors from the traditional object detection period to the current deep learning-based detection period where both the two-stage detectors and the one-stage detectors were described.

Four different object detection models were selected after the analysis made regarding the current state of the art object detection models in section 2.3. The selection of these models was made with the objective of creating a system the most automatic possible that allowed to be scaled in the future for more than one object, with an easy configuration and good performance. The selected models were the Haar feature-based cascade classifier and the HOG classifier that were chosen from the period defined as the traditional object detection period. These two models were selected for being simpler models that use only a Central Process Unit (CPU) and are usually compared between each other to see which one is the best to solve an object detection problem. The objective of choosing these two models was to verify if these models could be used to solve this considered problem and compete with the more recent models. The other two selected models were the YOLOv4 and the EfficientDet object detectors, being the current state of the art models for both the one-stage and two-stage detectors, respectively. These two models are the ones with the lowest detection and training time, as well as the ones with the best accuracy regarding the competitions they were in. Contrarily to the other two selected models, the deep learning models allow to easily annotate and train models using a small sample size, particularly the YOLOv4 object detector.

The next chapter describes the approach taken to solve the problem of creating a system for the automatic validation of images.

Chapter 3

Approach

Thus far, Chapter 1 referenced the problems that arise from not validating the images sent by the ninjas to the brands and ninjas platform along with the objectives established in section 1.1 regarding how the validation system should function. Chapter 2 described the state of the art analysis made regarding object detection. Finally, this chapter describes the approach considered to fulfil Chapter 1 objectives regarding the image validation problem for the images received by the Brands&Ninjas platform.

3.1 Framework proposal

An overview of the proposed framework architecture proposal is demonstrated in Figure 3.1. The system starts by receiving three different inputs, the tasks to analyse, the responses obtained by these tasks and the model to analyse each of the considered tasks. The first input received are the tasks that should be analysed and correspond to any of the possible assignments asked to a ninja to perform during a mission. For example, these tasks could be something like, is a specific product on the supermarket shelves or something completely different, like if a particular promotion is correctly identified and within the specific place where it should be placed. The task that the final system should be able to answer corresponds to the validation of the presence of the products represented by the class Qalidus on an image. The second input is the ninjas' responses to each of the assignments performed during that specific mission that will be validated. These responses correspond to the answers given by the ninjas to the asked questions and the images taken. That could be used for validation of the answers provided by them. The third and final input corresponds to the information about which models should be used to validate each task regarding the ninjas' responses.

The AI validator corresponds to the automatic validation system. If the specific task requested to be validated is already implemented, the AI validator can, after receiving the inputs necessary, analyse the images provided by the ninjas' using the model indicated and compare the answers given by the ninjas with the results obtained with the analysis of the images provided. If both the answers given by the ninjas and the results are the same, then the image is considered as validated, on the other hand, if the answers differ, then the image is not validated.

In the case that the image is not validated, the approach for this particular system will be to notify the user that requested the validation to do a manual analysis so that it is possible to understand who made the error, if the system, if the ninja.

After the description made in this chapter regarding the approach taken to create the objective system, the next chapter specifies the requirements for building a system that follows this approach and can validate this specific task. Being this specific task, if an object of the class Qalidus is present or not on an image provided, comparing it with the ninjas' answers. The proposed system to validate the images received will be focused on the detection of products on shelves. Therefore, the system should be remotely accessed and incorporate the best object detection model for the considered problem from the chosen models with the state of the art analysis.

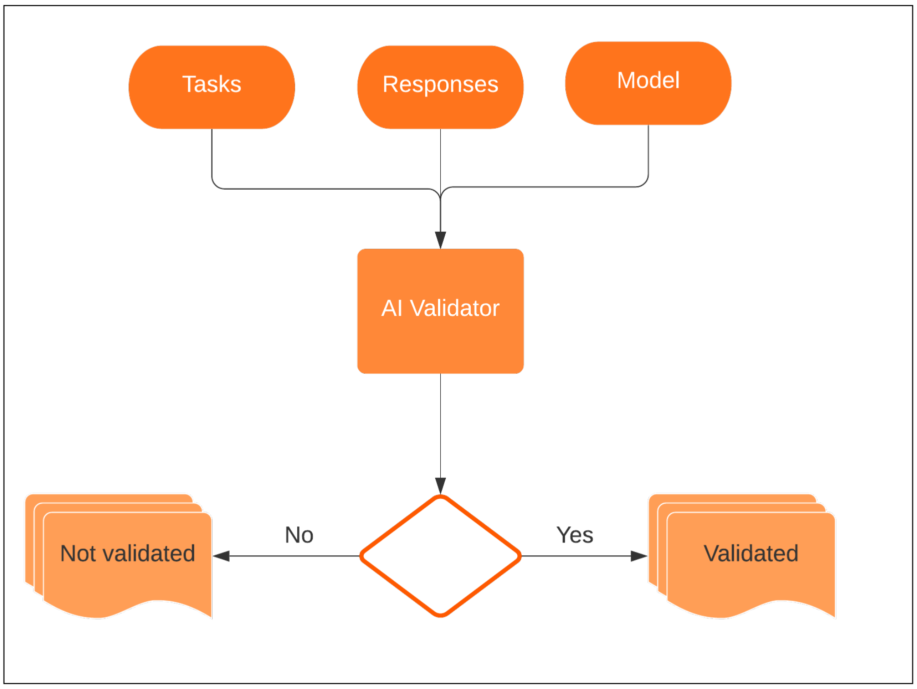


Figure 3.1: Framework proposal architecture

Chapter 4

Specification

This chapter describes the purpose of the system to build, whom the system is intended to be used by, and the systems intended use. Besides that, this chapter also describes the system requirements and the risk analysis made.

Considering that the system is only going to be constructed with the objective in mind of solving one of the most common tasks the ninjas need to perform for the Brands&Ninjas, that is to verify if a specific product is on a shelf by answering questions regarding their presence and send images that validate these answers. Therefore, the final objective of the work described in this dissertation is to build a system that can validate the images sent by the ninjas, comparing their given answers with the objects found in these images and answer for each image if the image is validated or not.

Any person can use the system inside the RedLight organisation, so this system needs to be both simple for any person in the RedLight organisation that needs to use it and abstract so that the system can be used without any specific knowledge about how it works or what happens inside the system. It needs to be simple since the RedLight organisation has persons with multiple backgrounds that might not know precisely what the system is doing. These persons need to be able to use the system to validate images if necessary. On the other hand, the system needs to be abstract for the user since what happens inside the system does not need to be known by the user that only tried to validate some images. For the user who requested the validation, it should be as simple as sending some images, and the images report with the ninjas' answers for each image and receive an answer for each image if that particular image is or is not validated.

Considering the objectives described above, a requirements description and analysis is presented next in section 4.1. However, with the construction of a system like this with existing time constraints, there is a high probability that problems will occur. So and in order to minimise the occurrence of certain risks and their effects, besides doing a requirements specification, it is also essential to do a risk analysis to minimise and prevent these problems. For this particular system, this risk analysis is described in section 4.2.

4.1 Requirements

This section describes how the system for automatic validation of images should be built and the requirements that the constructed model should meet, considering the system's necessity to remain abstract and straightforward to be used by anyone on the RedLight organization.

The system to build corresponds to a new product that is currently not a part of the Brands&Ninjas platform. However, although it is a new product, it can be seen as a future add-on to the Brands&Ninjas platform, since if correctly applied, it can become a part of the platform having a fundamental role in validating the images taken by the ninjas. Therefore, it is not supposed to be used by any other person outside the RedLight organisation, nor even the ninjas that performed the missions. For this system to be deployed and put to use together with the Brands&Ninjas platform, it assumes that the RedLight can provide a virtual machine that allows to deploy and use the system. Without this, it is impossible to test if the system can work remotely.

Next are presented the three system requirements in a table format, that correspond to tables 4.1, 4.2 and 4.3. The three considered system requirements are composed of twenty-eight associated tasks and were used as the guideline for all the work performed in the creation of a system capable of validating images received by the Brands&Ninjas platform regarding the presence of objects of the class Qalidus. All the created and present requirements are sequential, meaning that the inputs of a requirement are the outputs of the previous requirement, except for the first requirement, where the inputs are the images of previously performed images by RedLight.

Table 4.1: The first requirement specification

Requirement ID	Req01
Title	The system should comprise a training module to allow systematic and straightforward training of each considered model.
Description	The system should be able to train every selected model using a common dataset, and each model to be trained should follow the same steps, ensuring that all the trained models respect the same conditions and are trained in the same environment. The inputs of this requirement are the images of performed missions provided by RedLight. This requirement can be divided into two different parts. The first one corresponds to the creation of both a training and a testing dataset that should be used for training all of the selected models and the definition of all the training parameters common to all models. In particular, several data augmentation techniques to improve the robustness of the training dataset should be selected and implemented. The second part of this requirement corresponds to the steps that every object detection model to train should take, from the selection of the model to train to its effective training and execution on the testing dataset and storage of the resulting results for further analysis. The first part of this requirement is accomplished when both the training and testing dataset are finalised and the common parameters to all models have been defined. The second and last part is considered accomplished when all the chosen models have been trained.
Associated Tasks	T1. Selection of the images to be used as the dataset. T2. Annotation of the images where the objects of the class Qalidus are present. T3. Division of the dataset into the training and testing dataset. T4. Data augmentation techniques selection. T5. Data augmentation implementation on the training dataset. T6. Training parameters definition. T7. Model selection for training. T8. Definition of the training parameters of the model. T9. Dataset Adaptation to the training model specificities. T10. Training of the object detection model. T11. Saving of the trained object detection model. T12. Execution of trained model on the testing dataset. T13. Saving of the results obtained from executing the model on the testing dataset.

Table 4.2: The second requirement specification

Requirement ID	Req02
Title	The system should include an image validation module to allow the selection and use of the best object detection model.
Description	Training object detection models do not correspond to the creation of a system for automatic validation of images regarding the presence of objects of the class Qalidus. To do so, the selection of the best-trained object detector is necessary, as well as the creation of an image validation module that can compare the ninjas' answers with the results of the use of the selected model. The system inputs for this requirement correspond to the results obtained from the execution of the trained models in the testing dataset. With the obtained results from testing the different trained models, it is possible and necessary to select and apply metrics that can be used to compare the models. Using these metrics, it is then possible to select the best model that should be used to create the image validation module. This model should be able to receive both the ninjas' answers and the results of the execution of the object detector selected on the considered images and validate the given answers considering both answers. This particular requirement is considered accomplished when the best object detection model has been selected. And, the selected object detection model is used together with the ninjas' answers to validate the received answers and has been locally tested.
Associated Tasks	T1. Selection of the metrics for the model analysis. T2. Analysis of the results of all the trained models using the selected detection metrics. T3. Selection of the best model according to the obtained results. T4. Insertion of the selected object detection model into the system. T5. Validation module creation, able to compare the ninjas' answers and the object model. T6. Locally test the functioning of the created system for automatic validation of images.

Table 4.3: The third requirement specification

Requirement ID	Req03
Title	The system should allow remote access.
Description	The system should allow remote access so that it can be used for validating images present in any place and can be accessed by anyone with allowed access. The inputs for this requirement are the object detection model selected that achieved the best results and the image validation module. Only with the creation of an image validation module, the system cannot be used by anyone else, except by who uses the personal computer used for locally creating and testing the image validation module created. To allow remote access, a hosting location for this created model and the addition of an API is necessary, as well as a validation that the system is accessible and can validate the images sent to it. This requirement has been accomplished when the system can be remotely accessed and can validate images regarding the presence of objects of the class Qalidus.
Associated Tasks	T1. Creation of an API endpoint for receiving images. T2. Analysis of the results of all the trained models using the selected detection metrics. T3. Creation of a validation dataset composed of at least a mission's images and the ninjas' answers. T4. Run the validation dataset through the API endpoint and store the results. T5. Compare the obtained results with the ninjas' answers and evaluate the capacity of the system. T6. Deploy the system in a virtual machine. T7. Start the API remotely. T8. Test if the system can be accessed and used from outside the virtual machine. T9. Send the created validation dataset to the deployed system and check if the results obtained are the same

4.2 Risk analysis

Creating a system involves risk. Considering this and the fact that the time to build such a system is limited, it was made a risk analysis. The objective of this analysis is to make sure all the requirements are fulfilled, and any problems that might arise have already been thought about and have a solution or a minimisation strategy associated with them.

The risk analysis performed resulted in four different risks, having been created a table for each one that can be seen next, being this table composed by a description of the risk and the mitigation strategy for that particular risk.

Table 4.4: The first risk specification

Risk id	Risk01
Title	The time needed to train all the models is not enough
Description	The training and preparation of object detection models takes a long time. Therefore, in the case that it is verified that the number of object detection models selected with the state of the art analysis is excessive, it means that this risk has occurred.
Mitigation strategy	If the time needed to train all the selected models, the best mitigation strategy corresponds to a second selection of the models that can still be trained and tested in the remaining time. If no model can be trained with the time remaining, the selection of the best model should only consider the trained models until that point, and it should be expressed in the dissertation the occurrence of this same risk and the mitigation implemented.

Table 4.5: The second risk specification

Risk id	Risk02
Title	No trained object detection model is considered good enough to be used for the creation of the objective system.
Description	There is the possibility that none of the trained object detection models is considered good enough for identifying objects of the class Qalidus during the analysis of the object detector on the testing dataset.
Mitigation strategy	If this risk occurs, there are two options that depend on the time remaining. The first one corresponds to the retraining of the model that achieved the best results, or the selection of another object detector to be trained and testes, where probably retraining and tweaking of the training parameters would work better than selecting a new model. The second option corresponds to the case when there is no more time available, and in this case, the best option corresponds to the writing of the dissertation describing the work done and explaining the several possibilities that led to the trained models' inability to be used.

Table 4.6: The third risk specification

Risk id	Risk03
Title	No virtual machine is provided for the model deployment.
Description	There is the possibility that RedLight cannot provide a virtual machine, preventing the system from being deployed and the completion of the third requirement.
Mitigation strategy	If this risk occurs, the best option to mitigate it is to create the same system, validate it and do all the possible steps except the deployment and carefully explain what happened and the created system that was not deployed but could be used on a personal computer.

