

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Automatic recognition of groceries for smart shopping

Manuel Fernandes Sousa

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Luís Paulo Reis

August 19, 2024

Abstract

Smart shopping is a growing area that aims to create tools to revolutionize the shopping experience for both customers and retailers. The advantages for customers focus on an improved shopping experience through access to the best promotions and a more fluid local shopping experience by providing personalized recommendations and a seamless checkout process. For retailers, these advances promise access to diverse strategic advantages. It allows businesses to gather various metrics and statistics about their products and where they are sold. This helps them boost sales by analyzing the data, reduce stockouts, manage consumer interaction with products to increase engagement, and improve the efficiency of the checkout process and overall buying experience. This, in turn, increases consumer interest and generates revenue.

Computer vision is a pivotal tool in this field. It uses artificial intelligence (AI) to analyze images and the surrounding environment, which promises to revolutionize the industry by reducing the difficulties present in the market and helping with monotonous and repetitive work. For example, computer vision can be used for real-time inventory management, analyzing, and detecting products on shelves. It can also help monitor store traffic and customer behavior, providing insights that might be used to optimize store layouts and product placement. Additionally, it helps with the self-checkout process by recognizing products without the need for barcodes, thus speeding up the process. Although this last application is very promising, it also presents multiple challenges, particularly concerning the wide variety of products with the most diverse packages and designs. Even the same product may have different packages at a given time. However, if done well, this application in the checkout will allow retailers to increase the speed of the process and reduce costs and labor requirements, which, combined with a more efficient experience, will produce more revenue. To achieve this, it's important that the system can be easily adapted to the market in which it operates. This work seeks a solution for creating a system capable of carrying out this image classification process quickly and efficiently, focusing on the volatility of the appearance of the products and the products themselves, depending on the market. Particular attention was paid to the methodology for ways to create datasets to train these systems so that it would be easy for any user to allow retailers to have their own employees create the data needed for the system. Several datasets have been created to test these systems and understand how different backgrounds influence them. In the future, the idea is to implement the object detection and classification system in a shopping cart, so it's necessary to consider how easy it is to implement it in a mobile system with a small footprint.

Various systems are currently being developed to try to overcome the above-mentioned shortcomings. RFID and barcode systems are the simplest, but they require a constant inter-

net connection and have difficulties with overlapping products. More advanced systems, which already make use of computer vision, still have some difficulties, requiring weight sensors or object detection to be minimally effective. For this reason, the search for greater precision continues without ever losing sight of the need for a system capable of performing in real time for implementation in a real scenario. Some other alternatives try to create combinations of different networks, but this introduces a significant overhead, which requires more computing power and removes mobility. Some studies have begun with synthetic datasets showing promise, but performance in a real environment still needs improvement.

This work aimed to find the best way to develop a good dataset using advanced clustering and classification techniques that were explored for supermarket product images, using various feature extraction methods and clustering algorithms. Initial experiments with K-Means clustering and feature extractors (RGB Histogram, HSV Histogram, SIFT, VGG-16), along with preprocessing techniques like Center_Crop (that crops the image in a way that enhances the central part of a given image) and Bilateral Filters, showed suboptimal results. Adapting the Support Vector Machine (SVM) for clustering with multiple image perspectives per product improved performance. The best result was using a VGG-16 network as feature extractor and SVM achieving a silhouette coefficient of 0.355 and a Davies-Bouldin Index of 1.245, which gave 0 misclusterizations for the given images. However, of the different tests carried out, this was one of the only two cases in which there were 0 clustering errors, while there was always some failure in the others, which was proven to invalidate the datasets for subsequent training of the classification network.

Since the results were not satisfactory, an alternative was searched. Using a Mask R-CNN network, masks of the products to be detected were created, and by pasting them randomly and with some augmentation onto background images, synthetic images were created. These images were used to train the YOLOv5 object detection, achieving a precision of 89.8% and a recall of 94.5%. These findings underline the complexities of feature extraction, clustering and classification in variable environments. Given the good results with the synthetic datasets, further advancements should deepen this path, with testing in real-world scenarios.

In summary, this research focuses on developing a comprehensive system for smart shopping, leveraging deep learning and advanced clustering techniques to improve image processing and analysis, ultimately enhancing the shopping experience for customers and retailers alike. K-Means clustering in conjunction with various feature extractors produced less than-ideal outcomes in the initial attempts. Although a VGG-16 network with SVM produced the best results, clustering discrepancies persisted. Transitioning to artificial datasets generated by Mask R-CNN and YOLOv5 revealed notable enhancements. These results highlight the significance of ongoing testing and improvement in real-world settings and imply that synthetic datasets have promise for future advancement. The study highlights the necessity of flexible, effective picture dataset creation methods in dynamic retail environments, opening the door for advanced solutions that boost productivity, cut expenses, and raise consumer satisfaction.

keywords: Clustering, Computer Vision, Dataset, Deep Learning, Efficient-Net, K-Means, Smart Shopping System, Retail, YOLO

Resumo

As compras inteligentes são uma área em crescimento que visa criar ferramentas para revolucionar a experiência de compra, tanto para os clientes como para os retalhistas. As vantagens para os clientes centram-se numa experiência de compra melhorada através do acesso às melhores promoções e numa experiência de compra local mais fluida, fornecendo recomendações personalizadas e um processo de checkout sem problemas. Para os retalhistas, estes avanços prometem o acesso a diversas vantagens estratégicas. Permitem-lhes reunir várias métricas e estatísticas sobre os seus produtos e os locais onde são vendidos. Isto ajuda-os a aumentar as vendas através da análise dos dados, a reduzir as ruturas de stock, a gerir a interação dos consumidores com os produtos para aumentar o envolvimento e a melhorar a eficiência do processo de checkout e da experiência de compra em geral. Isto, por sua vez, aumenta o interesse do consumidor e gera receitas.

A visão por computador é uma ferramenta fundamental neste domínio. Utiliza a inteligência artificial (IA) para analisar imagens e o ambiente circundante, o que promete revolucionar a indústria, reduzindo as dificuldades presentes no mercado e ajudando no trabalho monótono e repetitivo. Por exemplo, a visão computacional pode ser utilizada para a gestão de inventário em tempo real, analisando e detetando produtos nas prateleiras. Também pode ajudar a monitorizar o tráfego da loja e o comportamento dos clientes, fornecendo informações que podem ser utilizadas para otimizar a disposição das lojas e a colocação dos produtos. Além disso, ajuda no processo de auto-checkout, reconhecendo produtos sem a necessidade de códigos de barras, acelerando assim o processo. Embora esta última aplicação seja muito promissora, também apresenta vários desafios, nomeadamente no que diz respeito à grande variedade de produtos com as mais diversas embalagens e designs. Até o mesmo produto pode ter embalagens diferentes num determinado momento. No entanto, se bem executada, esta aplicação no processo de checkout permitirá aos retalhistas aumentar a velocidade do processo e reduzir os custos e a necessidade de mão de obra, o que, combinado com uma experiência mais eficiente, produzirá mais receitas. Para isso, é importante que o sistema possa ser facilmente adaptado ao mercado em que atua. Este trabalho procura uma solução para a criação de um sistema capaz de realizar este processo de classificação de imagens de forma rápida e eficiente, focando a volatilidade da aparência dos produtos e dos próprios produtos, consoante o mercado. Foi dada especial atenção à metodologia para a criação de conjuntos de dados para treinar estes sistemas, de modo que seja fácil para qualquer utilizador; permitindo que os retalhistas tenham os seus próprios empregados a criar os dados necessários para treinar o sistema. Foram criados vários conjuntos de dados para testar estes sistemas e compreender a forma como os diferentes contextos os influenciam. No

futuro, a ideia é implementar o sistema de detecção e classificação num carrinho de compras, pelo que é necessário considerar a facilidade de o implementar num sistema móvel com uma pegada pequena.

Atualmente, estão a ser desenvolvidos vários sistemas para tentar ultrapassar as deficiências acima referidas. Os sistemas RFID e de código de barras são os mais simples, mas requerem uma ligação constante à Internet e têm dificuldades com a sobreposição de produtos. Os sistemas mais avançados, que já recorrem à visão por computador, têm ainda algumas dificuldades, necessitando de sensores de peso ou de detecção de objetos para serem minimamente eficazes. Por esta razão, a procura de maior precisão continua, sem nunca perder de vista a necessidade de um sistema capaz de atuar em tempo real para implementação num cenário real. Algumas outras alternativas tentam criar combinações de redes diferentes, mas isso introduz uma sobrecarga significativa, o que exige mais poder de computação e retira mobilidade. Alguns estudos foram iniciados com conjuntos de dados sintéticos que se revelam promissores, mas o desempenho num ambiente real ainda precisa de ser melhorado.

Este trabalho teve como objetivo encontrar a melhor forma de desenvolver um bom conjunto de dados utilizando técnicas avançadas de agrupamento e classificação que foram exploradas para imagens de produtos de supermercado, utilizando vários métodos de extração de características e algoritmos de agrupamento. As experiências iniciais com o clustering K-Means e os extratores de características (Histograma RGB, Histograma HSV, SIFT, VGG-16), juntamente com técnicas de pré-processamento como Center_Crop (que corta a imagem de forma a realçar a parte central de uma determinada imagem) e Filtros Bilaterais, mostraram resultados abaixo do ideal. A adaptação da Máquina de Vetores de Suporte (SVM) para agrupamento com várias perspetivas de imagem por produto melhorou o desempenho. O melhor resultado foi obtido com a utilização de uma rede VGG-16 como extrator de características e com a SVM, que alcançou um coeficiente de silhueta de 0,355 e um índice de Davies-Bouldin de 1,245, o que deu origem a 0 erros de agrupamento para as imagens em causa. No entanto, dos diferentes testes efetuados, este foi um dos dois únicos casos em que se registaram 0 erros de clusterização, enquanto que nos outros houve sempre alguma falha, o que comprovadamente invalidou os conjuntos de dados para posterior treino da rede de classificação. Como os resultados não foram satisfatórios, procurou-se uma alternativa. Utilizando uma rede Mask R-CNN, foram criadas máscaras dos produtos a detetar e, colando-as aleatoriamente e com algum aumento em imagens de fundo, foram criadas imagens sintéticas. Estas imagens foram utilizadas para treinar a detecção de objetos do YOLOv5, obtendo-se uma precisão de 89,8% e uma recuperação de 94,5%. Estes resultados sublinham as complexidades da extração de características, agrupamento e classificação em ambientes variáveis. Tendo em conta os bons resultados obtidos com os conjuntos de dados sintéticos, os avanços futuros devem aprofundar este caminho, com testes em cenários do mundo real.

Em suma, esta investigação centra-se no desenvolvimento de um sistema abrangente para compras inteligentes, tirando partido da aprendizagem profunda e de técnicas avançadas de agrupamento para melhorar o processamento e a análise de imagens, melhorando, em última análise, a experiência de compra tanto para os clientes como para os retalhistas. O algoritmo K-Means em conjunto com vários extratores de características produziu resultados menos que ideais nas tentativas iniciais. Embora uma rede VGG-16 com SVM tenha produzido os melhores

resultados, as discrepâncias de agrupamento persistiram. A transição para conjuntos de dados artificiais gerados por Mask R-CNN e YOLOv5 revelou melhorias notáveis. Estes resultados realçam a importância dos testes e melhorias contínuas em contextos reais e implicam que os conjuntos de dados sintéticos são promissores para avanços futuros. O estudo realça a necessidade de métodos flexíveis e eficazes de criação de conjuntos de dados de imagens em ambientes de retalho dinâmicos, abrindo a porta a soluções avançadas que aumentem a produtividade, reduzam as despesas e aumentem a satisfação dos consumidores.

palavras-chave: Clustering, Visão por Computador, Datasets, Deep Learning, EfficientNet, K-Means, Sistema de Compras Inteligente, Retalho, YOLO

Sustainable Development Goals

In 2015, the United Nations General Assembly approved the "2030 Agenda," a plan centered on 17 Sustainable Development Goals (SDGs) aimed at promoting sustainable development through actions focused on "people, planet, prosperity, and peace." This agenda addresses the economic, social, and environmental dimensions of sustainable development, expanding on the themes of its predecessor, the Millennium Development Goals. The SDGs are universal, applying to all countries to enhance quality of life while safeguarding the planet.

One of the aims was to ensure that this thesis is aligned with these objectives. Table 1 provides an analysis of the objectives that should be considered during the development of this study, and it outlines how to assess whether they are being achieved.

Table 1: Contributions to SDGs

SDG	Target	Contribution	Performance Indicators and Metrics
8	Achieve higher economic productivity through technological innovation (Target 8.2)	Increase economic productivity in the retail sector by implementing an autonomous checkout system, which allows workers to be reallocated to other company divisions.	- Increase sales revenue - Reduce operational costs
8	Improve global resource efficiency in consumption and production (Target 8.4)	Promote efficient resource use in retail operations, contributing to sustainability and reduced environmental impact.	- Reduce inventory waste - Improve energy efficiency

9	Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation (Target 9.4)	Develop a sustainable retail infrastructure with an advanced visual system for checkout, reducing the environmental footprint through decreased need for physical checkout counters and less energy consumption.	- Customer satisfaction: Increase by 10% - Decrease in energy usage - Increase in system efficiency
12	Ensure sustainable management and efficient use of natural resources (Target 12.2)	Improve resource efficiency in the retail sector by minimizing waste associated with traditional by reducing paper utilization and plastic usage.	- Reduction in waste produced - Reduction in paper/plastic usage
13	Improve institutional capacity on climate change mitigation (Target 13.3)	Reduce the carbon footprint of retail operations by integrating an eco-friendly checkout system that minimizes energy consumption and enhances operational efficiency.	- Reduction in carbon emissions by improving energy efficiency ratings

Agradecimentos

Este projeto marca o culminar da minha vida estudantil e académica. O simples facto de chegar a este ponto, não seria possível sem algumas pessoas que passaram pela minha jornada mais recentemente e outras que me acompanham há muito mais.