Table 4.7: The fourth risk specification

Risk id	Risk04
Title	Progress loss due to a system malfunction.
Description	A risk inherent to the creation of all systems is the loss of progress made by forgetting to save those progress or by occurring any type of system malfunction that causes this loss.
Mitigation strategy	The best option to mitigate this risk is to prevent it by using a tool like GitHub or GitLab to store the progress, have rollback points, and restore lost code if necessary.

Chapter 5

Experimentation

This chapter describes how the experimentation for the construction of the objective system was done, from the creation of the training and testing dataset to the results obtained with the analysis of all the selected models after their respective training and consecutive test on the testing dataset that resulted in the selection of the best model.

The four selected models that were trained and tested and their results and training procedure are described in this section. A Haar feature-based cascade classifier that corresponds to a detector that uses haar-like features, and for this experimentation it was used the version of this classifier on OpenCV[1]. OpenCV is an open-source library for CV, machine learning and image processing. A classifier that uses HOG features was also selected, initially like the Haar feature-based cascade classifier, it was selected an OpenCV version. However, this version on OpenCV was discontinued because it was not according to the paper implementations of the HOG features, so a different version was used, an open version available on GitHub[2], that was used for person detection. Both the Haar feature-based cascade classifier and the selected HOG classifier are models that use fixed features and are simpler models, so in this case, they were used to verify if these more straightforward implementations could be used to solve this particular problem.

The other two selected models were the YOLOv4 and the EfficientDet. The YOLOv4 object detector was chosen because it is the current state of the art one-stage detector. Furthermore, its appliance of previous network implementations together with the use of both the BoF and the BoS allow this detector to achieve great accuracy and enables him to be used for real-time detection. The EfficientDet object detector is the current state of the art two-stage detector that offers solutions for multiple problems. The EfficientDet is not just an object detector but a family of objects detector. The detectors EfficientDet-D0 until the EfficientDet-D3 from the EfficientDet family are considered real-time object detectors, where the EfficientDet-D3 detector is the one with the best accuracy among these four. The EfficientDet detectors from EfficientDet-D4 until the EfficientDet-D7 can not perform as real-time object detectors but have an improved accuracy at the cost of much more FLOPs for the parameter calculation. Going from 55.2 billion FLOPs and 20.7 million parameters for the EfficientDet-D4 model to 410 billion FLOPs and 77 million parameters for the EfficientDet-D7 model. Considering this and the available resources, the selected model for testing the EfficientDet family detector and understanding of how it compares to the other selected detectors was the EfficientDet-D3 detector.

[1]`https://opencv.org/`

[2]`https://github.com/SamPlvs/Object-detection-via-HOG-SVM`

5.1 Experimental setup

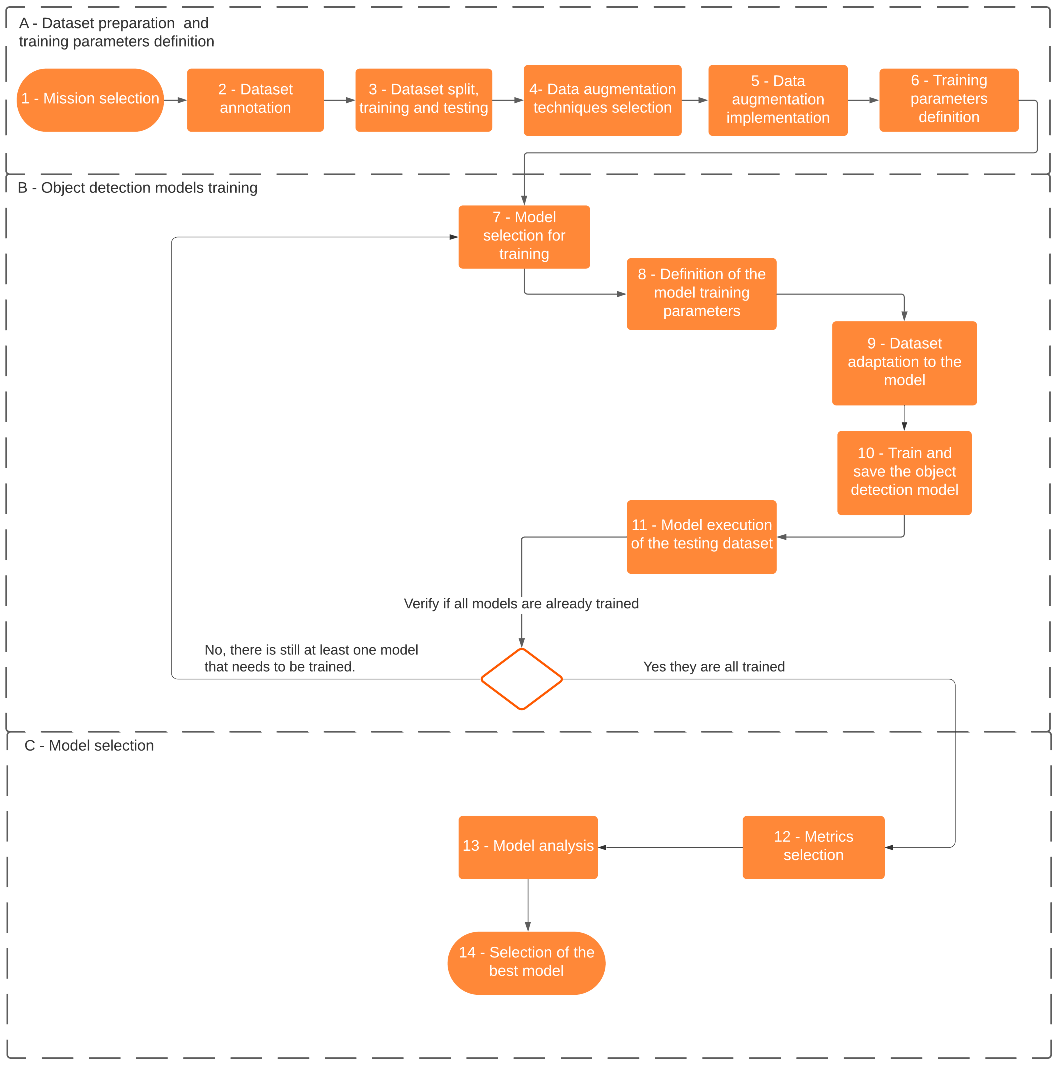
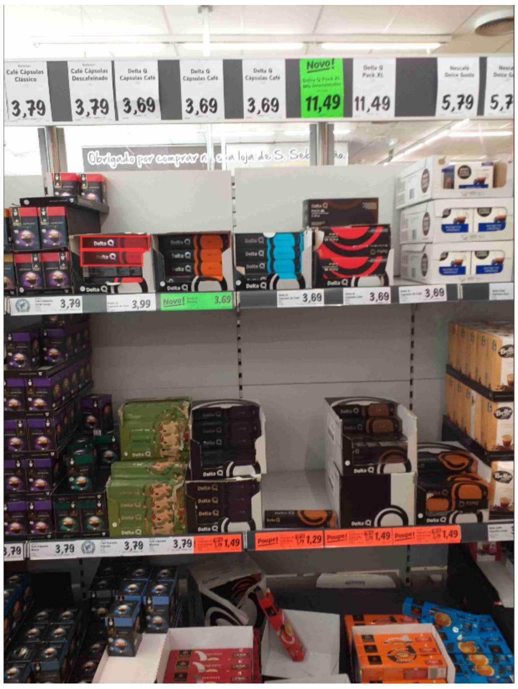

A - Dataset preparation and training parameters definition
1 - Mission selection
2 - Dataset annotation
3 - Dataset split, training and testing
4- Data augmentation techniques selection
5 - Data augmentation implementation
6 - Training parameters definition
B - Object detection models training
7 - Model selection for training
8 - Definition of the model training parameters
9 - Dataset adaptation to the model
10 - Train and save the object detection model
11 - Model execution of the testing dataset
Verify if all models are already trained
No, there is still at least one model that needs to be trained.
Yes they are all trained
C - Model selection
12 - Metrics selection
13 - Model analysis
14 - Selection of the best model

Figure 5.1: Training flowchart. The training is divided into three different sections. The first section(dashed square A) corresponds to the dataset preparation and the training parameters definition. The second section(dashed square B) corresponds to the object detection models training. Finally, the third section(dashed square C) corresponds to the selection of the best model.

For a better understanding of how the preparation, training and selection of the best model went, a flowchart was created, which can be seen in figure 5.1. This flowchart is divided into three sections corresponding to the different parts necessary to train and select the best object detection model. The first section, identified in figure 5.1 by the dashed square region A, corresponds to the creation and preparation of the dataset from the initial selection of the missions that should be used to create the dataset to the definition of the common training parameters. The description of the work performed according to this first section can be seen in section 5.1.1. The second section, identified in figure 5.1 by the dashed square region B, corresponds to the object detection models' training and is a loop since it is the training procedure used for all the models used in the experimentation. The description of the work performed according to this second section can be seen in section 5.1.2. The third and last section, identified in figure 5.1 by the dashed square region C, corresponds to the analysis of all the trained object detection models on the testing dataset and the resulting selection of the object detection model to be used for the systems' construction. The description of the work performed according to this third section can be seen in section 5.2.

5.1.1 Dataset preparation

The first step of the dataset preparation is the selection of the images that will be used to construct it. This dataset was composed of four distinct missions provided by RedLight. Three of these missions were missions where the objective of the ninjas was to verify the presence of objects of the class Qalidus in the supermarket shelves having images that were both negative and positive. Negative images correspond to images where no object of the class Qalidus is present in them, and an example can be seen in figure 5.2b. Positive images are images that have at least an object of the class Qalidus in them. An example of a positive image can be seen in figure 5.2a. The other selected mission had only negative pictures since it was not a mission where the objective was to verify the presence of objects of the class Qalidus.


(a) Example of a positive image that as at least one object of the class Qalidus. For a better understanding of the location of the objects of the class Qalidus, the image has been annotated, and the red bounding boxes correspond to the objects of the class Qalidus.

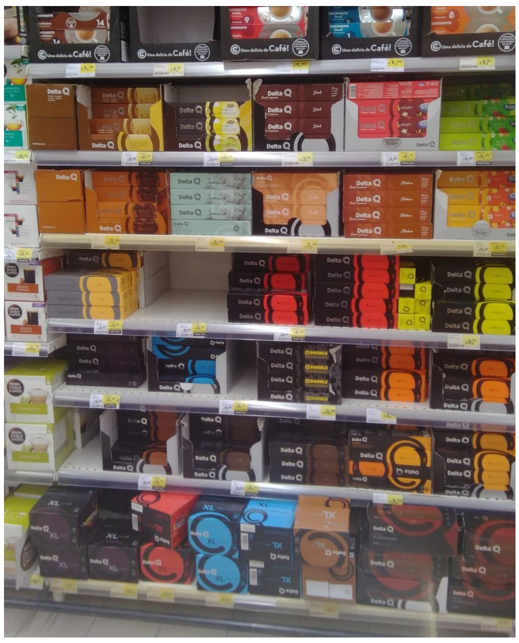

(b) Example of a negative image that has no object of the Qalidus class present in it.

Figure 5.2: Examples of positive and negative images, the figure 5.2a corresponds to an example of a positive image, and the figure 5.2b corresponds to an example of a negative image.

The second step after the creation of the dataset corresponds to the annotation of the images present in the dataset. The annotation tool selected for this annotation was *Make*

Sense[3], an open-source annotation tool. With *Make Sense*, the annotations were stored in the YOLO format.

In total, the created dataset was composed of 1485 images. Moreover, to precisely understand how was the dataset constituted of negative and positive images present on the dataset,it was made an analysis of the negative and positive images present on the dataset, which can be seen in table 5.1. From this analysis, it was possible to understand that 1126 of the 1485 images were negative, corresponding to about 75.83% of all the dataset images, and 359 were positive, 23.23%. On the other hand, from the positive images, 14 had only one object of the class Qalidus, and 345 had more than one object of the class Qalidus. Therefore, besides the number of positive and negatives, it was also analysed the number of objects that composed the dataset, consisting of 1715 objects of the class Qalidus, with an average of 4.78 objects of the class Qalidus per positive image.

Table 5.1: The number of positive and negative images in the complete dataset

	Number of images	Percentage
Negative images	1126	75.83%
Positive images with only 1 Qalidus object	14	0.94%
Positive images with more than 1 Qalidus object	345	23.23%
Total	1485	100%

After creating, annotating, and analysing the dataset, the next task was to split the dataset into both the training and testing datasets. Having been selected three missions for the training dataset and the one for testing.

The number of images and the total number of positive and negative images of the training dataset can be seen in table 5.2. The total number of images of the training dataset was 803, from what 510 were negative images, about 63.51% of the dataset, and 293 were positive images, about 36.49%. From the positive images, only 9 had only one object of the class Qalidus, and 284 had more than one object of the class Qalidus. Thus, in relation to the number of objects in the training dataset, 1511 objects of the class Qalidus were present, with an average of 5.16 objects of the class Qalidus per positive image.

Table 5.2: The number of positive and negative images in the training dataset

	Number of images	Percentage
Negative images	510	63.51%
Positive images with only 1 Qalidus object	9	1.12%
Positive images with more than 1 Qalidus object	284	35.37%
Total	803	100%

The number of images and the total number of positive and negative images can be seen in table 5.3. The total number of images in the testing dataset was 682, from which 616 were negative images, about 90.32% of the images in the testing dataset, and 66 were positive images, about 9.67% of the total images in the testing dataset. There were also 204 objects of the class Qalidus in the testing dataset, with an average of 3.09 objects of the class Qalidus per positive image.

As shown in the flowchart that can be seen in figure 5.1, the fourth step is the implementation of data augmentation in the training dataset. This step was introduced after verifying that the YOLOv4 detector uses data augmentation with good results, which is

[3]`https://www.makesense.ai/`

Table 5.3: The number of positive and negative images in the testing dataset

	Number of images	Percentage
Negative images	616	90.33%
Positive images with only 1 Qalidus object	5	0.73%
Positive images with more than 1 Qalidus object	61	8.94%
Total	682	100%

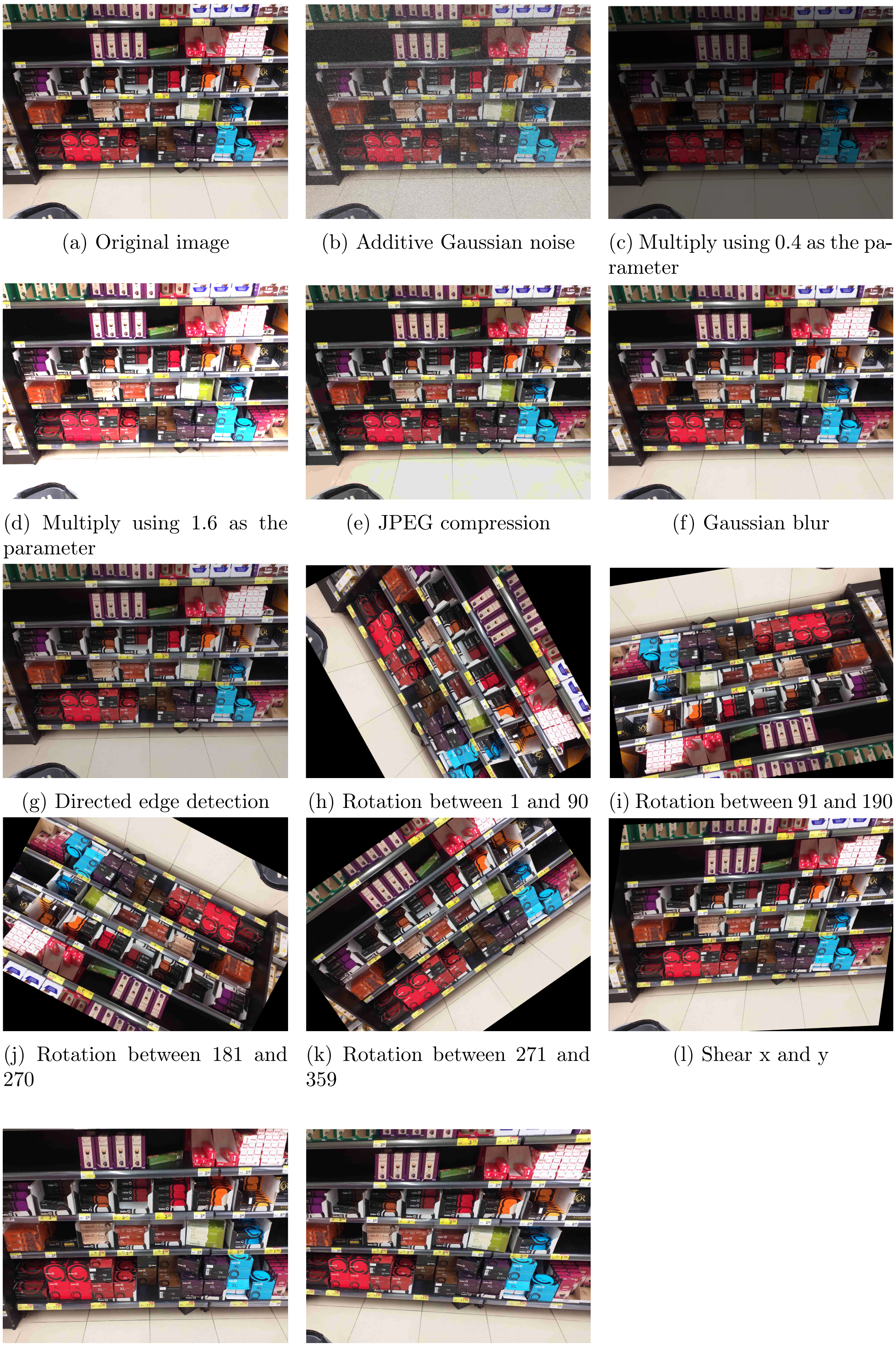
applied in general. Hence, the objective was to verify if the use of data augmentation created explicitly for the way the pictures are taken and the conditions that might appear in a supermarket would result in an object detector that would have better results than the provided version of the YOLOv4 model. Therefore, this fourth step corresponded to the selection of the data augmentation techniques. An example of the data augmentation techniques selected can be seen in Figure 5.3.

In the fifth step, several data augmentation techniques were implemented to the training dataset. Thus, twelve new images were created using a different data augmentation technique for each image in the training dataset. An example of the appliance of these data augmentation techniques to an image of the training dataset can be seen in Figure 5.3. Figure 5.3a shows the original image, and the remaining are the results of the appliance of these data augmentation techniques. To do the appliance of these techniques, it was used the imgaug python library[4].

The selected data augmentation techniques can be divided into two groups, the lightning and noise image transformations and the perspective transformations. The lighting and noise transformations have the objective of dealing with the fact that the pictures taken to the shelves can have multiple types of lightning. The camera of the phone that took these pictures can have numerous types of quality. To apply these transformations were used several augmenters. The first augmenter used was the additive Gaussian noise that corresponds to adding noise to an image. The added noise is sampled once per pixel of the image from a normal distribution N(0, s), where s is sampled once per image and varies between 0 and 0.4*255. An example of the result can be seen in figure 5.3b. The second augmenter used was the multiply augmenter, where all pixels in an image are multiplied with a specific value, thereby making the image darker or brighter. In this case, the value used to multiply the image for is selected once per image between 0.4 and 1.6, being the extremes 0.4, an image much darker like the one in the figure 5.3c, and 1.6, an image much lighter like the one in the figure 5.3d.

The third augmenter from the lightning and noise image transformations group used was the JPEG compression augmenter. This augmenter corresponds to the degradation of the quality of the photos JPEG-compressing them, simulating images taken with phone cameras of really bad quality, using a compression between 70 to 95%, an example of the result can be seen in figure 5.3e. The fourth augmenter used regarding lighting and noise transformations was the Gaussian blur augmenter that blurs images using Gaussian kernels, which as the objective of simulating low-quality cameras like the JPEG compression augmenter, an example of the result can be seen in the figure 5.3f. The fifth and last augmenter regarding lighting and noise transformations was the directed edge detection, which corresponds to the detection and marking of the edges in a black and white image that overlays the result with the original image. The black and white image overlays the initial image using an alpha value between 0 and 0.5 calculated for each image. An example of the result of using this augmenter can be seen in figure 5.3g.

[4]`https://imgaug.readthedocs.io/en/latest/#`



(a) Original image

(b) Additive Gaussian noise

(c) Multiply using 0.4 as the parameter

(d) Multiply using 1.6 as the parameter

(e) JPEG compression

(f) Gaussian blur

(g) Directed edge detection

(h) Rotation between 1 and 90

(i) Rotation between 91 and 190

(j) Rotation between 181 and 270

(k) Rotation between 271 and 359

(l) Shear x and y

(m) Scale x and y

(n) Perspective transform

Figure 5.3: Examples of the appliance of the considered data augmentation techniques to the original image(Figure 5.3a)

Besides the lighting and noise transformations, perspective transformations were also used to simulate pictures taken by different angles, an error identified by the work done in the first semester. The first augmenter used to apply these transformations was the affine rotation augmenter. This augmenter rotates an image by a random value between two defined limits, and it was applied four times with different values used as limits. The objective of using this augmenter was to simulate the different types of ways a ninja can use its phone to take a picture. The first rotation rotates the image between 1 and 90 degrees in the clockwise direction. An example can be seen in 5.3h. The second rotation rotates the image between 91 and 180 degrees in the clockwise direction, and an example can be seen in figure 5.3i. The third rotation rotates the image between 181 and 270 degrees in the clockwise direction. An example can be seen in figure 5.3j. Moreover, the fourth and last rotation rotates the image between 271 and 359 degrees in the clockwise direction, and an example is represented in figure 5.3k.

The second augmenter used from the perspective transformations group was the shear augmenter, which shears images along a chosen axis. For the considered dataset, a shear was first applied along the x-axis and then a shear along the y-axis. For both shears, the limits correspond to a value selected for each image between -20 pixels and 20 pixels. An example of the appliance of this augmenter can be seen in figure 5.3l. The third augmenter used from this transformation group was the scale augmenter, which scales images along a chosen axis. As selected for the shear augmenter, the scaling was applied for both the x and y-axis, with limits for both axes being between -20 pixels and 20 pixels. This value was chosen for each image when applying the augmentation. An example of the result of applying this augmenter can be seen in figure 5.3m. The fourth and last augmenter applied was the perspective transformation augmenter. This augmenter applies perspective transformations using a random scale. The scaling limiting values used were 0.01 and 0.15, selected per image, where the scale corresponds to a measure of how far the perspective transformation's corner points are from the image's corner points. An example of the result can be seen in figure 5.3n.

Using the data augmentation techniques described above and ensuring the integrity of the annotations made, the training dataset is finalised, just like the testing dataset. With this, the training dataset comprises 10439 images, from which 6630 are positive images and 3809 are negative images, as shown in table 5.4.

Table 5.4: The number of positive and negative images in the training dataset with the use of data augmentation.

	Number of images	Percentage
Negative images	6630	63.51%
Positive images with only 1 Qalidus object	117	1.12%
Positive images with more than 1 Qalidus object	6513	35.37%
Total	10439	100%

Before the training of the models selected by the state of the art analysis, the last step was the definition of the training parameters common to all models. All the trained models' height and width were defined as 416 for both the height and width. The threshold for what is considered a correct detection during training was defined to 0.3. The batch size corresponds to how many images from the training dataset are passed to the model at each training iteration.

5.1.2 Models Training

After preparing the dataset and the definition of the training parameters, the second part of the experimentation is represented in figure 5.1 by the B dashed square region. This second part corresponds to the training of the object detection models. Thus, the illustrated steps correspond to the training and testing of the object detectors selected and make up a circular flow that is only interrupted when the last model is already trained and executed on the testing dataset.

The steps referents to the training and testing of each object detection model illustrated in figure 5.1 were performed for each model, several times if needed. The first step(step 7 of figure 5.1) corresponds to the selection of the model to train from the models that still are not trained. With the selection of the model, the next step(step 8 of figure 5.1) is the definition of the training parameters specific to this particular model. In the third step(step 9 of figure 5.1), the dataset is adapted to the specificities of the model. This adaptation can be something like, do all the images in the dataset need to be of the same size, or does the model resize the images during training? Some models require the resize to be done previously, and others do not need that. These adaptations are made in this particular step. The fourth step(step 10 of figure 5.1) corresponds to the training and saving of the object detection model. This step takes much time, lasting as long as weeks for some models. The final step(step 11 of figure 5.1) corresponds to the execution of the trained model on the testing dataset, storing the detected bounding boxes in a way that makes them easily accessible when comparing all the models. Finally, as represented in figure 5.1 by the decision box, after all the models' training and testing, it was verified if all the models were already trained. If they are not, then a new model was selected that needed to be trained, and the process was replicated for this new model. If they were all trained, then the second part of the experimentation is done, and it was possible to start analysing all the trained models and the obtained results.

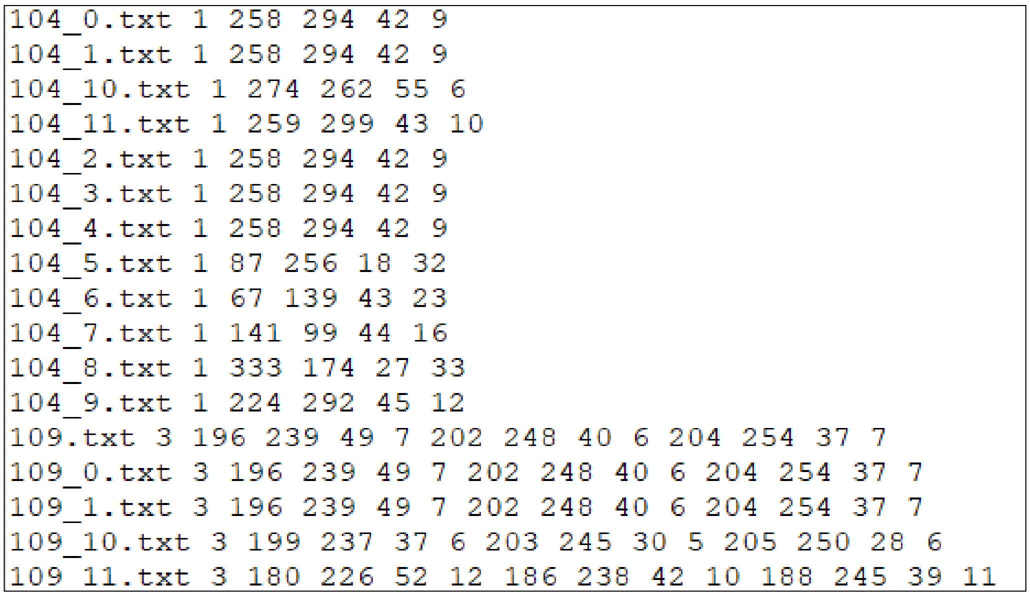
Haar feature-based cascade classifier

The first object detector selected was the Haar feature-based cascade classifier. When training a Haar feature-based cascade classifier, the first step is to resize and grey out both the positive and negative images in the training dataset to a specific size. Since contrarily to other object detectors, this step is not automatic. However, this size is equal for all object detectors used so that the object detectors can be fairly compared. The value defined was 416 pixels for both the width and the height. The resize of already annotated images needs to consider the format of the annotations, which in this case was the YOLO format. To resize these images and respective annotations, the imgaug resize augmenter was used, which uses a different annotation format. First, the annotations need to be changed to the imgaug annotation format, and then it is performed the resize of both the images and their annotations. Finally, with the image resized and the annotations updated, it is needed to save the annotations in the YOLO format again, converting them to the correct format.

With the images resized and their annotations adjusted, the next step was to define the size of the sliding window by calculating the average height and width of all the annotated objects. The annotated objects' resulting average values were 43 pixels for the width and 15 pixels for the height.