Em primeiro lugar gostaria de agradecer ao meu supervisor, o Mestre José Borges, por todo o acompanhamento realizado ao longo deste trabalho, e a disponibilidade dada para me auxiliar. A orientação (e pressão) fornecidas foram cruciais para a realização do mesmo. Gostaria também de agradecer em particular aos meus colegas da equipa de inovação, que me acolheram muito bem na equipa e me fizeram sentir em casa. O reconforto dado por alguém como eles, que já estiveram nesta posição, auxiliou-me a ultrapassar receios que foram surgindo.

Expresso a minha gratidão também ao meu orientador, Doutor Luís Paulo Reis, cujos conselhos astutos tiveram um papel fundamental no desenvolvimento deste projeto.

O acompanhamento dado pelos meus amigos durante a minha vida, quer a nível escolar, quer pessoal, garantiu que este caminho se tornasse mais fácil, agradável e definitivamente memorável. A vocês, que sabem quem são, o meu obrigado por alinharem em todos os momentos que vivemos.

Quero deixar um agradecimento especial também à Carolina, que me aturou nos momentos em que estava mais preocupado e de cabeça cheia, e sempre se revelou muito compreensiva e preocupada.

Por fim, o meu maior obrigado vai para os meus pais e o meu irmão, pois sem eles não poderia ter chegado a este ponto. Obrigado por sempre terem investido na minha educação e por terem acreditado em mim. E também a vocês, obrigado por terem muita paciência e preocupação por mim.

A todos os aqui referidos, deixo mais uma vez o meu agradecimento mais profundo e sentido.

Manuel Fernandes Sousa

*“A man who dares to waste one hour of time
has not discovered the value of life.”*

Charles Darwin

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Objective	2
1.4	Document Structure	3
2	Background	5
2.1	Video Processing	5
2.2	Machine Learning	7
2.3	Clustering Algorithms	8
2.3.1	K-Means	9
2.3.2	DBSCAN	9
2.3.3	Gaussian Mixture Models (GMM)	10
2.3.4	Support Vector Machines	11
2.4	Computer Vision	12
2.4.1	Deep Learning Methods	13
2.4.2	Datasets	15
2.4.3	Object Detection	17
2.4.4	Image Classification	21
2.4.5	Image Segmentation	25
2.5	Comparison Metrics	27
2.5.1	Clustering Evaluation Metrics	27
2.5.2	Object Classification Evaluation Metrics	28
2.5.3	Object Detection Evaluation Metrics	30
3	State of The Art	33
3.1	Advancements on Computer Vision	33
3.2	Checkout Retail Systems	35
3.2.1	Barcode and RFID Systems	35
3.2.2	Visual Systems	36

4	Methodologies and Experiment Setup	41
4.1	Dataset Background Characteristics	41
4.2	Dataset Creation via Clustering	44
4.2.1	Video Extraction	45
4.2.2	Image Clustering	46
4.2.3	Dataset Testing	58
4.3	Synthetic Dataset Generation	59
4.3.1	Generating the Objects Data	59
4.3.2	Generating the new images	61
4.3.3	Converting the metadata	62
4.3.4	Synthetic Dataset Testing	62
4.4	Evaluation Metrics	63
5	Results and Discussion	65
5.1	Dataset Background Characteristics	66
5.2	Generated Datasets	67
5.3	Dataset Clustering	72
5.4	Dataset Testing	79
5.4.1	Datasets created for clustering testing	79
5.4.2	Dataset with metadata created via clustering	86
5.5	Synthetic Dataset Generation	86
6	Conclusions	91
A	Request for Photographic Record with Consent	93
B	Dash Interface	99
	References	103

List of Figures

2.1	Clustering Models [1]	8
2.2	Proposed system for image synthesis by Tang et al. [2]	16
2.3	SSD Network Structure. [3]	18
2.4	Faster R-CNN Network Structure. [3]	19
2.5	Basic Classification Network Structure. [4]	21
2.6	Structure of a Residual Block	23
2.7	DenseNet Network Structure[5]	24
2.8	Compound Scaling [6]	25
2.9	U-Net Network Structure	26
2.10	Mask R-CNN Framework [7]	27
4.1	Image Example of the different Datasets	42
4.2	Number of images per class in each dataset	43
4.3	Architecture of the <i>Original Network</i>	43
4.4	Application of the 'find_roi' function	48
4.5	Example images and respective histograms	51
4.6	Extraction of SIFT Features	52
4.7	Application example of the segmentation network	60
5.1	Confusion matrices obtained from testing	66
5.2	Frames extracted from the original video	67
5.3	Cropped video frames with single types of cereal.	68
5.4	Frames extracted from the Original Dataset 2 video	69
5.5	Frames filtered for the Original Dataset 2	70
5.6	Original Dataset 3	71
5.7	Original Dataset 4	71
5.8	Testing Dataset	72
5.9	Original Dataset 1 clusters created by K-Means algorithm with histogram RGB features	73
5.10	Features extracted by the VGG-16 network of an image of the Original Dataset 2	77
5.11	Training loss and accuracy for Original Network trained with Original Dataset 2	80
5.12	F1-Score and confusion matrix resultant from testing for ResNet network trained with Original Dataset 2	81

5.13	Training loss and accuracy for original network trained with Original Dataset 3	82
5.14	F1-Score and confusion matrix resultant from testing for Original Network trained with Original Dataset 3	82
5.15	Training loss and accuracy for original network trained with Original Dataset 4	83
5.16	F1-Score and confusion matrix resultant from testing for Original Network trained with Original Dataset 4	84
5.17	Training loss and accuracy for ResNet network trained with Original Dataset 2 modified	86
5.18	F1-Score and confusion matrix resultant from testing for ResNet Network trained with Original Dataset 2 modified	87
5.19	Synthetic images generated	87
5.20	Metrics graphs resultant from training and validation on the synthetic images dataset	88
5.21	Confusion matrix generated from the validation dataset	89
5.22	Validation results	89
B.1	Main dashboard page	100
B.2	Main dashboard page displaying results from a clustering test	101
B.3	Second dashboard page	102

List of Tables

1	Contributions to SDGs	vii
5.1	Metrics obtained from background comparison testing	66
5.2	Clustering performance on Original Dataset 1 using single image per centroid .	74
5.3	Clustering performance on Original Dataset 2 using single image per centroid .	75
5.4	Clustering performance on Original Dataset 2 using two images per centroid . .	76
5.5	Clustering performance on Original Dataset 3 using two images per centroid . .	77
5.6	Clustering performance on Original Dataset 4 using multiple images per centroid	78
5.7	Metrics obtained from the dataset classification testing	85

Abbreviations and Symbols

BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Key
CNN	Convolutional Neural Network
DoG	Difference of Gaussians
FAST	Features from Accelerated Test
IOU	Intersection-Over-Unit
IQR	Interquartile Range
JSON	JavaScript Object Notation
ORB	Oriented Fast and Rotated BRIEF
SIFT	Scale Invariant Feature Transform
SSD	Single Shot Detector
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
ROI	Region Of Interest
R-CNN	Region-Bases Convolutional Neural Network
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Context

In today's society, where people are increasingly busy, anything that can help save time is considered a valuable improvement in quality of life. Roy et al. state that multiple retailers already introduced new technologies in their stores to enhance the shopping experience [8]. And although they are being utilized, they all have some limitations, such as the time or effort required during checkout, the cost of implementation, or even the environmental impact of existing systems. As a result, smart shopping has gained significant attention in recent years. That's where the theme of this dissertation wants to fall in. This dissertation aims to develop a competitive algorithm for the automatic recognition of retail products.

One of the main problems that consumers encounter is related to the final amount to pay. The fact that they add products to the cart without having a way to control the sum of the products' value makes this control difficult. For this reason, consumers often arrive at checkout with more products than they can afford [9]. Implementing a system like the one proposed in this work not only will solve this problem but will also reduce waiting times and enhance the shopping experience of the customers by providing easily accessible product information, increased accessibility and interaction, customized product selection, and enhanced customer service.

It also helps the workers to keep an up-to-date inventory streamlining order planning and shelf restocking, as well as lowering labor costs and efforts [8]. This is reinforced by Melek et. al [10] which states that "31% of the customers who cannot find the requested item on the shelf buy the same product from another grocery store, 22% buy the different brand of the same product, and 11% give up the product when they cannot find it". And, as recalled by Wei et. al [11], this type of system associated with other in store technologies (IST [8]) can assist visually

impaired people to shop easily or even independently. And although not directly seen, the actual buying method involves big logistics, especially the payment part.

1.2 Motivation

The context of retail is the base of the idea for this project and what puts the biggest constraints and challenges in it.

Nowadays, retail shopping is a big part of the day-to-day life for the majority of people. It's important to reduce the time spent on any of the stages of the purchasing process. As Guimarães et al. [12] state, people want this process to be fast and simple, finding great value in tools that allow them to have the most customized and streamlined experience. Starting with the product brand decision process, searching for the respective product in-store, browsing in-store, and finally, the payment process.

Generally using human labor, this market has sought to evolve in recent times. It is a market in constant expansion and with constant changes, as new products are constantly introduced and many of the existing ones change. This is particularly encouraged by companies' search for more efficient marketing of their products. This often leads to changes in various appearance factors of the products [12]. The first characteristic is the logo, which is the main point of identification of a certain product, but thanks to more recent studies and market analyses, alternatives have increasingly been tried to appeal to the consumer. Changing the format of a package or two colors, changing the arrangement of graphic elements, or even adding graphic elements that indicate temporary promotions. Although very appealing to the end consumer, they introduce major limitations for systems based on computer vision.

Even so, because of these limitations, product detection and recognition in retail stores is an important computer vision problem with ongoing investigation from both industrial and academic researchers [10].

1.3 Objective

The final physical realization of the concept is a shopping cart equipped with built-in cameras. These cameras are connected to a computer that does a real-time analysis of the inside of the shopping cart and identifies the products on the inside. At the same time that the products are added to the cart and identified, they are added to a virtual shopping cart, allowing for automatic deduction of the total cost from the customer's account upon exiting the store. This system needs the account and the system to be pre-configured.

However the focus of this work is primarily on the image processing subsystem.

To grant the final goal, some activities will be held as objectives:

- Development of a dataset creation system that is easy for the average user to implement and use. The idea is to find an alternative that makes it easier to create datasets, given the need to constantly update them due to the variability of product appearance. In the case of this work, any dataset developed is focused on the Portuguese market;
- Development of the algorithm that will handle the taken images, filter them, identify the products in them, and then label them. This requires the use of computer vision and deep neural networks;
- Improve the developed system;
- Compare the developed system with the existing technologies.

1.4 Document Structure

This dissertation is divided into six main chapters, each focusing on different aspects of the research carried out. The organization of these chapters seeks to provide a logical path, ensuring that readers can follow the progression of ideas.

Chapter 2 introduces some concepts that will be necessary to understand the work developed. It includes concepts related to the processing of video and images to extract data from these media. It also covers different deep learning technologies for identifying and classifying objects and the metrics commonly used to compare them.

Chapter 3 reviews existing literature and solutions comparable to the one developed in this work. Some are equivalent in terms of usability, others in terms of implementation, but the idea is to capture the main performance of the different systems to have a comparative basis.

Chapter 4 details the methodologies used throughout this work and describes the experimental setup, including the dataset's characteristics, its creation, model implementation, and training techniques.

On chapter 5 details the outcomes of the experiments, dataset analysis, clustering techniques, and the creation of synthetic datasets are covered in this chapter. Future research based on the findings is also explored.

Finally, chapter 6 provides a brief summary of everything that was explored throughout the report and concludes with the implications for future work.

Chapter 2

Background

The remarkable progress in the fields of machine learning and video processing has been largely driven by advanced clustering and computer vision algorithms. Algorithms like DB-SCAN, K-Means, Support Vector Machines, and Gaussian Mixture Models play a crucial role in unsupervised learning by uncovering inherent patterns in data. On the other hand, computer vision has evolved into a vital component of artificial intelligence, enabling the understanding and interpretation of visual data. Deep learning techniques, particularly transfer learning and continual learning, have revolutionized the approach to complex tasks in this domain. These models' robustness is strengthened by the use of diverse datasets, including augmented and unlabeled scenarios. The path made in recognizing and categorizing visual elements is evident in object detection techniques like Single-Shot Detector (SSD), Faster R-CNN, and YOLO, as well as image classification frameworks such as AlexNet, ResNet, DenseNet, and EfficientNet. Additionally, image segmentation methods like U-Net and Mask R-CNN enhance the accuracy of object detection in images and allow for a precise separation between the wanted object and the rest of the image. By employing specific metrics tailored for object detection, object classification, and clustering, the efficacy of these methods is meticulously evaluated, ensuring a comprehensive assessment of model performance. This chapter comprehensively explores these foundational concepts, laying the groundwork for an in-depth examination of their implications and applications in advancing computer vision and machine learning.