After the definition of the sliding window size, the next step was to separate the resulting training dataset into two different folders. One for all the positive images named "positives" and another for all the negative images named "negatives".

With both folders created, a file was needed to describe each folder's contents, each image's name and the respective location of the annotated objects in that image. The name of this file in the folder with the positive images was info.dat. This file has the information about all the locations of the bounding boxes for all the positive pictures. Each line of the info.dat file represents the location of the bounding boxes for the positive image described in that line. An example of the format of this file can be seen in figure 5.4. The structure of each line of this file is described as follows. First, it starts with the name of the positive image, followed by a space. Then it is the number of bounding boxes present in that image. After that, followed by a space again and then for each bounding box, the x value of the top left corner, the y value of the top left corner, the x value of the bottom right corner and the y value of the bottom right corner all separated by a space, with a line break at the end.



```
104_0.txt 1 258 294 42 9
104_1.txt 1 258 294 42 9
104_10.txt 1 274 262 55 6
104_11.txt 1 259 299 43 10
104_2.txt 1 258 294 42 9
104_3.txt 1 258 294 42 9
104_4.txt 1 258 294 42 9
104_5.txt 1 87 256 18 32
104_6.txt 1 67 139 43 23
104_7.txt 1 141 99 44 16
104_8.txt 1 333 174 27 33
104_9.txt 1 224 292 45 12
109.txt 3 196 239 49 7 202 248 40 6 204 254 37 7
109_0.txt 3 196 239 49 7 202 248 40 6 204 254 37 7
109_1.txt 3 196 239 49 7 202 248 40 6 204 254 37 7
109_10.txt 3 199 237 37 6 203 245 30 5 205 250 28 6
109_11.txt 3 180 226 52 12 186 238 42 10 188 245 39 11
```

Figure 5.4: Example of the format of the info.dat file for the positive images

It was also needed a file to describe the contents in the folder for the negative images. Therefore, this file in the negative images folder was also named info.dat. The format of this file is more straightforward than the one for the positive images since it only requires the name of each negative image, all separated by a line break or all in a different line in the file.

The next step after the creation of the info.dat file for both the positive and negative images folders was to create the vector that represents the objects of the class Qalidus described in the info.dat file of the positive images folder. For that, the executable opencv_createsamples from OpenCV[5] was used with the parameters maxxangle, maxyangle and maxzangle set to 1.0 to eliminate all rotations and data augmentation techniques used over the objects already annotated and augmented. The parameter num was also defined as the number of positive images on the training dataset.

With all the files needed for training created, the next step was to start training the classifier. For that, the OpenCV executable opencv_traincascade was required, along with the definition of its parameters. The size of the sliding window that was calculated above, being 43 for the width(the parameter for representing the width is the w) and 15 for the height(the parameter for representing the height is the h). The number of positive images

[5]https://opencv.org/

using the numPos parameter allows to define how many positive images are selected from all the positive images in the training dataset and should be used at each training stage. The used value, in this case, was 3200 that corresponds to about half of the number of positive training images. Second, the number of negative images is defined using the numNeg parameter. This parameter indicates how many negative images are selected from all the possible negative images at each training stage. According to the OpenCV documentation and discussions, this value should be about 1.5 times greater than the number of positives parameter, having been defined as 4600. The other parameters that were defined without being used its default value were the precalcBufSize and precalcIdxBufSize set to 1024 to define the amount of RAM available for calculations, the maxFalseAlarmRate parameter that was defined to 0.05 and corresponds to the percentage of negative samples allowed to be incorrectly classified as positive. Furthermore, with these parameters specified when executing the opencv_traincascade executable, the only parameter left to define was the numstages that corresponds to the number of stages(number of weak classifiers) to perform until the training can be considered done. The numstages parameter was initially set to 10 at the beginning of the training.

The size of both the images and the sliding window influence the training time. In this case, the used values were considered large values contrarily to those used by the Viola-Jones object detectors[29], resulting in a long training time. The testing dataset was analysed at the end of the tenth training stage, having been trained ten weak classifiers to create a more robust classifier. The results of this analysis were stored for later analysis and comparison with other models. After this, some images were selected to understand if the detector was able to detect objects of the Qalidus class. With this, it was understood that the created detector had not found any correct object of the Qalidus class in the selected images. So, the training for more stages continued, saving three more detectors at three different stages, one at the 12 stages, another at 15 stages and finally one at 16 stages. Each saved detector was executed on the testing dataset, and the results were also saved for later analysis.

Object detector using HOG features and an SVM classifier

After the Haar feature-based cascade classifier training and saving at different stages, the next object detector selected for training was an object detector that used HOG features and an SVM classifier[6]. Like the Haar features, the HOG features can only be calculated on a colour channel, so the images also need to be greyed out. To speed up the training process and considering that for the previous model, the training dataset had already been greyed out and divided into positive and negative images, this adaptation of the training dataset was also used for the training of this classifier.

So, the next step was to cut out the positive images' objects and adjust them to the size of the sliding window. In this case, like previously, the sliding window was defined as the average of all the objects in the training dataset, being 43 for the width and 15 for the height. To do the cutout, the function asarray from the python NumPy library[7] was used. The resulting images were then adjusted to the specified size for the sliding window and saved to a folder for the positive pictures named "positives".

For this particular use of the HOG features, the negative images also have to have the sliding window size. Considering that resizing the images to this size would not work and would produce erroneous results, the chosen solution was to split the images into smaller

[6]`https://github.com/SamPlvs/Object-detection-via-HOG-SVM`
[7]`https://numpy.org/`

pictures of the sliding window size. The resulting images were saved into a file named "negatives".

So after this, the next step was to extract the HOG features from both the positive and the negative images. With the features extracted, the training parameters needed to be defined. These parameters were the width of the sliding window that had already been defined as 43 pixels for the width and 15 pixels for the height. The threshold for what is considered a correct detection during training, which was in this case defined as 0.3. Moreover, the maximum number of iterations for the training of the SVM was adjusted to 10000. instead of the default 1000, considering the high number of images considered for training.

Three object detectors were trained using HOG features and an SVM classifier, where the only alteration between them was the number of randomly selected negative images. The first one was trained with 18301 positive images and 331500 negative images, the second with 18301 positive images and 132600 negative images and the third and last one with 18301 positive images and 66300 negative images. These three versions were trained to understand the capabilities of the selected version of a classifier that uses HOG features and an SVM classifier.

After training these three versions, each one was executed on the testing dataset, and the results were stored for later analysis and comparison with the other trained models.

EfficientDet object detector

So far, two simpler models were trained from the era described by Zhengxia et al.[31] as the traditional object detection era. A Haar feature-based cascade classifier object detector(section 5.1.2) and an object detector that implemented HOG features, and an SVM classifier(section 5.1.2). The next trained object detector was a two-stage object detector from the EfficientDet family of object detectors. Considering the available resources and the time available for training, the model selected from this family of object detectors was the EfficientDet-D3 model.

The EfficientDet-D3 model, contrarily to the models trained so far, can not be trained on windows, so the solution found was to use the Colaboratory[8]. Colaboratory is a free Jupyter notebook environment created by Google that runs entirely in the cloud and allows free access to GPU. Access to GPU is essential for training the EfficientDet-D3 model and other recent object detectors like the YOLOv4.

Training an object detection model on Colaboratory requires access to a Google Drive so that the training images can be accessed and the trained models can be saved. However, considering the current size of the augmented trained dataset, 18 GB, and the limited space on Google Drive, the training dataset would take much time to load up to Google Drive and would not fit entirely. So the solution found in this case considering the fact that the detector would resize all the images to the previously defined size of 416 pixels for the width and 416 pixels for the height was to resize the images first and only then uploading them to Google Drive, allowing to considerably reduce the size of the dataset to about 4 GB. Besides reducing the total size of the training dataset uploaded, it also significantly reduced the uploading time of the dataset.

With the training dataset on Google Drive, it was also needed to upload the annotations in the correct format. The EfficientDet family of detectors uses the annotations on the

[8]`https://colab.research.google.com/`

COCO format, contrarily to the annotation format used that was, the YOLO format. The YOLO format uses a file for each image to describe the objects present in that particular image, contrarily to the COCO format, where only a JavaScript Object Notation (JSON) file in a specific format describes all the objects present in all the images. To do the conversion from the YOLO format to the COCO format, a Git repository named "Yolo-to-COCO-format-converter"[9] was used. To do the conversion, it was followed the procedure described in this Git repository. Resulting in a JSON file in the COCO format that was then uploaded to the same folder of the training dataset on Google Drive.

After having the training dataset and the respective file, with the annotated objects in the COCO format, the next step was to adapt the EfficientDet tutorial provided by the authors in their GitHub repository to run the EfficientDet-D3[10]. Nevertheless, first, it was necessary to access the files on Google Drive. To do so, it was mounted Google Drive on Colaboratory. After this, the parameters required to start training were defined.

The first parameter defined was the one named MODEL that allows to specify which of the EfficientDet family of object detectors was selected for training. In this case, the chosen model for training was the EfficientDet-D3 model, so this parameter was set to efficientdet-d3. The second parameter specified was the threshold to define what is considered a correct detection during training. The name of this parameter is min_score_thresh, and it was defined to 0.3 as previously described in the common parameters for all models. Then the height and width were defined, indicating the values that the training images should be resized to, something that had already been done before as described previously, but that needs to be specified for training, being the width and height defined as 416. Another parameter defined that was the same for all models was the batch size, being set to 64. The last parameter defined was the number of epochs. The number of epochs corresponds to the number of times the entire dataset is passed forward and backwards through the dataset, and this number was defined to 50, meaning that the dataset will be fully seen 50 times.

With the training, parameters defined, it was now just necessary to specify the location of the file of the resulting model to be saved on Google Drive. And with that, it was all set to start the training that took a long time. The loss function of the trained EfficientDet model can be seen in figure . At the end of the training, the model is automatically saved to the location specified.

So after the model training was over, the remaining task was to execute the trained model on the testing dataset. To do so was also used the Colaboratory and the results were also stored on Google Drive, having been download after, for further analysis and comparison.

YOLOv4 object detector

The last object detector trained was the YOLOv4 object detector, a one-stage object detector. For this particular model, two different versions of this same classifier were trained. One version used the online data augmentation implemented by the YOLOv4 model, and another used the training dataset with offline data augmentation. The reason that led to the training of two different versions of the same model came from the fact that the standard YOLOv4 object detector uses online data augmentation, implemented with the BoF, which would result in an unfair comparison with the rest of the models trained. So the solution found was two train two different versions. One with the standard YOLOv4

[9] `https://github.com/Taeyoung96/Yolo-to-COCO-format-converter`
[10] `https://github.com/google/automl/tree/master/efficientdet`

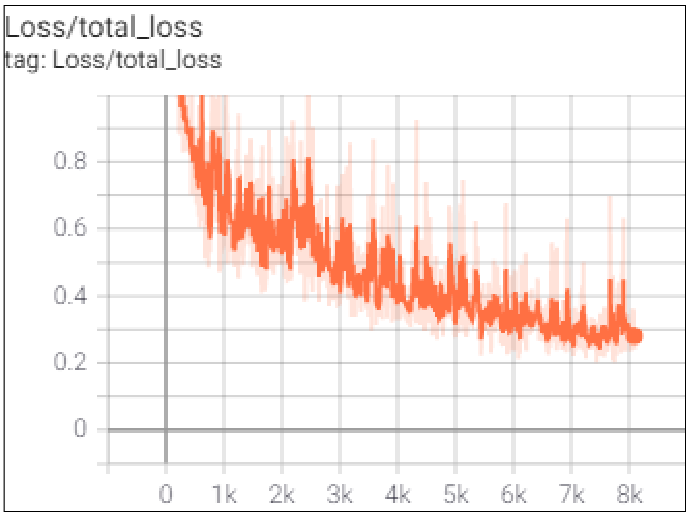
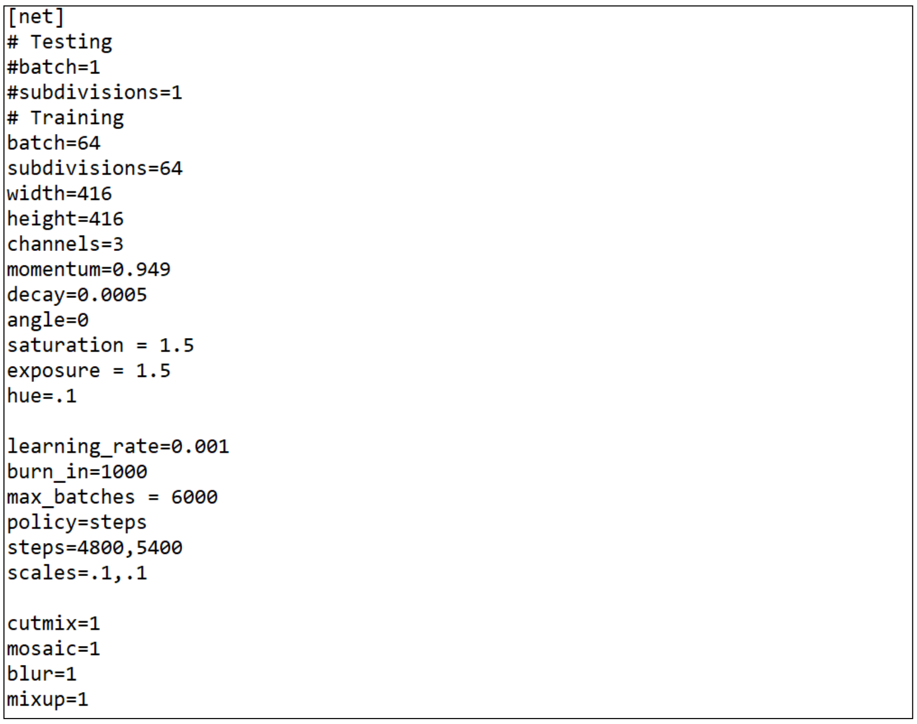


Figure 5.5: The loss function of the trained EfficientDet model

data augmentation activated and using the initial training dataset. The other one with the standard YOLOv4 data augmentation deactivated and using the training dataset with offline data augmentation.

The training dataset for both versions of the YOLO implemented was already prepared, as well as the annotations that were already in the correct format, the YOLO format. So the first model version trained was the version that used online data augmentation, and for this version, the next step was to define the training parameters on the YOLOv4 configuration file. The YOLO configuration file name is "yolov4-obj.cfg". The configuration file of the YOLOv4 object detector trained with online data augmentation can be seen in figure 5.6.



```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

cutmix=1
mosaic=1
blur=1
mixup=1
```

Figure 5.6: Configuration file of the YOLOv4 object detector trained with online data augmentation.

To define the training parameters, it was needed to edit the YOLO configuration file.

Table 5.5: The parameters defined for the YOLOv4 models trained

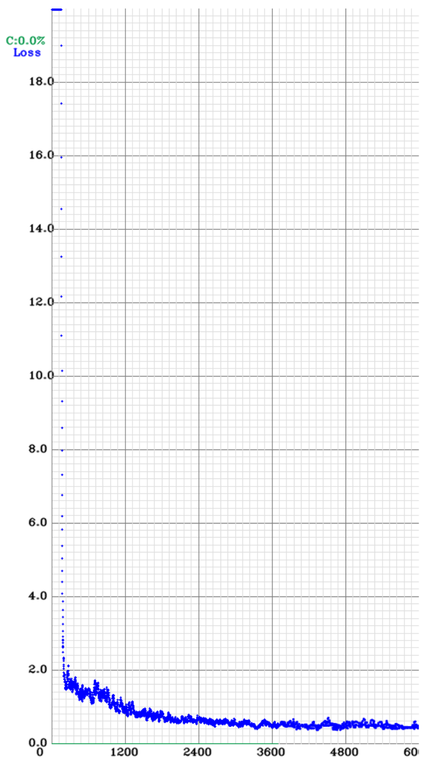
	Parameters defined
Batch size	64
Subdivisions	64
Width	416
Height	416
Classes	1
Filters	18
Max batches	6000
Steps	4800,5400

The edited parameters were the batch size that, just like for the EfficientDet model, was set to 64 and corresponds to pass 64 images at each training iteration. The subdivisions parameter was set to 64. The subdivisions parameter is used to know the number of images that are processed in parallel. The number of images processed in parallel is calculated by dividing the batch size by the subdivisions, resulting in this case in the value of one for the number of images processed in parallel. As defined for the previous detectors, the width and height parameters were set to 416 pixels for both parameters. The classes parameter was set to 1 and corresponds to the number of different classes. In this case, it was only the Qalidus class. The filters parameter was set to 18, and this value was calculated using the formula ((number of classes + 5) * 3). Since the number of classes is just one, the resulting value of using this formula is 18 as defined for the filters parameter. The filters correspond to the number of learned parameters by the YOLOv4 model, being usually randomly initialised, updated during training to minimise the loss, and fixed after the training has ended. The maximum number of batches was set to 6000 as recommended by the YOLOv4 specifications when having the BoF activated, and only one class is used. The last edited parameter was the steps, set to the value (4800,5400) that corresponds to 80 and 90% of the maximum number of batches, respectively. For an easier comprehension of all the configuration parameters, they are represented in table 5.5.

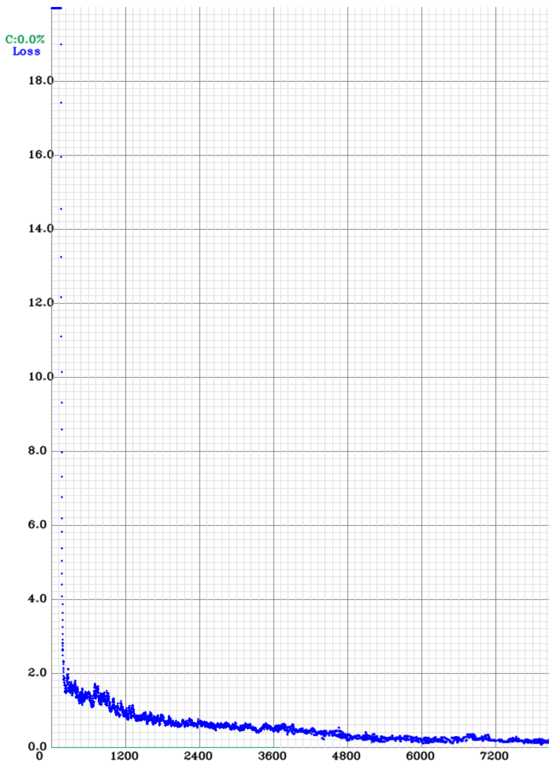
After finishing the edition of the YOLOv4 configuration file and corresponding parameter definition, the first version that used the standard YOLOv4 online data augmentation was ready to be trained. The training loss graph that resulted from the training of this YOLOv4 model was saved and can be seen in figure 5.7a. When the train ended, the model was stored and executed on the testing dataset, having the results been saved for further analysis like the other trained models.

The second version of the YOLOv4 object detector trained did not use the standard YOLOv4 online data augmentation, so the first step was to turn off the online data augmentation. To turn off the data augmentation techniques implied by the standard implementation of the YOLOv4, it is needed to change the YOLOv4 configuration file, defining the variables cutmix, mosaic, blur and mixup to 0 since these are the parameters that indicate when training the YOLOv4 classifier that it should be applied the data augmentation techniques present in the BoF. The resulting configuration file of the YOLOv4 object detector trained with the online data augmentation turned off can be seen in figure 5.8.

The parameters defined for the previous version of the YOLOv4 object detector were also used for this version, except for the maximum number of batches parameter. The maximum number of batches parameter was set to 8000, corresponding to about 50 epochs, just like the used for training the EfficientDet model. However, it were saved versions at the 6000,



(a) The training loss graph that displays the training iterations of the YOLOv4 object detector trained with online data augmentation for 6000 iterations.



(b) The training loss graph that displays the training iterations of the YOLOv4 object detector trained with offline data augmentation for 8000 iterations.

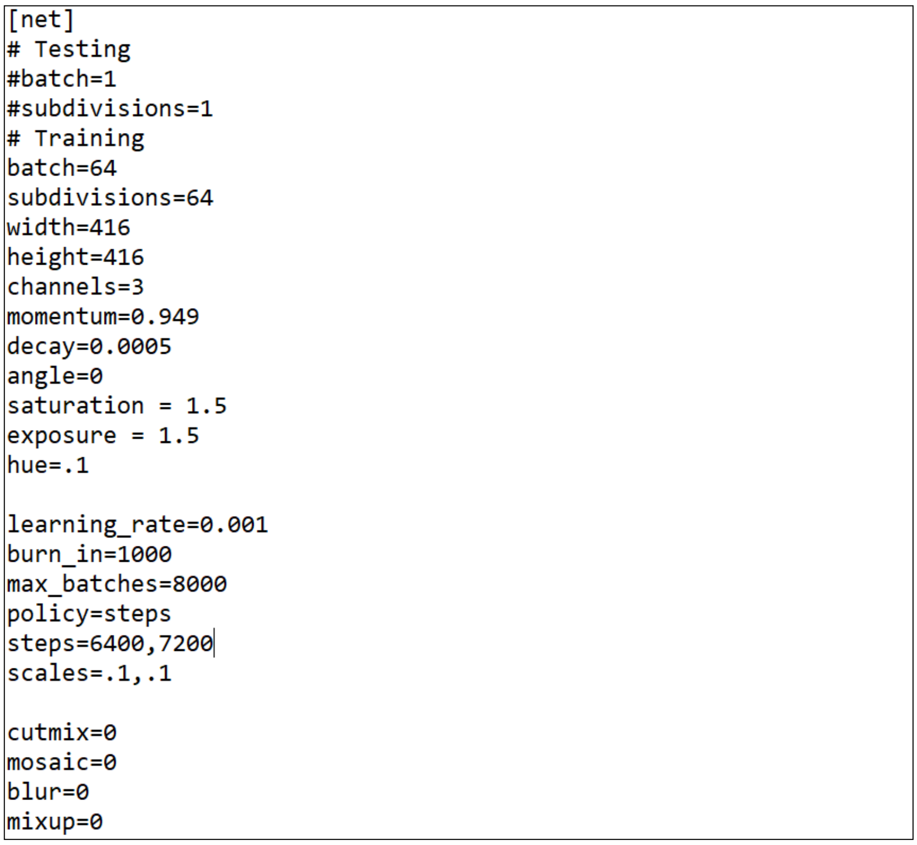
Figure 5.7: The training loss graph for the YOLOv4 models trained

7000 and 8000 iterations of this same model for analysis and fair comparison between the YOLOv4 trained versions and the other trained object detectors, particularly with the EfficientDet model that was trained for 50 epochs. After the alterations to the YOLOv4 configuration file, the model was trained, and versions of this same model were saved at 6000, 7000 and 8000 iterations. At the end of the training of all these versions, each one was executed on the testing dataset, and the results were saved for future analysis. The loss function graph for the YOLOv4 model trained with offline data augmentation for 8000 iterations can be seen in figure 5.7b. The loss for the 6000 and 7000 iterations saved models can be seen in this figure as well.

With all the object detectors trained, the second part of the experimentation is finished, so the next part corresponds to the analysis of the experimental results and selection of the best model that is going to be used for deployment. This third and final part of the experimentation is described in the next section.

5.2 Experimental results

This section describes the experimentation results, and the procedure taken is represented in figure 5.1 by the dashed square C. The first step corresponds to the selection of the metrics that should be used to compare the trained models. The second step corresponds to the analysis of the trained object detection models using the selected models. Finally, the last step corresponds to the final selection of the best model that should be used for



```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches=8000
policy=steps
steps=6400,7200
scales=.1,.1

cutmix=0
mosaic=0
blur=0
mixup=0
```

Figure 5.8: Configuration file of the YOLOv4 object detector trained with the online data augmentation turned off.

training.

The first step corresponded to the selection of the metrics used to compare the object detection models. As described in section 2.3.1, the metric selected for comparison and selection of the best model was the PASCALVOC mAP using the R. Padilla et al. GitHub repository[11]. In this case, considering that it was only a class, the mAP corresponds to the AP. The AP used considered an IoU threshold of 0.5.

The first model trained was the Haar feature-based cascade classifier, having been saved four different versions of this same classifier at different stages. So the first analysis made corresponded to the verification and selection of the best model only among all the versions of the Haar feature-based cascade classifier. To do so, it was created the precision*recall curve and AP for each model. The resulting precision*recall curve and AP of each model can be seen in figure 5.9. Contrarily to what was expected, the model with the best AP was the one with the lowest number of training stages, the model with 10 stages, having achieved an AP of 6.48%. The second-best model was the one that was compounded of 12 stages, with an AP of 5.07%. The other two versions of this classifier had similarly bad results of 1.30% AP for the classifier with 15 stages and 1.23% AP for the classifier with 16 stages.

With the results obtained from the training of the Haar feature-based cascade classifier and the selection of the model with 10 stages as the best Haar feature-based cascade classifier, the AP of this model indicates that if this was the best model from all the trained ones, it would not be able to be used for the deployment since it is well below the baseline of

[11]`https://github.com/rafaelpadilla/Object-Detection-Metrics`

50% for the AP, this baseline is also represented in figure 5.9. Therefore, these results show that a classifier like the Haar feature-based cascade classifier could not be used for the creation of the objective system.

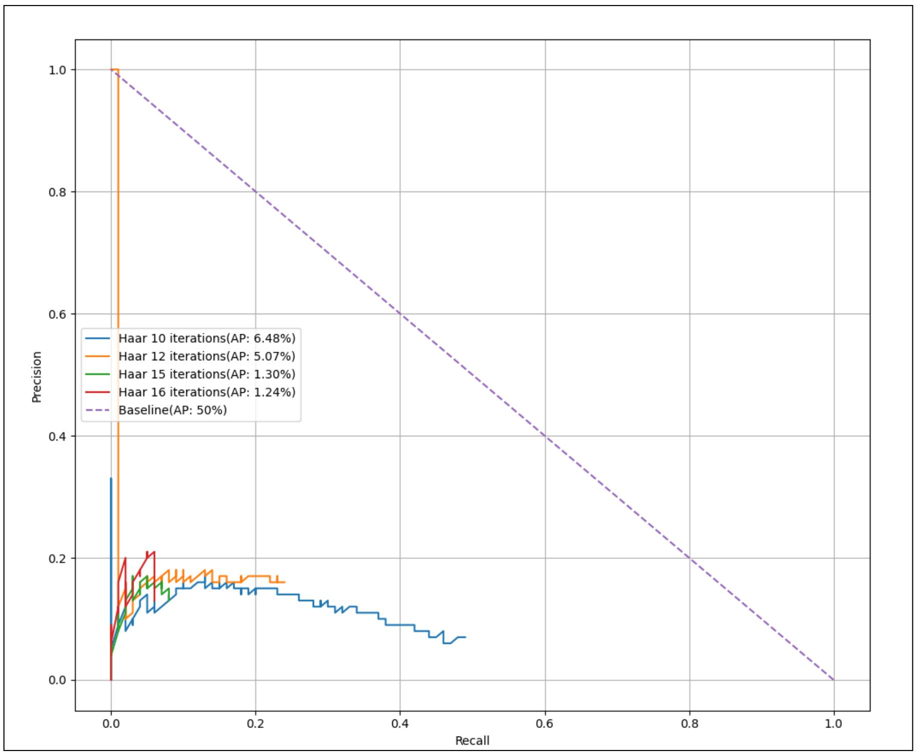


Figure 5.9: Results of the AP for the Haar feature-based cascade classifiers and the corresponding precision*recall curve

The second analysis made was regarding the second model trained, an object detector using HOG features and an SVM classifier. The results obtained from analysing the model precision and accuracy for different thresholds can be seen in figure 5.10, where the precision*recall curve and the AP of each model are plotted. Unfortunately, with these results, no trained model was selected for comparison with the remaining trained models since no object detector had even an AP bigger than one. These results were disappointing, and there are several causes for this. For example, a reason might be that the implementation used was incorrect or too simpler for the considered problem. Another reason might have been that some parameter essential was not tweaked enough or was not correctly set. Another possibility is that the HOG features can not be used to identify objects of the class Qalidus in these particular conditions. The reality is that this model was tried, and the results were awful. The causes are all suppositions, and in the end, this model did not even make it to the final analysis.