2.1 Video Processing

Due to the advent of smartphones, digital video is currently one of the most prevalent types of media in everyday life [13]. From them, it is possible to obtain a large amount of information

in a short period of time. To obtain such information, it is important to analyze it, process it appropriately, and improve it within the limitations of the digital environment. There are multiple ways to process these videos. Through adjustments of contrast, brightness, color balance, and noise reduction, which substantially facilitate their analysis. The introduction of forms of video stabilization also helps in situations where the camera is not static, as it reduces shakes through the detection of motion patterns. One of the main limitations of video, however, introduces a major challenge. Since it captures several images per second, this type of media acquires a lot of inadequate information to analyze and work with. Despite being indifferent to the human eye when viewing videos, the blur present in image frames introduces a major obstacle to obtaining reliable information.

Blur detection in videos can help restore specific frames or help extract keyframes. Traditional methods often rely on analyzing gradients. Since sharp edges cause a significant reduction in the gradient's magnitude, their identification helps detect blur. Another approach utilizes the Laplacian operator, which detects rapid changes in image intensity. Additionally, frequency or contrast analysis can reveal areas with blur. This analysis can be performed using the Fourier transform or Wavelet transforms. These transforms represent signals using small localized waves in both time and frequency domains. These waves are created by decomposing the original signal, and at each phase, they produce coefficients that encode information about the amplitude. At each stage, they generate coefficients that encode information about the signal's components' amplitude and phase. Finally, image auto-correlation can also be used. In image auto-correlation, a decrease in the correlation value indicates the presence of blur [13].

The method proposed in [13] uses discrete cosine transform (DCT) and motion vectors to analyze DCT coefficients and track moving objects for efficient blur detection without the need for complex calculations.

This method begins by analyzing the relationship between coefficients and edge sharpness, identifying key characteristics of DCT coefficients. When used to follow objects, motion vectors also help detect partial blur. The entire detection process falls within the DCT domain, to guarantee maximum computational efficiency.

The experimental results of the study demonstrated the efficiency of using this method in different video feeds, including regular and sports programs. The system successfully detected partial and global blur, surpassing other traditional methods.

In 2022, Senapati et al. [14] developed another system already based on Deep Learning. The author's objective was to overcome some more limitations that come from traditional systems due to different types of blur, failures in edge detection, complex image features, and real-time performance.

The method in this work uses a two-stage process through the use of YOLO, an object detection network, and a personalized convolutional neural network. The first stage detects the largest object in the scene, while the second stage identifies whether the blur belongs to the entire image or just the previously identified object. This work proved to be relatively effective, achieving a blur detection efficiency of 73%. Some limitations were identified due to the difficulties introduced by different types of blur and limitations inherent to YOLO, such as the fact that it only detects 80 objects. This limitation makes the entire frame's processing mandatory, increasing the computational cost. The dependence on high image resolution and the system's high computational cost were also reinforced as one of the system's limitations.

2.2 Machine Learning

In Artificial Intelligence, there is a subfield called Machine Learning. As specified by Gupta and Sehgal, "Machine Learning (ML) is the ability of a system to adapt to new data" [15].

The idea of ML is to create algorithms capable of being trained and learning independently to predict the data for which they were prepared. These algorithms are made up of tunable parameters that are automatically adjusted [15].

One of the essential elements in ML is the dataset. This can contain information of different types, such as numbers, text, or images, and can have a label assigned to each value. These datasets are then used to train the developed model, where the algorithm automatically adjusts the model parameters at each iteration. Regarding training, there are two types [16]:

- **Unsupervised Learning** where training occurs mainly through matching a pattern or some relationship between data. This training method uses unlabeled data.
- **Supervised Learning** depends on labeled data, which is used in the training phase to prepare the model for new data that may be introduced and never seen before.

After the model is trained, it can then be used on data that was not used for training and thus predict the result. From the training and testing results, it is possible to obtain some metrics such as accuracy, f1-score, and precision. A very good tool for evaluating the performance of a classification system and giving an overview of it is the confusion matrix. This is a square matrix in which, in each cell, you can compare any of the previously specified parameters between all or part of the test data used and the actual expected result [15].

There are also other types of machine learning, such as reinforcement learning. In this case, models evolve through their interaction with the environment in which they interact. Through appropriate or incorrect behavior in the environment in which it operates, the model receives rewards or punishments respectively. This allows it to evolve over time.

2.3 Clustering Algorithms

One of the best-known unsupervised learning methods is clustering techniques. A cluster is understood based on some rules, which, despite being defined, have not yet reached common sense. Objects in a cluster should be as similar as possible, and objects in different clusters should be as different as possible. The measure of similarity and non-similarity must be clear and have a practical meaning [17].

As explained by Xu et al. [17], the process starts by collecting, selecting, and processing the features to be used. After this, the clustering algorithm is developed and trained with the collected data to obtain then suitable metrics to compare different runs and with other models.

Several distances can be used to calculate clusters. Some of the most common are the Standardized Euclidean Distance and the Mahalanobis distance, both of which can be calculated by the equations 2.1 and 2.2, respectively.

$$\left(\sum_d^{l=1} \left| \frac{x_{il} - x_{jl}}{s_l} \right|^2 \right)^{\frac{1}{2}} \quad (2.1)$$

$$\sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)} \quad (2.2)$$

In equation 2.1, x_{il} and x_{jl} are the l -th features of the i -th and j -th data points, respectively, s_l is the standard deviation of the l -th feature, and d is the total number of features. In equation 2.2, x_i and x_j are vectors representing the i -th and j -th data points, respectively, and S is the covariance matrix of the data.

The following image shows how some of the models that will be discussed behave when it comes to clustering points.

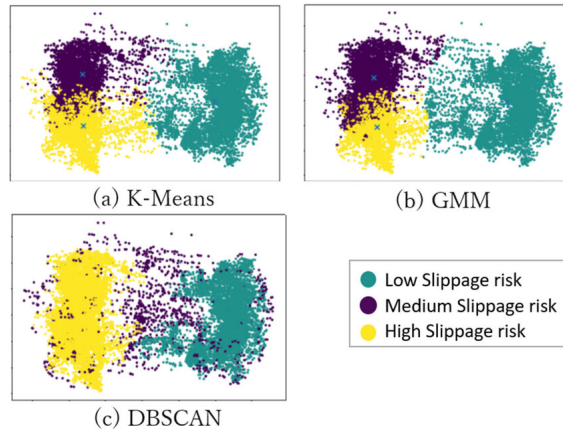


Figure 2.1: Clustering Models [1]

2.3.1 K-Means

Within clustering algorithms, K-means is considered a partitional clustering algorithm. This type of algorithm seeks to minimize the sum of the square error by analyzing the minimum of this value between the different points in the dataset and the average of the points already belonging to the clusters [18]. After obtaining these values, assign the point under analysis to the closest cluster. [18] Having the algorithm k clusters $C = c_j$ where $j = 1, 2, \dots, k$ and, x_i is the i -th data point, μ_k is the centroid of the k -th cluster, C_k is the k -th cluster, and $\|\cdot\|$ denotes the Euclidean distance, the algorithm seeks to minimize the equation 2.3.

$$J(C) = \sum_{k=1}^k \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 [18] \quad (2.3)$$

The basic operation of the k-means algorithm begins by defining a certain k , which corresponds to the number of centroids to which the dataset will be assigned. Afterwards, the distance from all points to the clusters is calculated, and the points are assigned to the one closest. After that, the new values of the cluster centers are calculated. These last two steps are repeated until stability is achieved in the evolution of the cluster centers [18].

Generally, k-means uses Euclidean distance to calculate the distance, and outliers introduced by noise significantly alter the results [18]. Although the temporal complexity of the algorithm depends on the amount of input data, more recent studies argue that the temporal complexity can be described by $O(nkl)$, where n is the number of data points, k is the number of clusters and l is the number of iterations [19].

2.3.2 DBSCAN

Another algorithm is DBSCAN, which, although very popular and versatile, has some difficulties with big data and a low execution speed. [20] This algorithm has the ability to find any shape class in any dataset, requiring the user to indicate the "global constant parameter neighborhood distance and threshold" in order to run. [21]

The algorithm works as follows [21]:

- A value is set for ϵ (epsilon), which defines the radius within which neighboring points are sought, and minPts, which specifies the minimum number of points required to form a dense region;
- The algorithm then calculates the distance between each point and all others. If the distance between two points is less than or equal to the minimum radius (ϵ), it adds that point to a temporary array;

- If the number of points in this temporary array exceeds minPts, those points' neighborhoods are classified as core points, initiating the formation of clusters;
- Finally, the algorithm evaluates the core points of the newly formed cluster. If these core points are reachable from core points in more than one cluster, those clusters are joined together to form a new cluster.

In 2021, Gholizadeh et al. [20] developed a new clustering system, intended to be an evolution of DBSCAN, called K-DBSCAN. This system begins by grouping the data using the K-means++ algorithm. K-means++ seeks to overcome the shortcomings of the original K-means, in which the results depended to a large extent on the initial points. K-means++ iteratively selects centroids based on the distance to existing centroids. This change allows for more robust results and faster convergence. After grouping the data, the K-DBSCAN system is run for each group of data created in the previous step, individually and joining the clusters as necessary.

2.3.3 Gaussian Mixture Models (GMM)

The Gaussian Mixture Model (GMM) is a probabilistic model that can be used for data clustering and is defined as the weighted sum of M component Gaussian densities, as indicated by the following equation. [22] [23]

$$p(x|\lambda) = \sum_{i=1}^M w_i g(x|\mu_i \Sigma_i) \quad (2.4)$$

Where $g(x|\mu_i \Sigma_i)$ is the i^{th} component of the mixture defined by the parameters: mean μ_i and covariance matrix Σ_i . [23]

Once the above parameters have been initialized, the Expectation-Maximisation (EM) algorithm is run. The aim of this algorithm is to maximize the log-likelihood function in relation to the parameters of the 2.4 equation. [23] This algorithm begins with the expectation step, in which it calculates the posterior probability of each point belonging to each Gaussian component. This calculation is followed by the maximisation step, which updates the parameters of the equation 2.4 for each Gaussian distribution in order to maximise the likelihood of the data based on the probabilities calculated in the previous step. This EM algorithm is repeated until the log-likelihood has converged and based on the following criterion: $||\log[p(X|\Theta_{t+1})] - \log[p(X|\Theta_t)]| < \epsilon$, where epsilon is a previously defined threshold [23].

Once the model has been trained, it can be used to define which cluster, i.e. which Gaussian component, the new data belongs to. Once the model has been trained, it can be used to define

which cluster, the Gaussian component, the new data belongs to. One of the great advantages of GMM is its ability to form smooth approximations of arbitrarily shaped densities [22].

2.3.4 Support Vector Machines

Support vector machines (SVMs) were introduced in the 1990s by Vapnik and his colleagues [24]. This new algorithm has found its importance in various areas, such as handwriting recognition, classification, and clustering. The method was originally developed, and the simplest is called the Maximal Margin Classifier. It takes the basic concept of SVMs: they find hyperplanes in space to separate the data and maximize the margin between classes. If the data is not separable, the model tolerates misclassifications by assigning a penalty to ensure more robust classifications [24]. SVMs can also be used for non-linear data through kernel functions, which map the data onto the higher-dimensional feature space [24].

The main types of kernels and respective equations are as follows [25]:

- **Linear Kernel:**

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

- **Polynomial Kernel:**

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$$

- **Radial Basis Function (RBF) Kernel:**

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

- **Sigmoid Kernel:**

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + c)$$

SVMs try to draw as many lines as possible that completely separate the desired classes and then try to find the best one. According to Guenther et al., a suitable choice would be to choose a line that separates the edge and where the minimum distance between the line and the nearest point is maximized [26]. This set is called support vectors. However, the classes are not completely separable in most cases, so a certain amount of error is allowed. In the base case, where no kernel is used and only two classes are considered, the case can be formulated as follows [26]:

$$\begin{aligned}
& \underset{\beta_{ij}, \varepsilon_i, i=1, \dots, n; j=1, \dots, p}{\text{maximize}} && M \\
& \text{subject to} && \begin{cases} \sum_{j=1}^p \beta_j^2 = 1 \\ y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i) \\ \varepsilon_i \geq 0; \sum_{i=1}^n \varepsilon_i \leq C \end{cases}
\end{aligned} \tag{2.5}$$

SVMs can also be used as a clustering tool. If you only use one class, you can perform SVM-Internal Clustering, in which the algorithm tries to find the smallest sphere in the high-dimensional space. Another alternative is SVM-External Clustering, in which an SVM is trained with a small amount of data organized by another clustering algorithm, such as K-Means or DBSCAN. [27] From this data, the model starts trying to classify the unlabeled data and during this process, it tries to refine the decision boundaries it has already learned. This phase is iterated until it reaches convergence, improving the results by re-labeling misclassified vectors.

2.4 Computer Vision

Computer vision inserts itself in machine learning and is an area of research that has been expanding in recent years and increasingly has applications that can be encountered in everyday life.

The basic concept of computer vision is to create algorithms that can draw conclusions through pre-recorded images or images captured in real-time, depending on the context. Generally, prior processing is carried out on the image to improve its characteristics through changes in brightness, contrast, size, and color space, among others. Only after this step are the identification algorithms applied. These algorithms are very vast and have different stages, as stated by Wei et al. [11]: (1) Image capturing and collection. (2) Image pre-processing to enhance the conditions of the image itself and make the work easier for the next stage; as well as defining the regions of interest (ROI), the sections of the image where there might be products to analyze. (3) Feature extraction to define what are the most distinguishable and unique details of each particular image. (4) Feature classification where the algorithm analyses each feature and takes a conclusion based on specific decision rules. (5) In the output of the recognition, the algorithm predicts the category of the observed product.

There are several applications regarding this topic, but the main ones focus on object detection and image classification.

2.4.1 Deep Learning Methods

With the need to create more advanced methodologies, methods that use deep learning models began to be developed.

Convolutional neural networks are a type of deep learning model commonly used to process images. The main idea is to have a system that mimics the functionality of a brain, specifically the cat's visual cortex [11].

These networks are made up of multiple layers. The initial group of layers is constituted by convolutional layers, which perform a convolution operation to detect spatial patterns like edges or specific features. Following one of these convolutional layers can be a pooling layer, which reduces the spatial dimensions of the input and reduces over-fitting.

After some of these first layers, the output is connected to the fully connected layers, which are similar to the layers used in the commonly used fully connected neural networks and are made up of neurons, which are the base elements of these layers, corresponding to neurons in a biological brain. These neurons perform a weighted sum of the inputs they receive. The result of this average is then passed through an activation function. This function adds non-linearity to the network, allowing it to learn complex patterns. The output of this function is then connected to the next layer.

Backpropagation is used to train these networks. This is a process that updates the value of the neuron weights depending on the error calculated at the last layer after executing the neural network and taking into account the values of current weights.

With these models trained, it's possible to improve the results obtained with computer vision, since the model automatically tunes the parameter of these being humanly chosen.

The advantages of using a CNN instead of a common neural network are multiple, particularly for a network dedicated to working with images. In CNN's, the use of the convolution operation allows analyzing relationships between features, performing an area analysis; something that does not happen in normal networks, which deal with each feature individually. The use of convolution also allows the reduction of noise and variations that may be introduced by the camera and the capture environment. CNNs have a degree of invariance due to the use of shared weights and pooling layers. This allows CNNs to track features, even if they do not have a fixed position. Convolutional networks are also recognized for their great efficiency in analyzing data with a matrix-like structure, as is the case with images.

2.4.1.1 Transfer Learning

Since training a neural network from scratch can be very time-consuming, a good technique to apply is "Transfer Learning".

This technique consists of using models already trained for certain contexts and adapting them to the desired context. This is done by freezing some layers in order to retain the weights and biases of those layers while training the network with the new dataset. According to Guimarães et al [12], this is a great solution for cases where the dataset is small and insufficient to train a model from scratch. And Wei [11] also affirms the advantage of this method since it can apply already obtained knowledge in a fully different environment to the desired environment. This could potentially lead to better results.

2.4.1.2 Continuous Learning

As described previously, in retail, there is always a constant renewal of products and new ranges are always being introduced. Considering this, it is important to find solutions that guarantee that the neural network can learn and adapt to these changes in the dataset. Unfortunately, common training is not enough, as one of the most common problems arising from it is catastrophic forgetting. This problem is common, as a model's new ability to handle new results diminishes its ability to handle old ones. It is important to try to balance learning plasticity and memory stability [28].

The first solution could be to keep all the old training samples, which introduces large memory restrictions and computing overheads. So, different approaches have been developed. Wang et. al [28] describe them. Here are some examples:

- **Replaying** old training samples is a way to improve the performance of old tasks, but this option carries a great risk of creating resource overhead. This may require the need to save past data;
- There is the **regularization-based** approach that creates a copy of the old model in order to regulate parameter changes in the new learning phase of the new task. One of the main methods is Elastic Weight Consolidation. This method uses the following loss function:

$$\mathcal{L}_{\text{EWC}}(\theta) = \ell_k(\theta) + \frac{\lambda}{2}(\theta - \mu_{k-1})^T \hat{F}_{1:k-1}(\theta - \mu_{k-1}), \quad (2.6)$$

Which penalizes deviations from the current parameters (θ) to those of the previous model (μ_{k-1}). $\ell_k(\theta)$ is the task-specific loss, and $\hat{F}_{1:k-1}$ is the Fisher Information Matrix that quantifies how much information about the parameters is contained in the likelihood of the observed data. This equation also introduces the λ hyperparameter that controls the strength of the regularization;

- Another method is with a model that has a **dynamic architecture** that is able to expand itself if its capacity is at the limit and it needs to learn a new task;
- **Meta-Learning** or learning-to-learn is an approach that seeks to teach the algorithm to analyze different characteristics or patterns of incoming data. Through this, they can change their own biases. The general idea is that the model learns a learning algorithm that allows it to quickly adapt to new tasks.

2.4.2 Datasets

Datasets play a fundamental role in computer Vision. They are a set of images with corresponding labels that is used for training, validation and testing of the convolutional neural networks. Its collection in quantity and quality is crucial to obtaining good results both in terms of training and in terms of results when it comes to computer vision.

Unfortunately, most datasets are created through human labor, which makes them limited and excessively difficult to expand [29]. The creation of datasets must take into account the image capture conditions, such as light, contrast, saturation, noise, or background where the product is located. Particular care should be taken with the latter, as it can help identify the product completely without mixing it with other objects, facilitating the training process.

2.4.2.1 Augmentation

To overcome this lack of material in datasets and save work, augmentation is used. This is a process that is commonly done when dealing with datasets with a low quantity of training data [11]. It employs multiple techniques, some of which are semi-automatic. These ones can improve existing images or modify them within certain limits, but unfortunately, this also creates redundant images. [29]

In [30], some semi-automatic techniques are proposed:

- **Removing-and-Inpainting:** This technique removes objects or parts of them from the images and fills this area with content that makes this change appear as natural as possible;
- **Cut-and-Paste:** This algorithm cuts the object that is supposed to be identified and pastes it into several images to create images with different backgrounds;
- **Image Variation:** This method applies several transformations to the original image, such as brightness, contrast, color space, contours, color temperature, or noise;

On the other hand, automatic systems use generative methods, which synthesize images through models designed on a computer. An example of this is the model developed by Tang et

al. [2], which synthesizes images from CAD models. Importing the CAD models into a program capable of handling the models, such as Unity3D, where you can capture photographs of the model from different angles and with different light conditions or backgrounds. However, this method can be applied without the need to have a CAD drawing, modeling the object directly on the chosen program.

Both semi and fully-automatic systems are also particularly important for balancing classes, which is a recurring problem in computer vision and reduces the effectiveness of model training.

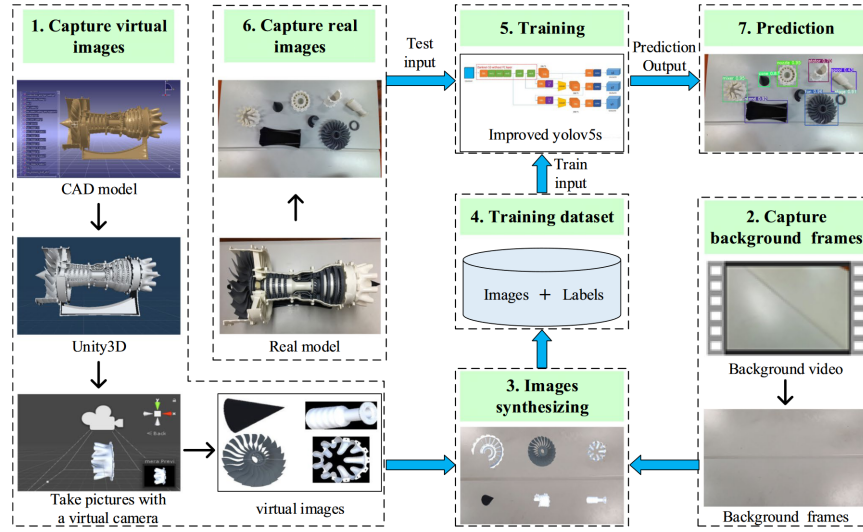


Figure 2.2: Proposed system for image synthesis by Tang et al. [2]

2.4.2.2 Unlabeled Cases

Sometimes, it is even possible to make a large collection of photographic material using existing datasets or material on the internet, but these resources are not labeled, which does not allow the training of supervised models [31]. This work can be done manually, but for large quantities of images, the labor cost is high.

There are two main methods to help with this process [31]:

- **Active Learning**, which tries to classify images automatically but uses human input in cases that raise doubts, using this extra information for subsequent classifications. Unfortunately, it introduces some delay since the process is iterative, and it is necessary to pass some initial information to the classifier;

- And **Group-Based Learning** that aggregates all similar images, facilitating the labeling process. This method may be less effective with data that has more than one label per image.

2.4.3 Object Detection

As previously specified, given the large amount of information captured in an image, it is important to detect the content, in this case, the desired product, before attempting any classification.

There are several traditional algorithms, with the sliding window implementation being the most common. This method used a window that scrolled through the image and looked for the object in that window, making this method highly inefficient and consuming computational resources [11]. These methods are not so relevant to this research since the objective is to focus on an alternative based on deep learning and are. Therefore, the latter ones those that will be explored in more depth.

"CNN's have been the major deep learning technique for object detection" [11]. And since they are so efficient nowadays, all the object detection deep learning algorithms that have been developed are based on CNN's. The object detection methods can be divided in two categories [11]:

- **One Stage Models:** These models are able to find all the objects in one pass;
- **Two Stage Models:** These models first identify the regions where the object might be located, usually via features like textures, edges, or colors. And only after that is defined the model detects the object.

Although the one-stage models are faster [10], the two-stage models are more accurate. So there are going to be presented both types of models, but the more important are going to be the one-stage models, since in this dissertation the main goal is to develop a real-time system as previously stated.

2.4.3.1 Single-Shot Detector (SSD)

The Single-Shot detector is a one-stage detection model based on a convolutional neural network. This type of detection method has a high accuracy even on low-resolution input images while being able to detect both large and small objects using features of different depths. [10]

Although it is a one stage network, it can be partitioned into two components, the first one is a pre-trained convolutional neural network, usually based on architectures like ResNet or a

VGG-16, and the second part is referred as an SSD Head. The SSD Head is composed by some additional convolutional layers that give bounding boxes and classes of objects as outputs. In its core function, SSD starts by passing the image through the first neural network component, the CNN, which extracts the features at multiple scales. After that, the CNN produces feature maps with different scales that capture information about the objects. Following this, the system defines default bounding boxes, termed Anchor Boxes, that generate scores for all classes contained in the box (confidence scores). That is, it generates a value representative of the likelihood score of a certain object class within each anchor box [3].

During the training step, the predictions generated by the SSD, can be separated into positive and negative matches. The factor that decides this is the localization cost. If the misalignment of the calculated box is greater than 0.5 of the ground truth, is positive, otherwise, is negative. Since the majority of the cases are classified as negatives, is necessary to balance the cases. So, hard negative mining is applied. This process organizes the results by highest confidence loss for each default box, in order to guarantee a more optimized and effective training. The ratio of positives and negatives mustn't be greater than 3:1 [3]. The final results are obtained by applying non-maximum suppression. This process eliminates redundant and low-confidence detections, keeping only the most reliable results.

The main appeal of SSD is the faster speed when compared with other object detection models as YOLO or Faster R-CNN [3], which makes it appealing for a real-time system like the one that this dissertation aims to develop.

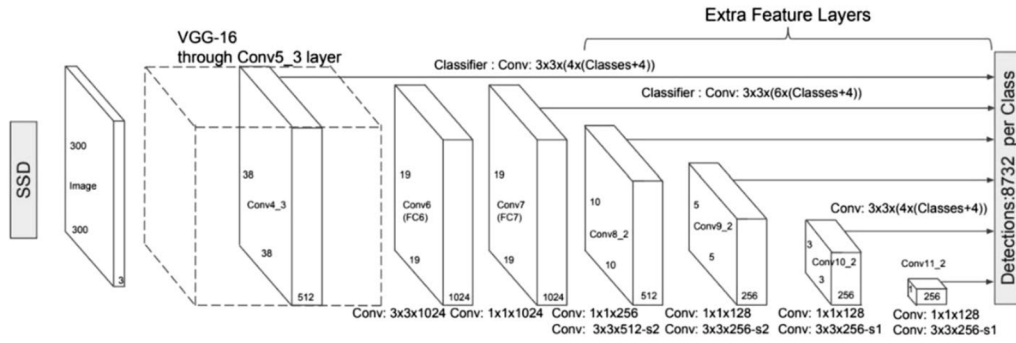


Figure 2.3: SSD Network Structure. [3]

2.4.3.2 Faster R-CNN

Starting with the Region-Based Convolutional Neural Network (R-CNN), which was the first network of this type developed, the main operating principle is based on the proposal of regions. This type of network is made up of two stages. The first one selects areas of the image

and proposes a set of bounding boxes that can contain objects. For this objective, the model uses the Selective Search Algorithm that segments the image into superpixels, which are groups of pixels with similar characteristics, such as color and texture. After that, the superpixels are joined together using a calculated similarity index, creating regions. This algorithm returns the bounding boxes of the candidate regions to be analyzed, which are passed to a CNN, which then warps the images to a fixed size and begins by extracting the features [3]. In this network, the classification is carried out by a Support Vector Machine, which helps to determine whether a certain object exists in a given region.

Unfortunately, this method has always been very slow, given the large number of proposed regions. To overcome this gap, Fast R-CNN emerged, which passes the image through a convolutional layer before proposing the regions, which produces a convolutional map of features that is then classified with the help of a SoftMax layer. This reduces the number of proposed regions, which improves model performance. However, since the speed remained below expectations, the need to improve this model remained, which is why Faster R-CNN was developed. Although, like Fast R-CNN, a convolutional map is obtained, this model replaces the Selective Search Algorithm with a network that is responsible for making region proposals, maintaining the same classification method. These changes introduced such significant improvements that the system could even be used in real-time [3].