The third analysis made was the one regarding the utilisation of the implemented offline data augmentation and the standard YOLOv4 online data augmentation. The resulting precision*recall curve and the AP of each model can be seen in figure 5.11. From these results, it was possible to understand that for the same number of training iterations, 6000 iterations, the YOLOv4 model trained with the offline augmented training dataset and no online data augmentation was able to achieve almost five more per cent AP with a 75.19% than the YOLOv4 model trained with the original training dataset and using online data augmentation, that only achieved 70.31% AP. These results confirmed the supposition that the use of data augmentation carefully taught regarding the conditions of the received images allows to achieve better results than the ones obtained with the use of online data

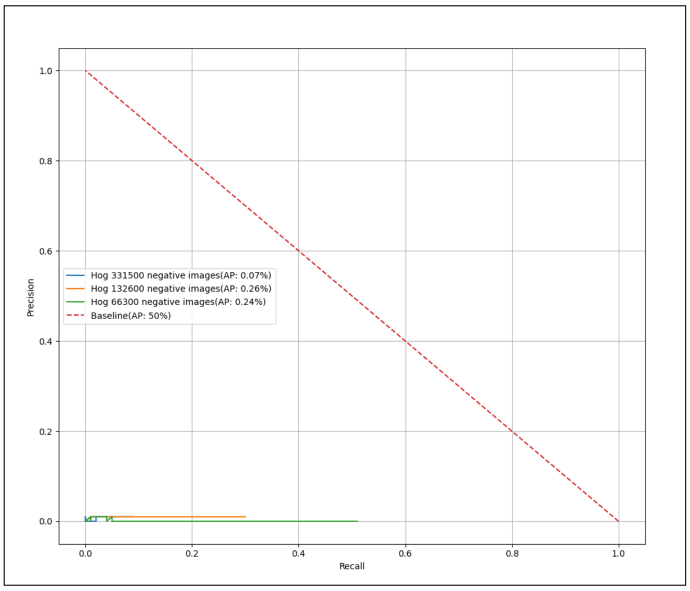


Figure 5.10: Results of the AP for the trained object detectors using HOG features and an SVM classifier with a different number of negative images and the corresponding precision*recall curve

augmentation by the standard YOLOv4 model.

The final analysis corresponds to the analysis made to select the model to be used for the deployment. The resulting precision*recall curve and the AP of each candidate model can be seen in figure 5.12. From this analysis, it is possible to verify that the worst model was the Haar feature-based cascade classifier with an AP of 6.48%. The second worst model from the candidate models was the EfficientDet model, having only achieved an AP of 42.05%, which is worst than the defined baseline. From the YOLOv4 trained models, the best one was the YOLOv4 model trained using data augmentation implemented on the training dataset and executed for 8000 iterations, having achieved an AP of 92.60%. The other two YOLOv4 versions trained performed better than both the Haar feature-based cascade classifier and the EfficientDet detector achieving more than 70% AP but being worst by more than 15% AP than the best YOLOv4 version.

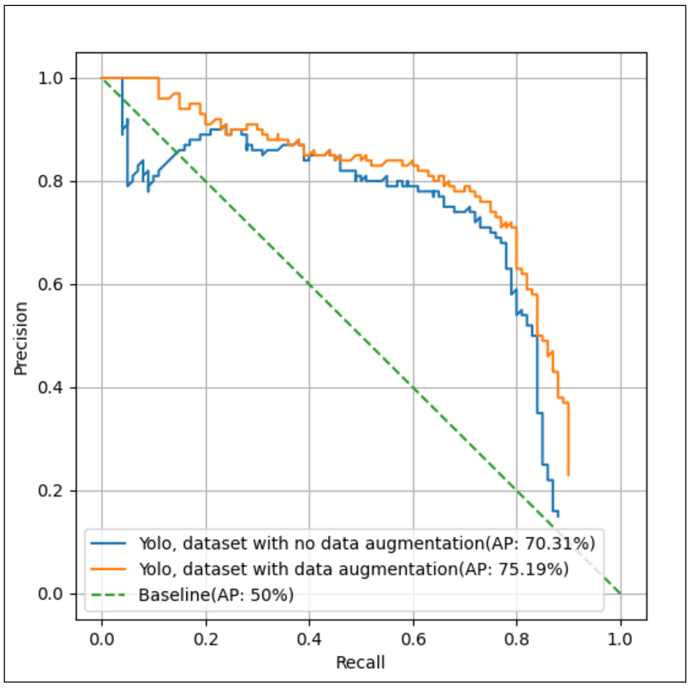


Figure 5.11: Results of the AP for comparison of the YOLOv4 model trained with the original training dataset(blue line, AP: 70.31%) and the YOLOv4 model trained with the augmented training dataset(orange line, AP: 75.19%)

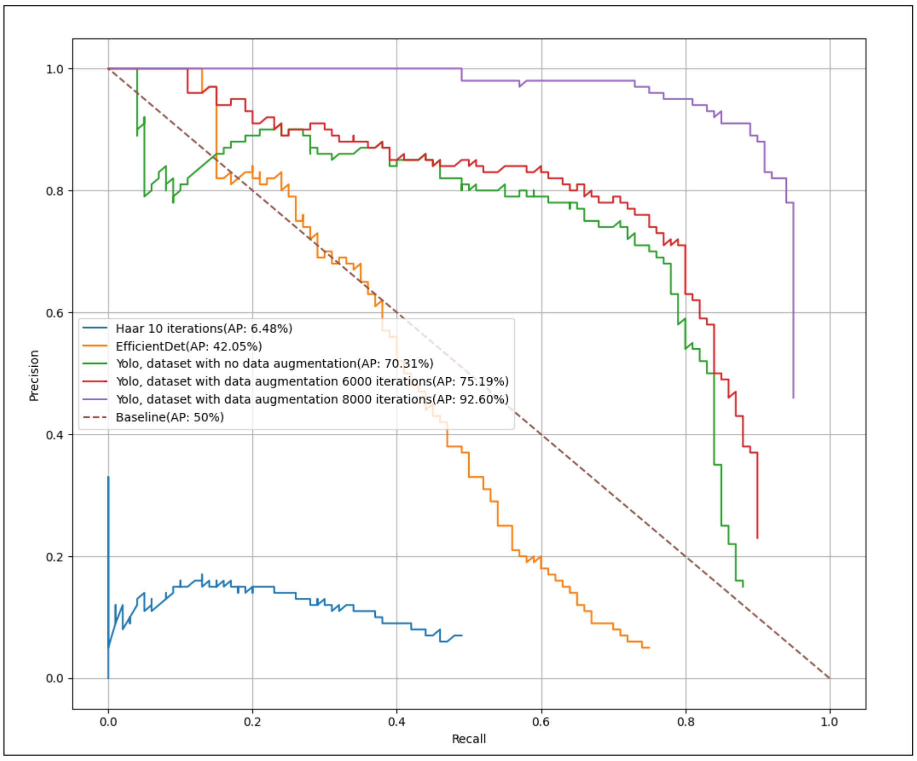


Figure 5.12: Results of the AP for all the best models and the corresponding precision*recall curve for all models

Chapter 6

Deploy and Integration

So far were described the several experimentation steps and the obtained results that led to the choice of the YOLOv4 model with the use of offline data augmentation on the training dataset trained for 8000 iterations, about 50 epochs, as the best model. Furthermore, this model was selected as the one that should be used for the next phase that involved the model validation and consequent integration of this model inside an Application Program Interface (API) to create a system that can ultimately solve the initially described problem. This next phase is described in this chapter.

The way the system should behave, from sending the images to validate to the return of the validation results, can be seen in figure 6.1. The first step corresponds to the aggregation of the images and responses given to be analysed in the correct format and sending them to be analysed by the system. The following step is the receiving of the images and information provided by the ninjas by the system' API endpoint. The third step corresponds to the execution of the object detector on the received images. With the execution of the object detector on the received images, the next step is the comparison of the object detectors result with the ninjas' answers provided, done automatically. The fifth step corresponds to the validation of the images provided and preparation of the results to be returned. Finally, the last step corresponds to the return of the validation results to the user that requested the validation.

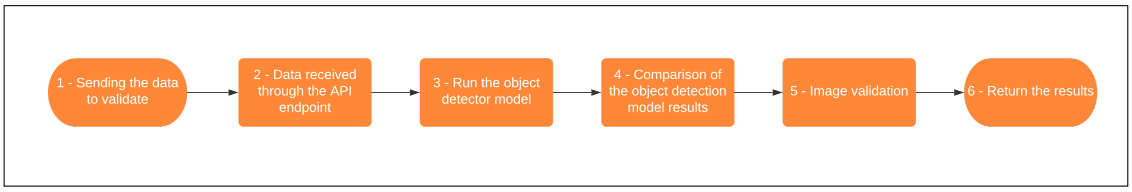

1 - Sending the data to validate
2 - Data received through the API endpoint
3 - Run the object detector model
4 - Comparison of the object detection model results
5 - Image validation
6 - Return the results

Figure 6.1: Representation of how the integrated system should function.

Before integrating all the model parts into one system, an API was created to fulfil the first two system steps represented in figure 6.1. It was used Django, which is a high-level Python web framework, to create this API. With this API, an endpoint was created that could be accessed from the exterior to remotely receive the information provided by the ninjas, both the answers relative to the presence of an image and the images that verified those answers. Before moving to the next step, it was locally tested the sending of example data in the correct format to the API, verifying if the received information was the same as the one sent.

The following section describes all the steps performed to integrate all the parts created and selected independently in a system for validation of the images regarding the presence of objects of the class Qalidus and the validation done to verify if the system could be used as part of the Brands&Ninjas platform.

6.1 Model integration

The integration of the model parts corresponds to the fulfilment of the first three steps, from the sending of the data, its receiving and analysis of the images regarding the presence of objects of the class Qalidus using the selected YOLOv4 object detection model. To do so, the YOLOv4 model was integrated into the API endpoint in a way that this trained model could be executed after the data was received. With this, the system was compound of two different parts the API, which receives the images, and the object detection model that is executed from the API.

With the integration made, the next part corresponded to the automatic comparison of the obtained information by the execution of the object detection model and the ninjas' answers. To make this automatic comparison, after the execution of the YOLOv4 model, the objects found for each image were compared with the ninjas' answers'. This comparison corresponded to the fourth step represented in figure 6.1 and resulted in the system's capacity to automatically compare the two obtained answers from the ninjas and the object detection model.

The fifth step corresponded to the identification of what was regarded as a validated and a non-validated image. There were four possible cases, and to each was it was associated a type if it was validated or not. If it was found at least one object of the class Qalidus and the ninja answer was that the Qalidus was present, the image was considered validated. As it happened when no object was found, and the ninja answer was that the Qalidus was not present. In the case that the object detection model found objects and the ninja answer was that the Qalidus was not present, the image was regarded as not being validated. When the model found no objects and the ninja answered that at least an object of the class Qalidus was present, the images were considered not validated.

After the construction of a way to automatically validate all the images, to fulfil all the steps of the system represented in figure 6.1, it was needed to format the results in a way that could be returned and easily accessible and understood by who requested it, which was done, resulting in the fulfilment of the last step and resulting in the complete system, ready to be used and tested.

After the construction of a way to automatically validate all the images, the last step that the system needed to perform to finish, as represented in figure 6.1, was to format the results and return them to the user who requested the validation. These results needed to be in a way that could be easily accessible and understood by who requested it, which was done, resulting in the fulfilment of the last step and resulting in the complete system, ready to be used and tested.

Before testing the created model, there was still a step that needed to be done. This last step corresponded to the definition of the threshold used by the object detection model to define what was considered a valid detection and what was not, meaning the level of confidence necessary consider a detection valid. To identify the best threshold value, first, it was made an analysis of the precision calculated for 21 different threshold values to the testing dataset. As shown in figure 6.3, this analysis resulted in 5 candidate values that achieved the same precision of 0.88. These values were the 0.05, 0.2, 0.3, 0.35 and 0.4.

Since there were so many candidate values, it was made a second analysis of the number of true positive objects found by the object detector for the same 21 different threshold values as the previous analysis, as shown in figure 6.2. Resulting in the choice of the value of 0.4 as the value used for the threshold by the object detector, since it was the one with the highest number of true positive cases, 192, being the difference to the second value was small of just one true positive case, 191.

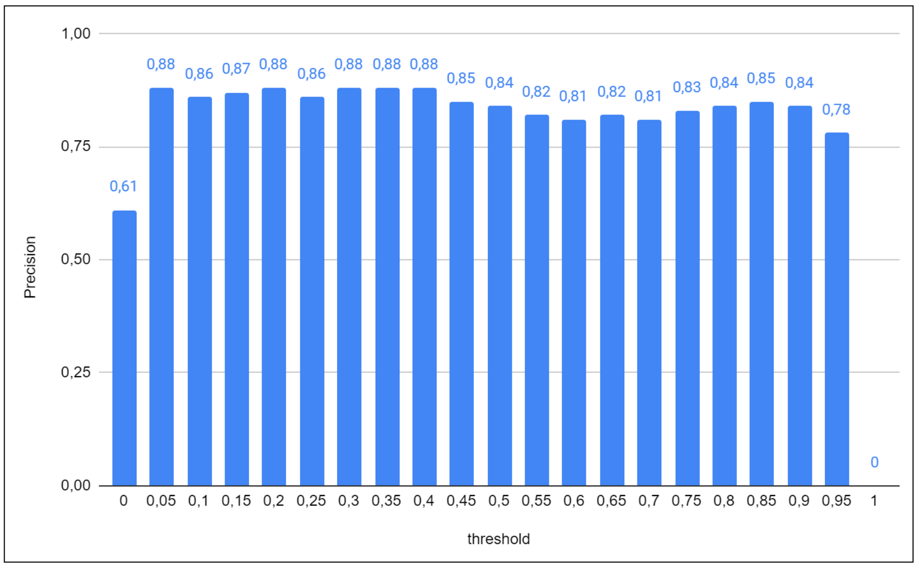


Figure 6.2: Calculated precision for each considered threshold.

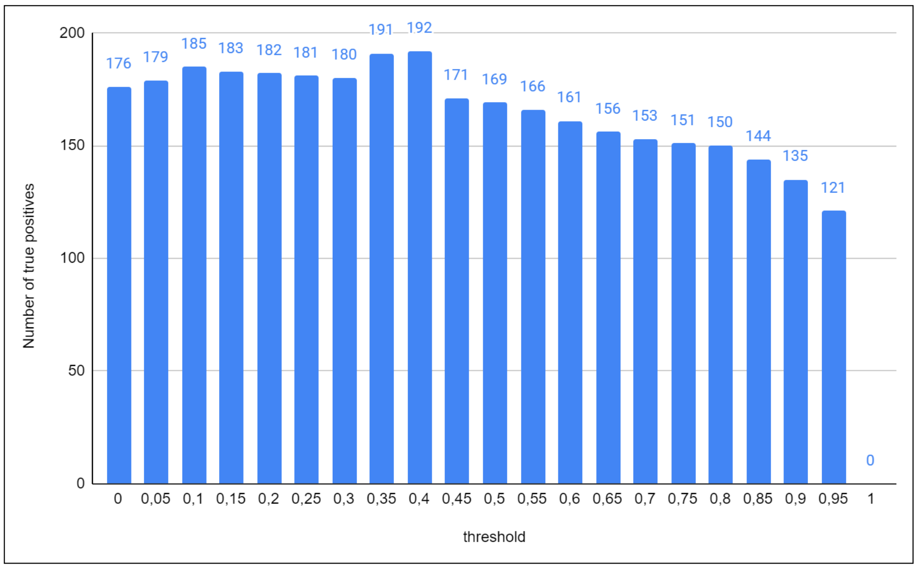


Figure 6.3: The number of true positives calculated for each considered threshold.

With the threshold value defined, it was necessary to validate the functioning of the created system with at least one entire mission to understand if the created system could be integrated with the Brands&Ninjas platform. So, three different images were selected, where the objective was to verify the presence of objects of the class Qalidus on the shelves of several stores. The number of images present in each of these missions can be seen in table 6.1. Mission one was constituted by 208 images. Mission two was constituted by 87 images, and mission three by 211 images.

With the missions selected and the system fully integrated, the remaining step to perform was to deploy the model in a remote machine from AWS[1] provided by RedLight. Since so far, all the tests had been performed locally. To test the performed deployment, the system's functioning was checked by selecting two images selected aleatorily and creating an answer for each one and sending them to the system to understand if the system could be accessed from outside and executed all the steps as expected. Initially, the system could not be accessed because the external ports were blocked, so after solving this minor issue, the test worked, and the platform was ready to be accessed from the outside.

After the selection of the missions for validation of the fully integrated system, the remaining step to perform was to deploy the model in a remote machine from AWS[2] provided by RedLight. Since so far, all the tests had been performed locally. To test the performed deployment, the system's functioning was checked by selecting two images selected aleatorily, and an answer was created for each one. Next, these two images were sent to the system to understand if the system could be accessed from outside and executed all the steps as expected. Initially, the system could not be accessed because the external ports were blocked, so after solving this minor issue, the test worked, and the platform was ready to be accessed from the outside.

Table 6.1: The number of images in each mission used for validation of the system

	Mission 1	Mission 2	Mission 3	Total images
Number of images	208	87	211	506

6.2 Platform in action

With the system ready to be used, the test to its capabilities corresponded to sending the missions selected and represented in table 6.1, one by one, and the respective ninjas' answers. To validate the obtained results, it was manually checked for each image if the answer given by the system regarding its validity was correct. To do so, the image would be open, and it would be verified if the objects of the class Qalidus were present or not and then compared with both the ninja answer and the system answer. The results of this analysis can be seen in table 6.2.

In table 6.2, the validated images correspond to images where both the system and the ninja answered the same about the presence of objects of the class Qalidus. Either by both answering that there was at least one object present or that no object was in the image. Of course, the validated images can be correct if the answer given by both the ninja and the system is the correct one and incorrect if both the ninja and the system answer incorrectly. Non-validated images correspond to images where the answers gave by the ninja and the system differ. When an image is non-validated, either the system or the ninja have made an error.

In mission one, the system was able to validate 87.02% of all the images, from which 98.89% were validated correctly, and 1.11% were validated incorrectly. The non-validated images corresponded to 12.98%, from which 51.85% were cases where the ninja made an error, and the system was correct by not validating it, and 48.15% were cases where the ninja answer was the correct one, and the system made the error.

For the second mission, the system was able to validate 88.49% of all the images, from which 98.71% were validated correctly and 1.29% were validated incorrectly. Therefore,

[1]`https://aws.amazon.com/`
[2]`https://aws.amazon.com/`

Table 6.2: The results of the validation of the images

	Mission 1	Mission 2	Mission 3	Total images
Validated images	179(86.05%)	76(87.36%)	200(94.79%)	455(89.92%)
Validated images Errors made by both the ninjas and the system	2(0.97%)	1(1.14%)	0(0%)	3(0.59%)
Non-validated images Errors made by the ninjas	14(6.73%)	6(6.90%)	6(2.84%)	26(5.14%)
Non-validated images Errors made by the system	13(6.25%)	4(4.60%)	5(2.37%)	22(4.35%)
Total images	208	87	211	506

the non-validated images corresponded to 11.5%, where 60% of the images non-validated were cases where the ninja made an error, and the system was correct by not validating it, and 40% were cases where the ninja answer was the correct one, and the system made the error.

In the third and final mission, the system was able to validate 94.79% of all the images, from which 100% were validated correctly and none was validated incorrectly. The non-validated images corresponded to 5.21%, where 54.51% were cases where the ninja made an error, and the system was correct by not validating it, and 45.49% were cases where the ninja answer was the correct one, and it was the system that made the error.

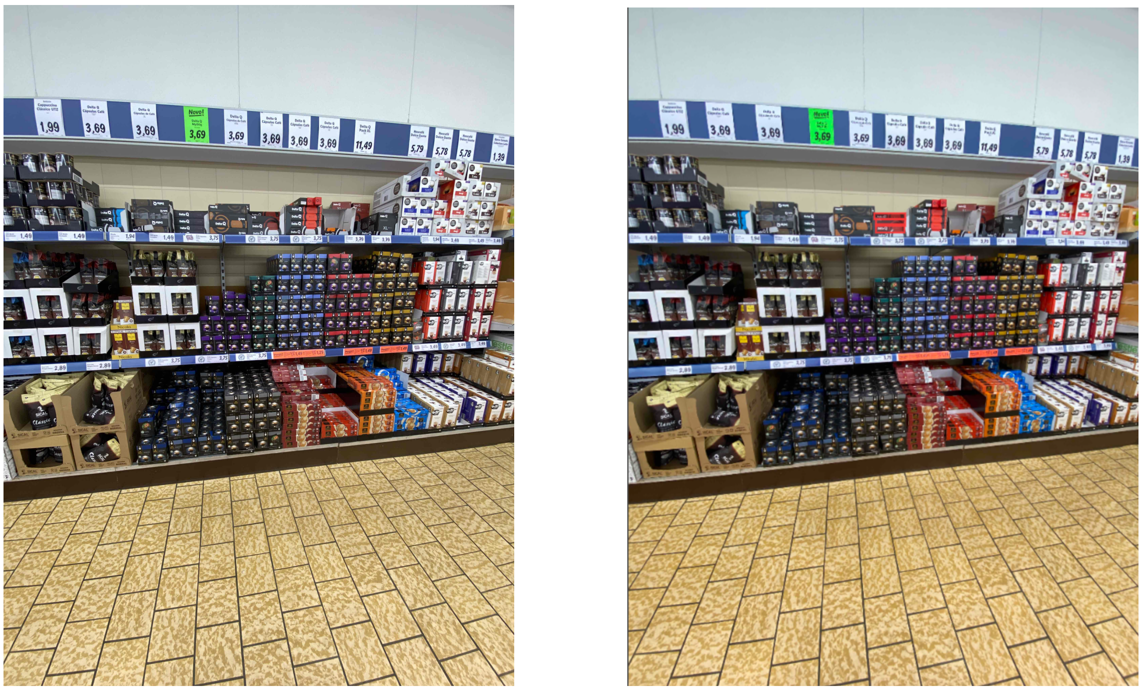
In table 6.2, the last column corresponds to the results when aggregating all the images. Thus, it is possible to understand that, in total, the system correctly predicted the presence of objects of the class Qalidus in 95.06% of the images, having mispredicted 4.94%. Thus, the system's validated images amounted to 90.51%, 99.35% were correctly validated, and 0.65% were incorrectly validated. The non-validated images correspond to 9.49% of all the images, from which 54.16% were cases where the ninja made an error and 45.84% corresponded to cases where the ninja answer was the correct, and it was the system that made the error. Meaning that over 90% of the images were correctly validated, and in total, about 95% of all the images were correctly predicted, which indicates that the built system could be used to validate images regarding the presence of objects of the class Qalidus by the Brands&Ninjas platform. From the images, it could not predict, 4.35% corresponded to images where the system found or was unable to found the objects and erroneously contradicted the answers given by the ninjas and 0.59% corresponded to validated errors made by the ninjas that the system was not able to solve.

The system could not find all the objects of the class Qalidus present in the validation missions used. Some of the errors made by the classifier came from the way the images were taken. Such as the error represented in figure 6.4, where the system could not find the objects of the class Qalidus present since they were too small for the classifier to find them, having been the image taken from too far away. This error probably occurred because the training dataset did not have any objects with such small dimensions, or another possible cause could be that the YOLOv4 model cannot find such small objects. A similar error made is represented in figure 6.5, where the system could not find the objects too close to the camera. In this case, the error made resulted from the fact that the system was not trained with any object occupying as much of the total image. So for future development of the system, this should be considered, and the dataset should be updated.

Another error made by the classifier resulted from the fact that the received images were taken too sideways, which result in just a part of the object being visible. Not even the use of the perspective skewing data augmentation technique could fully mitigate the occurrence of this error, as it is possible to verify in figure 6.6, where the system was unable to find the existing objects represented in the annotated image in figure 6.6b.

The system did not just made errors because of the way the images were taken. In some cases, the system incorrectly found objects where they did not exist. For example, as represented in figure 6.7, the system found an object and associated it with the Qalidus class incorrectly, being the found object represented in figure 6.7b. This error might have occurred because of the lighting in the position that the object was, demonstrating a case where the lighting data augmentation techniques applied were not able to prevent the occurrence of this error.

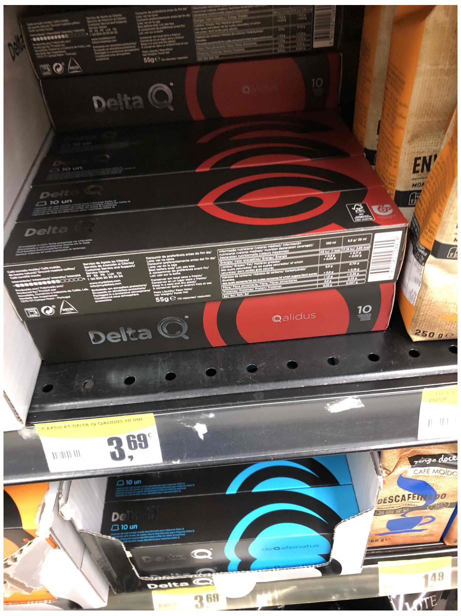
Although the system did make some mistakes, it also found errors made by the ninjas, as represented in Figures 6.8 and 6.9. In these cases, the system correctly identified the objects that the ninjas missed, as represented in figures 6.8b and 6.9b, where are represented the correctly identified objects of the class Qalidus by the system contrarily to the ninjas that were unable to found them.



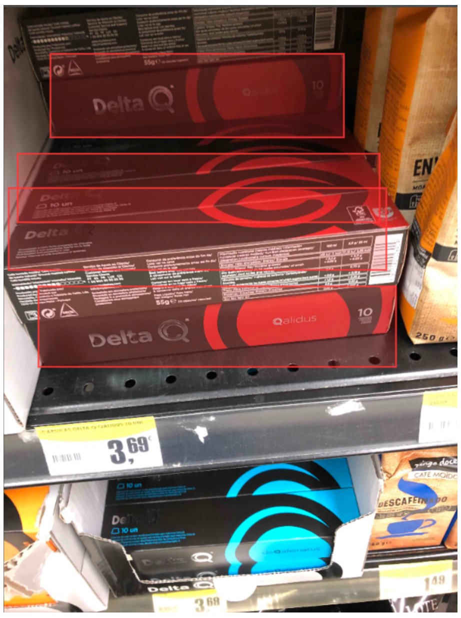
(a) The image received from the ninja that was taken too far away.

(b) The image received from the ninja that was annotated shows the objects the system should have found.

Figure 6.4: Example of a case where the image was taken too far away resulting in objects to small to be identified.

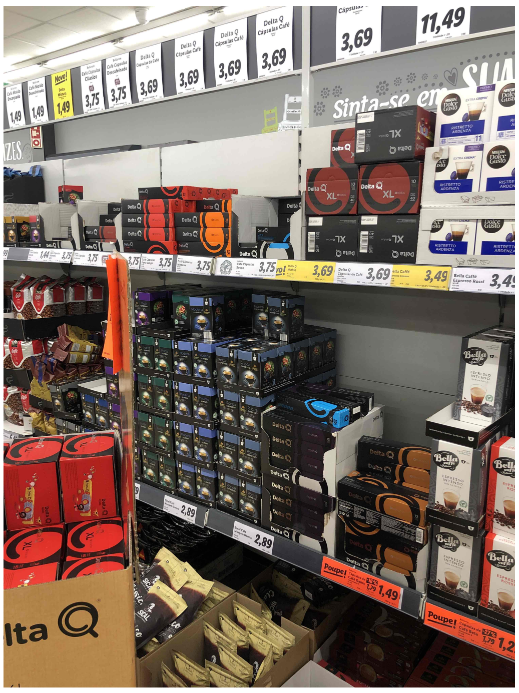


(a) The image received from the ninja that was taken too close.