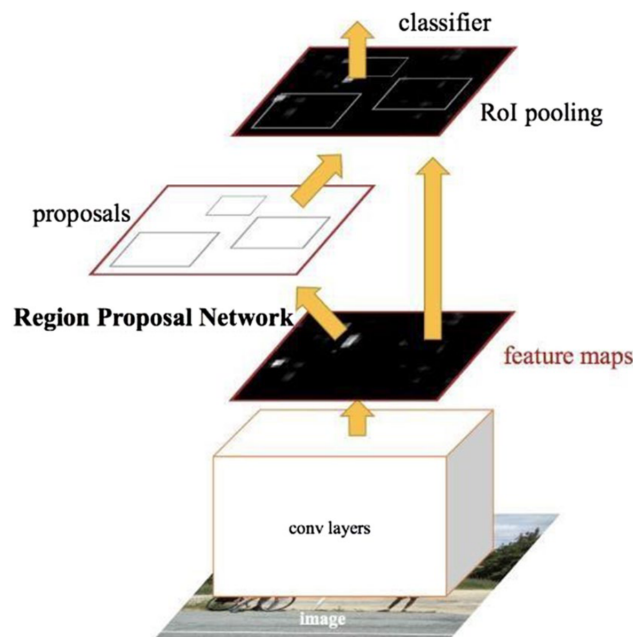


Figure 2.4: Faster R-CNN Network Structure. [3]

2.4.3.3 YOLO

The backbone for YOLO is Darknet. This is an open-source neural network framework which allows the creation of networks that support both CPU and GPU. Right from the start, the original image is divided into a grid and the algorithm predicts multiple bounding boxes. Each of them is assigned five parameters: the x and y coordinates (bx and by), the width and height (bw and bh) dimensions of the box and the confidence score (σ) that defines the probability of this box containing an object [32] [3].

$$bx = \sigma(t_x) + c_x \quad (2.7)$$

$$by = \sigma(t_y) + c_y \quad (2.8)$$

$$bw = p_w \cdot e^{tw} \quad (2.9)$$

$$bh = p_h \cdot e^{th} \quad (2.10)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.11)$$

In addition to this score, YOLO defines each class's probability for each predefined object using an activation equation. This activation function depends on the YOLO version and each use case. For example, as stated in the original paper for the v3 YOLO version, it doesn't use the softmax function since it didn't show off any advantage, so they settled on independent logistic classifiers [32].

After everything previously mentioned is defined, the bounding boxes are filtered based on their confidence score, with all those that have a score below a defined threshold being excluded. It also removes all redundant boxes. To do this, the algorithm calculates the intersection-over-union (also used as an evaluation metric as can be seen in section 2.5), which is a ratio between the area of intersection of two bounding boxes and the area of union, and then removes the box with the highest ratio, always taking into account the confidence score.

$$IoU(A, B) = \frac{\text{Area Of Intersection}(A, B)}{\text{Area Of Union}(A, B)} = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)} \quad (2.12)$$

When all boxes are selected and filtered, the algorithm passes the bounding boxes to a CNN to extract features and classify the system. This CNN also depends on the YOLO version. In

YOLOv3 it was used Darknet-53 [3][32].

2.4.4 Image Classification

After detecting the image, the next step is to find the relevant features of the object and identify it. There are several methods of carrying out this procedure. There are some traditional methods such as Scale Invariant Feature Transform (SIFT) or Oriented Fast and Rotated BRIEF (ORB) detectors, but more recent alternatives have relied on convolutional neural networks (CNN) such as AlexNet, ResNet or DenseNet.

The CNN's are usually separated into two parts. At the network's first levels, the objects' most important details are highlighted, and the most irrelevant are suppressed. And a second part that analyzes the selected features. [4]

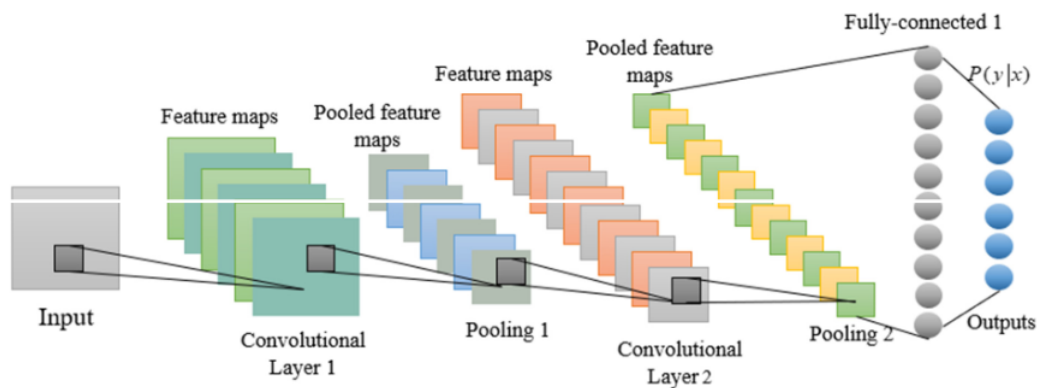


Figure 2.5: Basic Classification Network Structure. [4]

2.4.4.1 Traditional Pattern Recognition Methods

Images can be filtered so that certain parts are highlighted. These features are generally detected as "corners, blobs, edges, junctions, lines, etc." [33]. From the extraction and analysis of these features, it is possible to compare and match them with other previously selected features to confirm a given object's classification.

Introduced by D. G. Lowe, SIFT [33], is one of the systems that implement this type of classification. Uses the Difference of Gaussians operator (DoG), which its result is obtained by subtracting a blurred version of an image and another version, both of which were produced with recurring to a Gaussian filter with a different standard deviation. The resultant of the operation highlights the maximum and minimum values of the image, extracting the keypoints. After that, these keypoints are located with greater precision through the use of a threshold value and

finally, they are assigned an orientation so that they are invariant to the image rotation. [34] As indicated in the name, the algorithm is scale-invariant, but also invariant to rotation and affines [33]. Another algorithm is Speeded Up Robust Features (SURF), introduced later, but with an operation very similar to SIFT. This algorithm is based on the determinant of the Hessian Matrix [33]. It is used to calculate the local minima and maxima that extract the features. These changes optimize the algorithm on a large scale, making it faster and more efficient than SIFT.

More recently, other methods that have been developed and highlighted are (Binary Robust Invariant Scalable Key) BRISK, and Oriented Fast and Rotated BRIEF (ORB). BRISK detects corners by searching for maxima in a scaled-normalized version of the image at each level of a scale space pyramid. Its descriptor compares the intensity values of keypoints to generate a binary representation, which is used to match other images [10]. On the other hand, ORB utilizes Features from Accelerated Test (FAST), Harris corner detector and BRIEF keypoint descriptor to extract the key points; resulting in fewer but more significant features [34].

This type of method kept evolving, with the ORB, BRISK, and AKAZE being among the most advanced currently. Unfortunately, the traditional pattern recognition methods are very weak by themselves and are unable to obtain good enough results for application in a real scenario since they use manually designed features rather than learning new features directly [11].

2.4.4.2 AlexNet

This type of CNN was developed in 2012 with image classification in mind. In its time, CNN was a major advance in the field, surpassing the most used neural network. This was the network that caused interest in deep learning to grow rapidly [35].

The first five layers of this network are convolutional. The network starts by performing convolution operations in the first layer, with an 11x11 filter, reducing the size of this filter to 5x5 in the second layer and to 3x3 in the remaining three. After the convolutions, Local Response Normalization is applied in the first two layers, which aims to normalize the responses across adjacent channels. At the end of each of the convolutional layers, a ReLu and max pooling operation is also performed with 3x3 filters [35]. ReLu is added to increase the non-linearity of the model, replacing negative values with zero and max-pooling is performed to downsample the spatial dimensions of the image and feature maps.

After being selected in the convolutional layers, the features are passed to the fully connected layers. This network contains two, and they are used with dropouts. Dropout helps reduce overfitting by dropping a random number of neurons during training. The network is finished with a Soft-Max layer that converts the network output into probability scores for each class [35].

After AlexNet, other similar networks were launched, such as VggNet. These maintain the base structure of AlexNet, but adding some layers, modifying the size of the filters, changing the application of local response normalization, batch normalization or ReLu...

2.4.4.3 ResNet

One of the biggest problems with neural networks is their train time. In order to combat this problem, ResNet was developed. This is a neural network, which, by skipping two or three layers containing ReLu or batch normalization, managed to maintain good performance even though it is deeper and reduces training time [36].

The number of ResNet layers depends on the implemented network and can have 34, 50, 101, 152, or even 1202 [35]. These layers are divided into blocks, called Residual Blocks, so this network is a set of these blocks. These residual blocks can be described by the following equation:

$$y = F(x) + x \quad [35] \quad (2.13)$$

In this equation, the 'y' corresponds to the output. The 'F(x)' corresponds to the previous layer's output that went through common transformations, such as ReLu or Batch Normalization. The 'x' corresponds to the previous layer's output directly, without the values undergoing transformations [35]. This block allows the network to learn the best path to follow, avoiding going through unnecessary layers, and the network can adjust the layers' weights to choose between optimal mapping or identity mapping. This helps combat the vanishing gradient problem that is very common in very deep networks. The ResNet ends with a single Global Average Pooling that computes the average value of all the activations per channel of the feature maps.

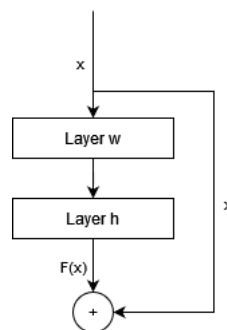


Figure 2.6: Structure of a Residual Block

2.4.4.4 DenseNet

DenseNet takes the same block concept as ResNet and applies it. In this case, the blocks are called Dense Blocks. These can be described by the following equation [35]:

$$x_l = H_l([x_0, x_1, x_2 \cdots x_{l-1}]) \quad (2.14)$$

This equation indicates that the output of a layer l corresponds to the output of a Dense Block (H) in a certain layer l , with this as input the outputs of all previous layers. Dense Blocks perform four consecutive operations: Batch Normalization, Leaky Rectified Linear Unit, a 3×3 convolution and dropout. Between these blocks there is also an additional layer, which performs a 1×1 convolution and a 2×2 max pooling [37] [35].

The main objective of this architecture is to allow all layers to have access to the feature maps of previous layers, which allows the reuse of features and ensures that the model is more compact and less subject to overfitting [37].

This network also has a hyperparameter called Growth Rate. This serves to define the number of new feature maps that the network adds to each layer [38].

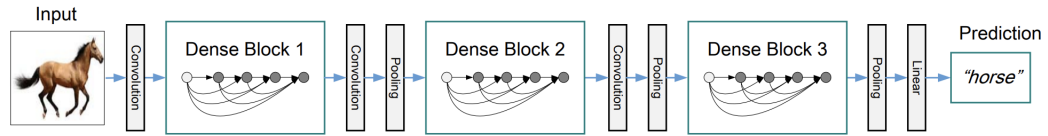


Figure 2.7: DenseNet Network Structure[5]

2.4.4.5 EfficientNet

Researchers are constantly looking for ways to improve the performance of convolutional neural networks. One of the simplest and best-known ways to do this is by scaling the networks. Networks can be scaled in width by increasing the number of nodes per layer or in-depth by increasing the number of layers. There is also a third dimension that can be changed, which is the size of the image. However, the process of scaling convolutional networks has never really been understood [6].

In 2020, tan et al. proposed a simple method called the compound scaling method, which uniformly scales the depth, width, and resolution of the network, as can be seen in image 2.8. The basic idea behind this method is that if there are higher-resolution images, the depth of the network should be increased, just as the network should be widened when the resolution is higher so that more fine-grained patterns can be captured [6].

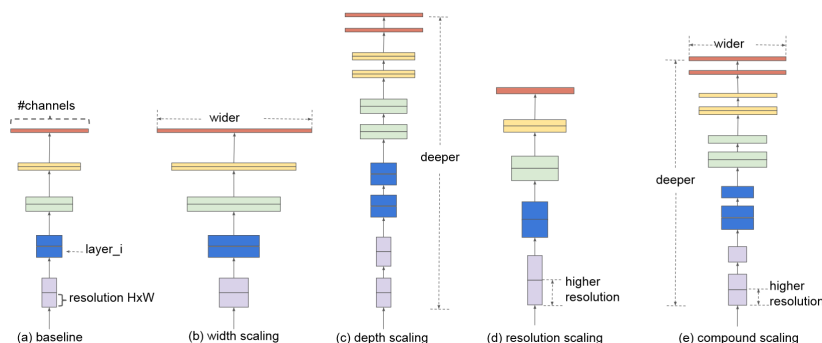


Figure 2.8: Compound Scaling [6]

EfficientNet takes a baseline network, which uses a mobile inverted bottleneck convolution block similar to MobileNet V2, but much wider. This baseline network is optimized for computational efficiency and accuracy. From this network, the model is scaled and thus gives rise to the EfficientNets family. The scaling phase is separated into two steps. In the first step, with the compound coefficient at 1, a grid search is carried out, and the values of the width, depth, and resolution coefficients that optimize the network are sought. In the second step, the compound coefficient is modified in order to scale the network while maintaining a balanced allocation of resources across all dimensions. This gives rise to the widest models, known as EfficientNet-B1 to B7 [6].

2.4.5 Image Segmentation

Semantic segmentation is a computer vision task in which each pixel in an image is assigned a classification. This type of network produces masks for each category, indicating where each category is located in the image. However, it does not distinguish between objects of the same class. Instance segmentation, on the other hand, classifies the pixels in each category and distinguishes between different objects in the same class. In addition to classifying and distinguishing objects, instance segmentation provides information about object boundaries and different object instances in the output.

2.4.5.1 U-Net

The U-net is a specialized image segmentation network primarily used in the biomedical field. It is named after its U-shaped architecture, as can be seen in figure 2.9, consisting of an encoder (contracting path on the left) and a decoder (expansive path on the right) [39].

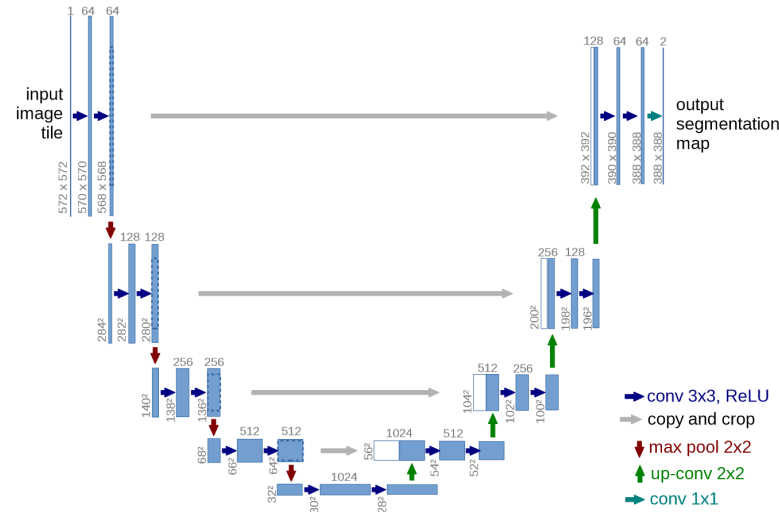


Figure 2.9: U-Net Network Structure

The contracting path follows a typical convolutional network architecture, with convolution layers followed by a non-linear activation function (ReLU) and 2x2 max pooling with stride 2 for downsampling. This path aims to capture the context and features of the input image [39].

Several convolution layers process the feature maps in the most compressed form (middle of the network). These processed maps are then passed to the expansive path, which upsamples the maps to the size of the original image using Up-Convolution [39]. After upsampling, a concatenation is performed between the upsampled maps and those at the same level on the contracting path to maintain both high-resolution and upsampled features. Finally, convolutional layers combine the features and refine the output, reconstructing spatial details and recovering location information lost during downsampling [39].

The U-net efficiently maintains textual and spatial information due to its symmetrical U-shape and the inclusion of skip links. It is a fully convolutional network, avoiding fully-connected layers, and is therefore more efficient for image segmentation tasks [39].

2.4.5.2 Mask R-CNN

Mask R-CNN is an advanced framework explicitly engineered for the task of object instance segmentation. It excels in detecting objects within an image and generating high-quality segmentation masks for each identified object. This method innovatively builds upon the foundational Faster R-CNN by incorporating an additional branch dedicated to predicting an object's mask in parallel with the existing branch for bounding box recognition. This network achieves an operational speed of 5 frames per second [7].

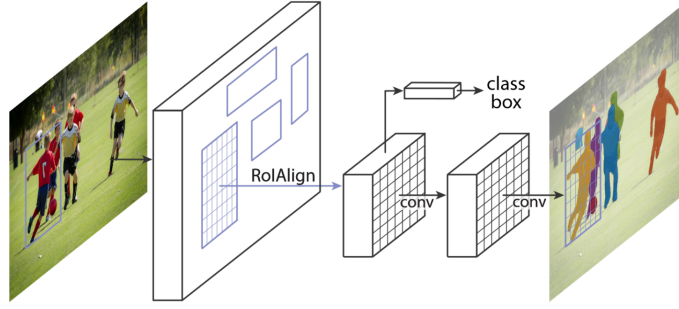


Figure 2.10: Mask R-CNN Framework [7]

Expanding upon the capabilities of Faster R-CNN, Mask R-CNN integrates a branch for the prediction of segmentation masks for each Region of Interest (RoI). This branch constitutes a small Fully Convolutional Network (FCN) that operates on each RoI, predicting segmentation masks pixel-wise. The implementation of this extension necessitated the resolution of the alignment discrepancy between the input and output pixels, a challenge adeptly met by introducing a novel layer known as RoIAlign [7]. RoIAlign aims to improve the alignment accuracy between region of interest (RoI) boundaries and the extracted features from a convolutional neural network. RoIPool quantizes RoI boundaries and feature bins, while RoIAlign preserves precise spatial information without quantization. This seemingly modest modification significantly improves mask accuracy, enhancing it by 10% to 50% [7].

Furthermore, Mask R-CNN employs a multi-task loss function encompassing classification, bounding box, and mask loss. This methodological approach effectively decouples mask and class prediction, enabling the network to independently generate a distinct binary mask for each class [7].

2.5 Comparison Metrics

2.5.1 Clustering Evaluation Metrics

There are different metrics for evaluating clusters. The best known is the **Silhouette Co-efficient (S)**, which measures the results of clustering, pointing to dense and separate clusters [40].

$$S = \frac{d_n - d_w}{\max(d_n, d_w)} \quad (2.15)$$

The equation 2.15 shows how it is calculated. In this equation, d_w is the average distance between a point and all the points in the same cluster. d_n , on the other hand, is the distance between a point and all the other points in the nearest cluster. The value of the coefficient can vary between -1 and 1, with a value of -1 representing a bad cluster and 1 a good clustering with dense and well-separated clusters [40].

To compare the clustering outcome (X) and the ground truth clustering labels (Y), we can use the **Normalized Mutual Information (NMI)** defined in equation 2.16, where $MI()$ is the mutual information and $H(z)$ is the entropy of z . The NMI value has a ceiling of 1, which indicates perfect clustering, and a floor of 0 for incorrect clustering results [40].

$$NMI = \frac{MI(X,Y)}{\text{mean}(H(X),H(Y))} \quad (2.16)$$

The **Rand Index (RI)** measures the similarity between two sets of data clusters.

$$RI = \frac{a+b}{N} \quad (2.17)$$

In the 2.17 equation, a corresponds to the number of true positives and b to the number of true negatives. And N is the total number of points. With this calculated, we have the **Adjusted Rand Index (ARI)**, on equation 2.18, where $\mathbb{E}[\]$ is the expectation value. A value of 1 in the case of the ARI indicates perfect clustering [40].

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]} \quad (2.18)$$

Finally, we have the **Davies-Bouldin Index (DB)** which compares the similarity between each cluster and the next most similar cluster.

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{\delta_i + \delta_j}{\Delta_{ij}} \right) \quad (2.19)$$

In its 2.19 equation, δ indicates the average distance between all the centroid points within each cluster and Δ_{ij} is the distance between the centroids of clusters i and j . A value of 0 in this index indicates that the clusters are dense and well-separated [40].

2.5.2 Object Classification Evaluation Metrics

The performance of deep learning algorithms can be influenced by a variety of factors. Hardware, the foundation for processing, significantly impacts inference and training time. Leveraging high-performance GPUs and TPUs can greatly reduce the time needed to train large models,

enabling the use of increasingly complex architectures and larger datasets. Technological advancements leading to better hardware also contribute to faster models and the potential for real-time usage.

Another influential factor is the size and quality of the dataset. A larger dataset can enhance a model's performance but may extend training time. Conversely, a higher-quality dataset is invaluable for streamlining the training process and enhancing the model's ability to generalize from previously unseen data.

Most algorithms used in computer science need to be described regarding their efficiency, complexity, and hardware requirements.

Regarding their time or space complexity, the big-O notation is usually used. The formal definition for this notation describes the upper bound on the growth rate of an algorithm's time or space complexity. However, in deep learning models, this time complexity needs to encompass other factors that are introduced in the learning nature of the systems. Because of that, these models are usually evaluated in terms of the total time taken to be trained and the inference time, which assesses the time it takes for the trained model to make new predictions. It must be noted, however, that these values are dependent on where the models are trained and run [3].

Regarding their performance, networks can be characterized based on their accuracy and their F1-Score. To do this, it is important to identify the True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN).

In classification tasks, TP is the case in which the model can correctly classify the object; FP is when the model incorrectly classifies the object; and TN is when the model correctly detects that a certain class is not present in the image. In the case of classification, the concept of FN is not usually used. In the case of object detection, a result is considered TP when the center of the bounding box is within the ground-truth box [10]. It is considered FP when the object is detected but is outside this box. If a certain object is not detected but exists, it is considered a False Negative. In the case of detection, the concept of TN is not usually used.

With this in mind, the general equations for accuracy and F1-Score are as follows [41]:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (2.20)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.21)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.22)$$

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.23)$$

During training, it's essential to prevent overfitting. When overfitting occurs, the training accuracy is exceptionally high because the model effectively learns the training data, including noise and outliers, resulting in poor performance with new data. Techniques such as regularization, dropout, or cross-validation are crucial to address overfitting. A validation dataset is valuable for assessing the model's progress during training.

Regarding object detection algorithms, using mAP as a metric is common. mAP stands for Mean Average Precision and is represented by the mean of the average precision values calculated for different classes or object categories [3].

It's also important to keep in mind that while a model invariably demands a lengthy training period, the inference time varies and is pivotal based on the application to which the model will be applied.

2.5.3 Object Detection Evaluation Metrics

In the case of the object detection model, the metrics that are primarily used are somewhat different.

The most common is mAP (Mean Average Precision), which can be calculated using the equation 2.24, where N is the number of classes and AP_i corresponds to the average precision of class i . This is used to calculate the mean of the average precision values calculated for different classes or object categories.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.24)$$

There is also Average Precision, which numerically expresses the precision-recall curve, thus providing a metric that allows you to evaluate performance across all classes [42]. The precision-recall curve mentioned above is a graphical curve that helps to understand the trade-off between precision and recall of the different threshold definitions.

There is yet another very common metric when it comes to detection, called Intersection over Union (IoU). This measures how much the predicted bounding box overlaps with the ground-truth bounding box. As can be seen in the equation 2.25, it is defined as the area that overlaps divided by the area in union of the predicted and ground-truth bounding boxes [42].

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (2.25)$$

A common metric derived from these two is the mAP-50, defined as the Mean Average Precision, but with a fixed IoU of 0.5.

Chapter 3

State of The Art

This chapter delves into the latest advancements and innovative applications of computer vision, focusing on its integration into retail checkout systems. The first section provides a comprehensive overview of recent developments in computer vision technology, highlighting how these breakthroughs are broadening the scope of image and video analysis. Then, an examination of retail checkout systems which have undergone significant changes due to these advancements, is conducted. Starting with simpler technologies such as Barcode and RFID systems and advancing to more complex ones based on visual systems. These visual solutions, which improve accuracy, efficiency, and customer experience, hold the potential to revolutionize the retail industry. This chapter presents a comprehensive study of these cutting-edge technologies, focusing on their potential applications and future trajectories within the retail sector.

3.1 Advancements on Computer Vision

In recent years, the field of computer vision has seen significant progress, going beyond the most common methods and increasingly focusing on deep learning techniques. In 2019, Buhrmester et al. [43] carried out an analysis of explainers that explain how deep learning algorithms work to try to deepen knowledge about them, which are like a black box. This need arose because these algorithms have been criticized for being non-transparent and unfair, using explainers to understand the inner workings. Many of them focus on interpreting DNN predictions, particularly in computer vision tasks. The main innovation of this study is that by allowing algorithms to be understood, they can be developed in a much more conscious, robust way geared towards real progress rather than experimentation.

This area is constantly being integrated into others. Health is one of the main areas in which it is very relevant. Rajam et al. [44], investigate the application of computer vision in the field

of ophthalmology. In this article, they discuss automatic retinal analysis methods, which help ophthalmologists detect abnormalities in the structure of the retina, such as optic disc and blood vessels, or diagnose sight-threatening retinal diseases. The use of the canny edge filter combined with neural networks allows retinal blood vessels to be automatically detected in RGB images. The authors study various methodologies, specifically 6 main categories: (1) matched filtering, (2) vessel tracking, (3) mathematical morphology, (4) multi-scale approaches, (5) model-based approaches, and (6) machine learning methods. One of the first analyzed studies uses a Gaussian mixture model to obtain the clusters corresponding to blood vessels. The main advantage of using this model is that it is unsupervised and, therefore, inherently has a lower computational cost. Another study has opted to use a support vector machine, which is a supervised method. It is reported to be suitable for the task of segmenting elongated structures. A few more papers use those methodologies, but inevitably, the latest systems begin to rely on convolutional neural networks and deep neural networks, which can guarantee much more accurate results but at a higher computational cost.

Another area is video surveillance. Improving human security through video surveillance requires methodologies capable of detecting abnormal situations without human interaction. In 2020, Dubey et al. [45] proposed a system consisting of a 3D ResNet, the deep MIL method, and a new proposed ranking loss. The 3D ResNet is used for feature extraction using a pre-trained network with the Kinetics dataset. The videos that were passed to the network were divided into 32 non-overlapping videos segmented in each frame, resized in 112 by 112 pixels, and with a fixed frame rate of 30 fps. The deep MIL method refers to Deep Multiple Instance Learning, a form of supervised learning in which a bag of instances is used instead of a single one. In this method, the task is to classify bags with characteristics common to the instances they contain. The need to create a new loss function arose from the lack of instance-level labeling. This new function also considers possible false alarm cases; since it is designed for the purpose, it guarantees more effective learning.