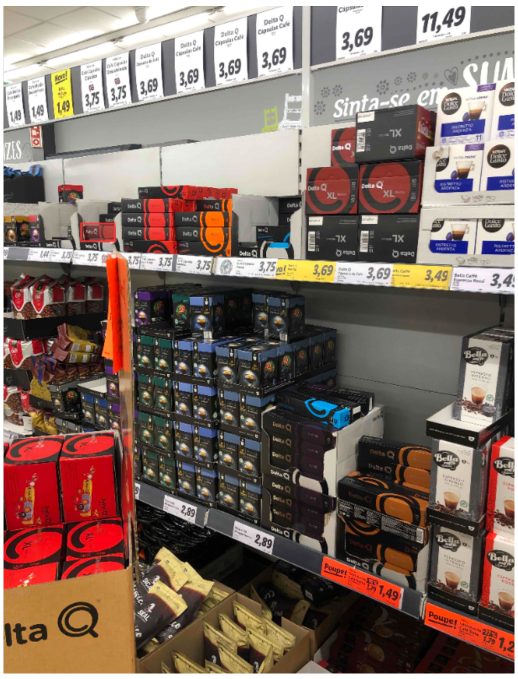


(b) The image received from the ninja that was taken too close, annotated regarding the objects the system should have found.

Figure 6.5: Example of an image that was taken too close to the objects of the Qalidus class and that the system was not able to identify.



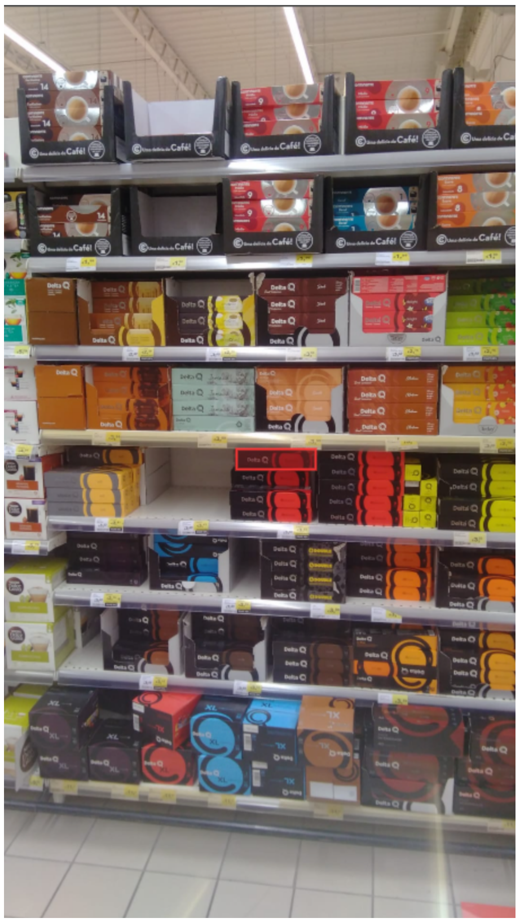
(a) The image received from the ninja that was taken too sideways.



(b) The image received from the ninja that was taken too sideways, annotated regarding the presence of objects of the class Qalidus.

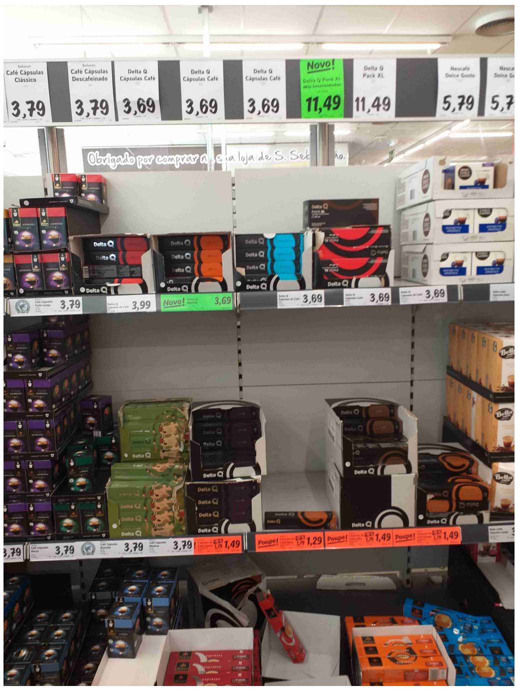
Figure 6.6: An example of a case where the image received was taken too sideways in perspective.

(a) The image received from the ninja that did not contain any objects of the class Qalidus.

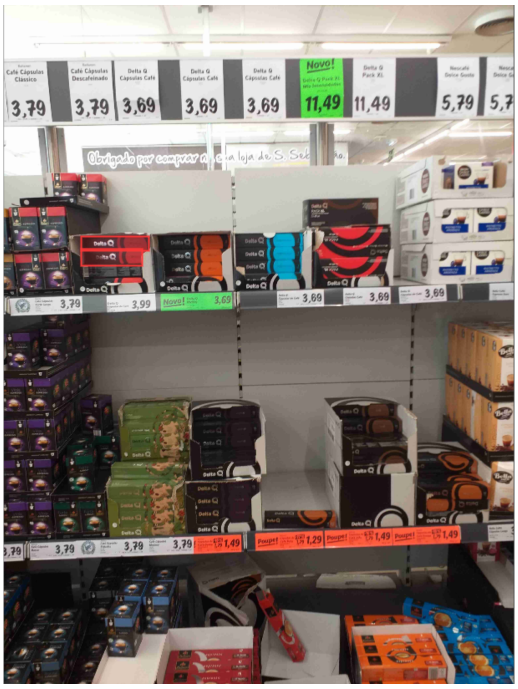


(b) The image received from the ninja that did not contain any objects of the class Qalidus and the annotated object found by the system.

Figure 6.7: An example of an image where the created system should not have had found any objects, contrarily to what happened.

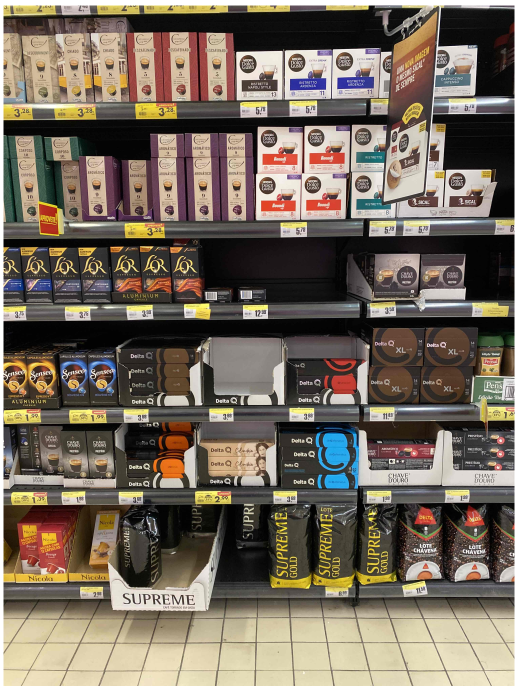


(a) The image received from the ninja that had the answer associated that no object of the class Qalidus was present.

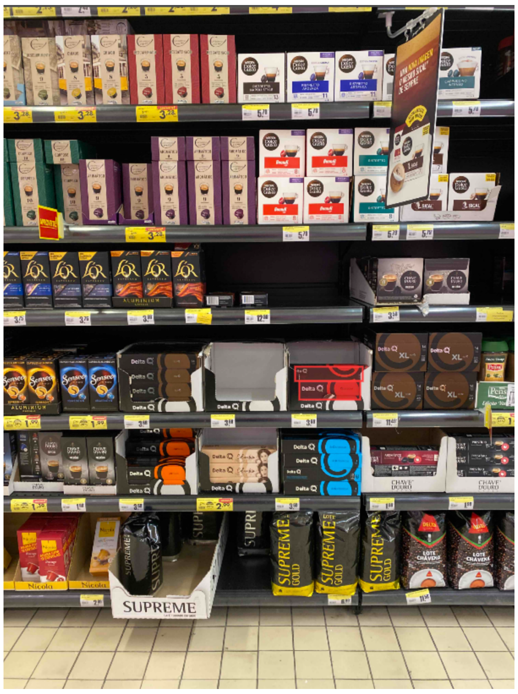


(b) The image received from the ninja with the object of the class Qalidus found annotated, contrarily to what was the ninja answer.

Figure 6.8: First example of an error made by a ninja that the system was able to identify.



(a) The image received from the ninja that had the answer associated that no object of the class Qalidus was present.



(b) The image received from the ninja with the object of the class Qalidus found annotated, contrarily to what was the ninja answer.

Figure 6.9: Second example of an error made by a ninja that the system was able to identify.

Chapter 7

Conclusion

The purpose of this dissertation was to create a system for the automatic validation of images. The reason that led to the creation of such a system came from the fact that the ninjas' answers provided from the performed missions were not being fully validated. Only a maximum of 20% of the received images was being validated, and the 10 to 15% error percentage indicated that there were probably multiple errors that were being unseen and leading to incorrect results to give to the brands who requested the missions. So, to demonstrate that such errors could be solved, a system was constructed, being selected a more straightforward task of validating the presence of one product on an image to demonstrate it. The product selected for this was the Delta Q Qalidus package of 10 capsules product, class Qalidus. The work described had the objective of validating missions regarding the presence of objects of the class Qalidus and the creation of a system that could be integrated into the Brands&Ninjas platform.

The first step in the creation of a system for detecting the presence of objects of the class Qalidus corresponded to an analysis of the most suited CV tasks to create this system. With this analysis, considering that an image might have multiple different objects from multiple classes, the choice could be either object detection or instance segmentation. However, since image segmentation uses extra calculations to compute the found objects shape, the choice was to use object detection models. To verify that object detection was indeed the best CV task to use, an analysis of the RedLight direct and indirect competitors and their approaches to related problems was made. The competitors' approaches' analysis showed that object detection is indeed their chosen CV task to solve problems related to the existing one, which verified the assumption that object detection is the most suited CV task.

Considering the objective of identifying the presence of objects on the shelves, a survey state of the art on object detection was performed. The objective of this survey was to understand what should be the object detector used for creating a system to solve the considered task, since the objective was not to create a new object detector model specific to this particular problem. Instead, the objective was to understand what worked for different solutions and try different models to choose the best one to be used as an essential part of the system. Therefore a state of the art analysis on object detection was performed, from the more classical approaches to the more recent ones. The objective of this analysis was to select object detection models that could be used to create a system the most automatic possible that could be used for future scaling for more than one object, with an easy configuration and good performance. As a result, the analysis made resulted in the choice of four different models. Two more classical and straightforward that used

Haar and HOG features. And two of the most recent that have achieved better results in the latest competitions being the current state of the art models on object detection, the YOLOv4 model that is a one-stage detector and the EfficientDet that is a two-stage detector.

The experimentation showed that it was possible to verify that recent models based on deep learning work better than those based on template features, such as HOG or Haar features based models. The model with best performance was the YOLOv4 model that was trained using data augmentation techniques manually implemented with an AP of 92.60% on the testing dataset. The results obtained with the YOLOv4 object detector and its training simplicity compared to the other trained detectors led to the choice of the trained YOLOv4 model as the object detector used in the next step, the system deployment.

The obtained results show the potential of the developed approach for the Brands&Ninjas platform. So the object detector was integrated into a system capable of using the object detector answers compared with the ninjas' answers to validate the images provided. To test the created system were used three distinct missions, to validate its capability as a whole to validate images regarding the presence of objects of the class Qalidus. In total, in all the used missions for testing, the system correctly identified the presence of objects of the class Qalidus on 95.06% of them. Furthermore, from the set of 506 images used, 90.51% were validated, corresponding to cases where the system and ninjas' answers were the same.

From all the 458 images validated, 99.35% were images correctly validated, corresponding to 455 images, and 0.65% were incorrectly validated, corresponding to only three images where both the system and the ninja made a mistake. On the other hand, the non-validated images amounted to 9.49% of all the 506 images, from which 54.16% corresponded to cases where the ninja made an error, that corresponded to 26 images and 45.84% to cases where the system made an error, and the ninja answer was correct, corresponding this value to 22 images. Finally, the obtained results significantly improve over the initial maximum of 20% manually validated images, compared to the system's ability to validate 90% of the images sent by the ninjas and their answers to each one regarding the presence of objects of the class Qalidus, with a 99.35% possibility of an image having been validated correctly.

Although all requirements were met and the building of the system was successful, there are still steps needed to be done in the future to improve the created system and make it more robust. First, the subsequent development that should be done is to create a system that can detect objects of multiple classes and not only objects of the class Qalidus. To do this, subsequent development, it is necessary to validate which option has the best results, the creation of an object detection model for each new class or a creation of a model that can detect multiple different classes, taking into consideration the fact that there will always be needed new classes and that this number is not supposed to be static.

A partial development that the YOLOv4 model already has and that was partially analysed in this dissertation is the number of objects in each class that the detector can find, and that corresponds to a different task performed by the ninjas. This task from the ninjas perspective corresponds to how many objects of the same class are present on the shelf, being this an important task usually requested by the brands.

References

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

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] Pedro Felzenszwalb, David Mcallester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. volume 8:, 06 2008.

[5] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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

[8] Ross Girshick, Pedro Felzenszwalb, and David McAllester. Object detection with grammar models. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Lecture Notes in Computer Science*, page 346–361, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[11] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[12] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, and et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, Mar 2020.

[13] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[18] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[20] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[21] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[23] Adam Schmidt and Andrzej Kasiński. *The Performance of the Haar Cascade Classifiers Applied to the Face and Eyes Detection*, volume 45, pages 816–823. 10 2007.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[25] Kamrul Hasan Talukder and Koichi Harada. Haar wavelet based approach for image compression and quality assessment of compressed image. *CoRR*, abs/1010.4084, 2010.

[26] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[27] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020.

[28] Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012, 11 2012.

[29] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001.

[30] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[31] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.