Computer vision is an area where innovation is constantly being sought. A 2020 comparative study [46] compared different datasets to train three popular CNN architectures: AlexNet, GoogLeNet, and VGG16. The idea was to evaluate scene recognition capabilities using deep learning methods. In one of the articles analyzed, the problem of dataset biases resulting from different scaling conditions was solved by using a multi-scale model, which combines multiple specialized CNNs to increase performance in various image recognition tasks. One of the first conclusions they also drew was that CNNs perform well in recognizing single images and, therefore, have much potential to generalize to video data. They also highlight the importance of a study in which a ResNet pre-trained with the ImageNet dataset, called T-ResNet, obtained better results on the YUP++ dataset than a normal ResNet. T-ResNet uses transfer learning to trans-

form the spatial-centric residual architecture into a spatiotemporal-centric network. With all the research that came out of the analysis, the authors concluded that the main point to consider is the data being analyzed. It should be noted that scene recognition tasks cannot be generalized to all cases, due to the nature of the scene images and the scene categories overlap.

In [47], the authors present their work on the use of multivariate Gaussian distribution (MBG) to fit normal data in pre-trained deep feature representations, particularly using transfer learning. They use the Mahalanobis distance as an anomaly score. In this study, EfficientNet models are implemented, given their high performance in classifying the ImageNet dataset. The method developed through data training with augmentation revealed superior performance in anomaly detection. The authors concluded that these features are difficult to learn from scratch using normal data only and propose using feature representations generated by large-scale discriminative training in a transfer learning setting instead.

3.2 Checkout Retail Systems

As indicated before, this is an area in which there has been constant and recent investment, so multiple works are seeking a solution, already developed and some applied. Some authors opted for alternatives that focus only on a terminal at the store exit, others focused on solutions that analyze the products on the shelves, and others have already tried to implement systems similar to what this work proposes.

3.2.1 Barcode and RFID Systems

The objective of modernizing the shopping method started some time ago and is still present in our quotidian to this day. The barcode system uses a code formatted in an identifier composed of bars of specific lengths and spaces. It's a simple and accurate system, but it has some downsides since it usually checks the information on a cloud-based database instead of using one stored in the device, which requires access to the internet. This method also needs an operator, or the user itself, to search for the code on the product package and, only then, scan it [12].

A more recent approach is RFID, which is becoming more prevalent among Portuguese retailers. The system is faster than the barcode system since there is no need to scan a code; the products must be placed inside the defined area, which ranges from the sensors. Ultimately, the system falls short, having difficulty identifying multiple products since the radio waves overlap or block themselves. It also isn't very sustainable since the RFID recycling process is difficult, resulting in a higher cost [11].

There have been attempts to combine the aforementioned systems with shopping carts to make them mobile. This reduces waiting time, as products are scanned when they are picked up from the shelf, and the bill is automatically sent via email. [48] developed precisely a system that follows these lines. An application that users could install to scan products was implemented, and the microcontroller in the cart was paired with it. The latter was then responsible for using the load cells installed in the cart to confirm that the product inserted in the cart was the same one with its barcode scanned.

A more advanced system is described in [9]. The author developed a system that used multiple forms of product analysis to achieve redundancy and greater accuracy. The main way of detecting products is through an RFID sensor that would be found in the shopping cart. The user is supposed to pass the product through this sensor, meaning that after analysis, the system would fetch the price and ID from the database. After that, the user is expected to place the product in the shopping cart, which also has a load cell, which allows the system to analyze the product's weight and thus re-verify its identity. A third verification method is also used through visual analysis, which uses a camera installed in the shopping cart. If this step fails, an LED warns the user of the discrepancy. This third system comprises a pre-trained CNN as the base model, specifically, an Inception v3 [9]. Starting from that, the authors used transfer learning to use self-made images and train the network. This provided very good results with a low quantity of images. They used images with different angles and lighting conditions.

This process is repeated for all products, and at the end of purchases, payment is authorized if the estimated weight is equivalent to the total obtained by the sensor. However, this payment is made at a payment gateway at checkout and not in the cart itself. If there is a discrepancy, an alarm is triggered, and the store is notified, blocking all payment gateways. While RFID tags are cost-effective to install, they do come with some limitations as previously mentioned. Moreover, their intricate design can make them susceptible to technical malfunctions, adding an element of unreliability [9]. In some instances, employee intervention may be necessary to address potential fraudulent activity, potentially hindering the overall user experience. Although the system efficiently prevents fraud, it's important to weigh these drawbacks before incorporating them into your operations [9].

3.2.2 Visual Systems

Label identification systems have also been implemented. These systems identify parts of labels, such as images, words, or even the barcode, but through computer vision. One of the advantages of this system is that it no longer necessarily requires an internet connection and can only be based on the model contained in the device [12]. Unfortunately, despite being the

most advanced system of those presented, it is not completely reliable, requiring extra systems to assist in the identification, as is the case with Amazon Go, which requires weight sensors and Bluetooth on the shelves to achieve a higher degree of identification [12].

In [49], Leo et al. sought to develop a system capable of classifying grocery products. This study developed a dataset of 5125 images distributed across 81 classes, distributed in a hierarchical structure. This structure comprises three main categories: fruits, packaged food and drinks, and vegetables. Each category had 19, 9, and 15 coarse classes, respectively. An advantage of the dataset used in this work was the fact that it was balanced. The author tested the dataset with four distinct architectures: ResNet, DenseNet, SENet and EfficientNet. After that, the authors also tested ensemble approaches, combining several of the tested models individually.

The test results revealed that ResNet-101 was the individual model providing the best results for all the architectures tested. It achieved an accuracy of 92.55% and a F1-Score of 93.23%. However, while testing the ensemble architectures, they proved a performance improvement, particularly with Ensemble C. Ensemble C was the name the authors gave to this configuration that brought together all the models that achieved an F1-Score greater than or equal to 91%. This configuration obtained an accuracy of 93.48% and an F1-Score of 94.46%, proving to be better than the best individual architecture. After comparing traditional classification techniques and basic neural networks such as AlexNet and VGGNet, the author demonstrated the effectiveness of using an ensemble technique, offering a significant advance in grocery product recognition.

Then, in [10], an end-to-end recognition system is developed composed of different stages. One for product detection, another for classification, and finally, one for refinement. For **product detection**, the study compares three methods: a traditional approach, a learning-based ACF detector, and a Single Stage Detector (SSD). The traditional method starts by detecting shelf lines and only then selects product zones, with an enhancement step to compensate for missing information. However, from the methods employed, SSD was selected because it allows for the simultaneous dealing with different sizes and scales of different objects [10].

Before proceeding to the classification, the number of region proposals is reduced based on the aspect ratio and the color histogram. At the **product classification** level, their features are collected based on traditional methods such as SURF, BRISK, AND ORB. Here, a single method is not selected, but a hybrid between the three is used to improve the results and increase the overall discriminative power of the system. This hybrid system is created based on the analysis of the accuracy obtained for each product through the three different methods. Before selection, normalization is performed on the matching values obtained to select the best extraction method correctly [10].

The last stage of the end-to-end system is **refinement**, which seeks to combine the multiple bounding boxes detected for the same product through a neighborhood-related clustering

algorithm. Through these new clusters, improving the results obtained in the classification and location of products is possible. The parameters of the clustering algorithm are dynamically adjusted for different images [10].

Using the GP-20 dataset and the developed pipeline, the authors [10] obtained 78.8% precision and 81.3% recall. They also obtained a test time value of 0.37 seconds, which revealed the system's real-time usage capabilities.

In another study, Guimarães et al. [12] make an excellent compilation of the current advances in detection and classification systems specifically developed for retail markets. One of the main notes indicated that only the most recent studies have started to focus on datasets with fine-grained classes. In other words, datasets with specific classes for each product, such as: "Coca-Cola", "Fanta", and "Sprite". Some older studies focused more on datasets with coarse classes, that is, datasets that used broader classes, such as "carbonated drinks".

Although in [12] the authors try to compare the works analyzed, using different datasets and metrics makes comparison difficult. However, in this study, they focused on GroZi-120 and GroZi-3.2k. In terms of object detection, some of the models that stood out the most were the following:

- ScaleNet, which utilizes ResNet features and achieves a score of 0.578 AR@1k on the MS COCO dataset;
- Goldman et al.'s strategy, which incorporates an Expectation-Maximization (EM) unit, achieves a score of 0.492 AP on the SKU-110K dataset. AR@1k is defined as the Average Recall;
- Santra et al.'s R-CNN detector, which outperformed others with an F1 score of 0.802 on the GroZi-3.2 dataset;
- RetailDet, which utilizes a Feature Pyramid Network (FPN), achieves a mAP of 0.590 on the SKU110k dataset;
- OS2D, a one-stage one-shot detector, emerges as a top performer with an mAP of 0.850 on the GroZi-3.2k dataset.

At the classification level, multiple systems are mentioned, such as the one developed by Goldman and Goldberger, proposing a system based on the sequence of products on the rack, achieving a Mean Error of 0.129 or Klasson et al. focus on hierarchical classification, achieving an accuracy of 0.804. However, one of the most notable mentions is an end-to-end solution developed by Leonid et al. that introduces a model using synthetic data for fine-grained classification refinements, achieving a region-matching score of 0.447 (AP) [12].

In 2021, James [50] developed a system very similar to the one this thesis aims to achieve. It would detect the products to be paid for and then create the account and payment details. However, the training was completely based on an existing dataset, MS COCO, which, despite having some hypermarket products, limits the user as there is no way to add products specific to their region or interest. The MS COCO dataset is one of the most complete datasets available today, comprising over two hundred thousand labeled images of around 80 different product categories [50]. The author used the YOLOv4 network for his tests, with pre-trained weights, and obtained a PA of only 43.5%. This is not a bad approach, but training a network from scratch could prove beneficial, as it ensures that the network only calculates the weights for the desired classes. However, many images are required to train the network from scratch. Here arises the need for a way to create big datasets easily.

Lee et al. [51] looked for a way to simplify the dataset creation process. They combined 3 different algorithms to synthesize images from a small amount of images. A cut-and-paste algorithm, a GAN shadow synthesis system, and a light variation algorithm. The fallback from this method is that the cut-and-paste algorithm only accounts for regularly shaped objects, not performing well in objects with odd shapes. The 3 algorithms allowed the system to synthesize images with several products per image and shadows that reliably reproduce those that exist in reality, thus guaranteeing a high correlation with a real checkout situation. Multiple networks were trained from this dataset, including YOLOv3, YOLOv5, RetinaNet, and some lightweight networks such as MobileNet V3 and ShuffleNet V2, always using the same hyperparameters to provide a basis for comparison. [51]. The authors also used a deep learning optimization model called NVIDIA TensorRT to maximize the throughput of deep learning models on embedded devices using NVIDIA GPUs, thus reducing the inferencing time of the models. Although the system was trained with synthesized images, it was then tested with real images to ensure maximum approximation to real environments and obtain precision values in these scenarios. To further reduce prediction errors, frame difference was used through the use of the OpenCV library, which ensured that the model only performed inference when it did not detect movement in the image. Of the models tested, YOLOv5L stood out with mAP and cAcc of 98.5% and 97.33%, respectively. MobileNet V3 stood out of the lighter networks with an inference time of just 0.142s with a minimum degradation of 0.3% mAp and 8.16% cAcc after being optimized with TensorRT [51].

Improvements in retail systems and computer vision show promise for a wide range of applications. The application of deep learning techniques has produced more accurate, dependable, and efficient systems in a variety of industries, including healthcare, security, and retail. However, in order to further improve these systems, future research must address issues in terms of creating datasets for specific and constantly evolving markets, ease of use of the systems for the

average user, computational cost and sustainability.

Chapter 4

Methodologies and Experiment Setup

This chapter presents a comprehensive review of the experimental setup and procedures employed in this study. It starts by outlining the key characteristics of the dataset used and providing essential background information for the subsequent processes. Subsequently, it deepens into the dataset's creation through clustering, offering detailed insights into each process stage. The chapter covers pre-processing, feature extraction, normalization, clustering algorithms, testing techniques, the development of a robust interface, and dataset testing. Additionally, the development of synthetic datasets is explored, encompassing object data creation, image generation, metadata translation, and testing. Lastly, the chapter concludes with an overview of the evaluation criteria used to assess the accuracy and effectiveness of the employed techniques.

4.1 Dataset Background Characteristics

One of the main challenges of this thesis depends on the fact that there is no assigned dataset. So, in addition to developing the system, the objective was to create a dataset. This is one of the most volatile points since all the knowledge the neural network can obtain comes precisely from this information set. Ensuring that this point is as optimized as possible is essential.

For this reason, in the first stage, images were collected from another dataset [52], which had some products that, despite coming from other countries, had a common appearance with national products. The original candidate image set contained 12917 images, with a total of 6348 products. These images contained the product on a white background. The original query image set contained 373 images of 104 randomly chosen products. From these two sets, we chose some products that were similar to products on the Portuguese market and built three different datasets to use as training:

- The first dataset consists of images composed of a white background captured in a controlled scenario. The files for these images have an 800 per 800-pixel resolution and 24-bit depth;
- The second dataset consists of images captured inside a supermarket, with the products held by a hand and a random background. The files for these images have a 3024 per 4032-pixel resolution and 24-bit depth.
- The third dataset was created from the second dataset, but with the images cropped along the edges of the products to reduce the image's background/product ratio.

To test this, images of the same products were taken in Portuguese supermarkets, with a supermarket background and the environment in which this system is most likely to be used.

In image 4.1, it's possible to see that the first three images are from the training dataset that was created from the dataset [52], and the last image was taken for the sole purpose of this study to perform testing.



Figure 4.1: Image Example of the different Datasets

The necessary metadata was created with the appropriate labels, and the data was augmented. The following types of augmentation were used for these datasets:

- **Random Horizontal Flip** - The image can flip horizontally with a specified probability of 50%;
- **Random Vertical Flip** - The image can flip vertically with a specified probability of 50%;
- **Random Perspective** - Applies a random perspective transformation to the image. The parameters were as follows: distortion scale at 0.2, $p = 0.2$, and interpolation at nearest;
- **Random Affine** - Randomly applies an affine transformation, which can be translation, rotation, scaling, or shearing. In this case, the parameters passed were 6° for the rotation, (0.1 , 0.1) for the translation, and (0.9 , 1.1) for the scale;

- **Random Rotation** - You can randomly rotate the image by a user-specified amount. In this case, 35 degrees.

Augmentation is performed offline. In other words, it is executed before the neural network is trained. Therefore, all images are saved locally, and all paths and respective labels for the generated images are added to the metadata. This guarantees a fair comparison between models since they all use the same dataset. It also speeds up the process since there's no need to run the augmentation process every time a new model is trained.

It was also ensured that the augmentation ensured a balance of all classes so that imbalances did not impact subsequent analysis. This can be verified on image 4.2.

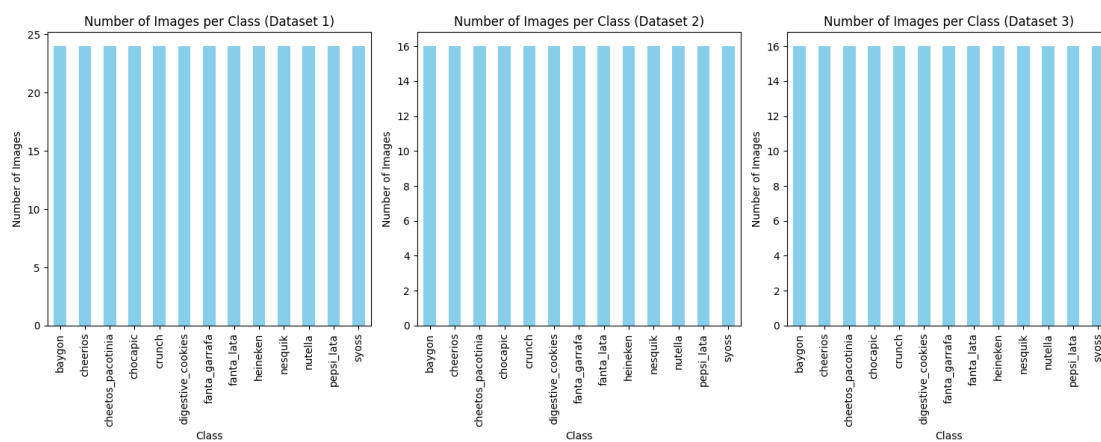


Figure 4.2: Number of images per class in each dataset

A simple CNN, denominated *Original Network*, was developed with the architecture displayed on image 4.3.

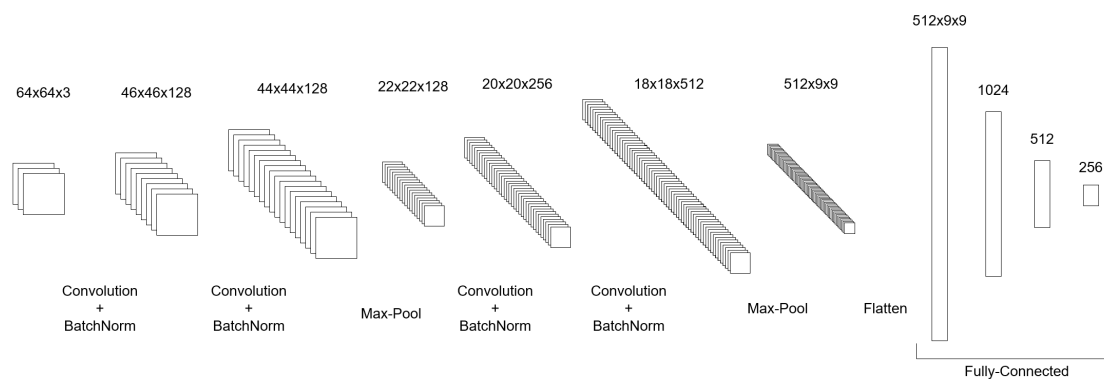


Figure 4.3: Architecture of the *Original Network*

In addition to what is shown in the image above, dropout layers were also added at the end of the first three fully connected layers to prevent over-fitting. Since this network did not have the objective of allowing a perfect classification but only to obtain some conclusions regarding the dataset to calculate the loss, cross-entropy was used, which is one of the most common functions for the situation that this network intends to face, as we seek to classify images with mutually exclusive labels. We intend to obtain the probability that the given label has to correspond to the past image.

Adam was used as the optimizer, as it is adaptive and converges faster than SGD (Stochastic Gradient Descent), a priority in this initial phase of work. A learning rate scheduler was also used, which varies the learning rate's value to avoid premature network optimization convergence and help the algorithm escape these situations, allowing the parameters to be explored more efficiently.

4.2 Dataset Creation via Clustering

Investigating the impact of background on training and product identification presents challenges due to the limited availability of images. The difficulty in finding an existing dataset featuring national market products necessitates the creation of a new dataset. To ensure successful model training, gathering a sizable amount of data is the main objective. With an insufficient dataset, there is a risk that the model might learn incorrect features. The intention behind creating this dataset is to establish a comprehensive collection of images that accurately depict national market products. Given this market's scarcity of datasets, generating a new, extensive dataset becomes imperative. The overarching aim is to compile a significant number of high-quality images to enable effective model training for accurately identifying and classifying products under various conditions.

The initial approach is to gather video images to maximize the volume and variety of data. It's important to capture photos in different light conditions and from various angles to provide a large amount of visual information. Initially, we only used images with objects on a white background to prevent the network from learning irrelevant features (as proven to be the best alternative in section 5.1). This approach aims to reduce the amount of data needed for training the network and prevent it from learning incorrect features. Creating this dataset is crucial for developing a product identification model that is accurate and versatile in handling real-world scenarios.

4.2.1 Video Extraction

The best way to get a lot of images in a short space of time is through videos. A video is nothing more than many images put together and played back at high speed. They usually have a frame rate, which is the number of frames per second they display, and a frame is one of these images. Therefore, it was essential to define that the data would be collected from videos, thus guaranteeing this large amount of data, since every small movement introduces a new angle of the product, a change in the background, or even an obstruction, which guarantees new data for the convolutional network to learn from.

The open-source FFmpeg tool can be used to extract the frames from a video with access to different settings via flags. Some of the most relevant allows setting the video and audio codec: “-vcodec” and “-acodec”; the video and audio bitrate: “-b:v” and “-b:a”, the frame rate: “-r” or even the resolution: “-s”. Although there is a specific flag to define the frame rate, it indicates the input or output frame rate depending on where it is positioned in the command. If placed in the output, it resets the number of frames in the output. However, one way to obtain greater control and smoother frame rate conversions is to use the video filter flag, which allows the user to adjust the frame rate directly in the video stream: “-fv fps=10”.

The frame rate was set to 60 fps for this project. This value was preferred to 24 or 30 fps to ensure that it is possible to get an image with minimal blur in a given second. A higher frame rate captures more detail each second, reducing the likelihood of motion blur affecting the quality of the extracted frames.

For instance, using the command below, we extract frames at a controlled rate:

```
ffmpeg -i input_video.mp4 -vf "fps=10" frame_%04d.png
```

In this example, we set the output frame rate to 10 fps to balance the volume of extracted images and the storage requirements, ensuring that each frame provides a new perspective without redundancy.

FFmpeg was chosen over the more common OpenCV because of its superior efficiency and control. FFmpeg is more optimized for processing large video files than OpenCV. Moreover, the fact that it offers extensive options in terms of codecs, bitrates, resolutions, and frame rates allows for more precise extraction while maintaining high image quality and optimum resource optimization.

4.2.1.1 Blur Detection

Although the large amount of data obtained from videos is a good starting point for creating a dataset, paying attention to its quality is necessary. The main concern is detecting blurred

images to reduce the number of images that are out of focus, which makes the feature extraction process more difficult when there is a large quantity. Two methods were used for this purpose. One begins by applying a threshold to the image in order to create a binary image and detect bright spots. A Laplacian filter is applied to this binary image, and the variance is calculated from the result. Based on the value obtained, it is possible to detect whether there is blur or not. Lower variance values indicate the existence of blur, while higher values indicate a clearer image. The other alternative involves the frequency domain, specifically a technique based on the Discrete Cosine Transform (DCT). By applying this transform to an image already in grayscale and normalized, we can obtain different coefficients that represent the different frequency components of the image. Low-frequency components represent the softer areas of the image. By calculating the sum of the squares of the coefficients, we can calculate the energy present in the high-frequency components, representing the details and edges of the image. It is possible to know whether the image is blurred by comparing the energy value with a threshold value.

To deal with cases of blur, an unsharpening mask was created, which increases the apparent sharpness. To do this, we start by creating a blurred image of the original image by applying a Gaussian Blur, with kernel size 5 and sigma value 2. The original image is blended with this softened image using the function `cv2.addWeighted`, where the original image's weight is increased by 1.0, and the blurred image's weight is subtracted by the same amount. This results in an image in which the details are more pronounced, reducing the existing blur in the image.

4.2.2 Image Clustering

Faced with a large amount of data, it became relevant to look for a way to label it in an easy way. In a case where it's only working with classification, the only thing needed for the metadata is the path for the image and a label. There's no need to indicate any bounding box or segmentation mask. One solution to help with this could be to aggregate all the images with similar characteristics and label them in batches.

The different datasets created manually were used to test this system and try to label them semi-automatically. A cluster has a set of images of a particular product pre-associated with it. The testing images are then aggregated into clusters, and the cluster's label is given to the newly associated images.

4.2.2.1 Pre-Processing

To ensure the best possible extraction of features, some kind of image pre-processing is necessary.

Equalization

One of the pre-processing attempts in the tests was image equalization. This consists of taking the image's histogram, a color space containing a channel that describes the intensity of the pixels, and distributing the values of this image to improve contrast and highlight details in very dark and very light regions of the image. Equalization is carried out by calculating the cumulative distribution function of the histogram and then normalizing this function. The old histogram values are then mapped to the new values according to the desired scale, and the new intensity values are applied to the corresponding channel.

Image Blur

Images contain a lot of information, and some feature extractors try to make the most of any edge or corner they can detect. However, sometimes, these are not very relevant or caused by camera artifacts or the environment in which the photo was taken. For this reason, another form of pre-processing is to apply a series of filters that apply a controlled blur to the image so that only important details stand out. One of the most common is Gaussian Blur, which applies a blur based on a Gaussian distribution function and a kernel of the size indicated by the user. However, the choice for pre-processing the images used in this work fell on the bilateral filter. This filter is highly effective at removing noise but keeping edges sharp, considering spatial proximity and similarity of intensities when averaging pixels. There are three parameters that were adjusted during the tests:

- **d**: the diameter of the neighborhood for each pixel;
- **sigmaColor**: Controls the similarity of colors. Higher values mean that more distant colors within the pixel's neighborhood will be mixed;
- **sigmaSpace**: Controls the spatial extent of the filter. Larger values mean that more distant pixels will influence each other as long as they have similar colors.

The main defect of this filter is its speed, as it is much more complex than the others. It is, therefore, not advisable to use a d greater than 5 for online applications, but since this test takes place offline, by default, a d equal to 9 was used, being only changed as needed.

Cropping / Region of Interest

At the pre-processing level, it's possible to optionally define areas that enhance and isolate the most significant parts of the image. There are two alternatives. One makes a pre-established cut in the center of the image, and the other calculates the region of interest (ROI) by analyzing the edges of the image.

The center crop function is designed to extract the central region of an image. This step is particularly useful when the primary focus is expected to be centrally located. The `find_roi` function performs the primary task of detecting the ROI within an image using edge detection and contour analysis. The steps are as follows:

- **Channel Selection:** The blue channel of the image is selected for further processing.
- **Edge Detection:** Canny Edge Detection is applied to the selected channel to identify edges within the image. Parameters such as `low_canny_threshold` and `canny_ratio` are used to fine-tune the edge detection process.
- **Contour Detection:** Contours are extracted from the edge-detected image using the function `cv2.findContours`. Contours with an area less than a threshold are filtered out, as they are related to noise and not to specific objects.
- **Bounding Box Calculation:** A bounding box is computed for each contour. The coordinates of these bounding boxes are used to determine the encompassing bounding box for all detected contours.
- **ROI Extraction:** The encompassing bounding box coordinates are used to crop the image, extracting the ROI.



(a) Original Image



(b) Image outputted by the function

Figure 4.4: Application of the 'find_roi' function

As shown in figure 4.4, the resulting image minimizes non-relevant background edges and primarily focuses on the product. This function worked primarily with images with a plain background, although it revealed a medium performance on images with cluttered backgrounds.

In the initial version of the code, a function was created to merge contours close to each other. The purpose of this function was to combine contours that belonged to the same object but were fragmented. This fragmentation of the object contours happened since the canny edge parameter values were tuned to exclude most of the contours that appeared in images with cluttered backgrounds. The pseudocode for this function is stated on algorithm 1.

Algorithm 1 Connect Nearby Contours

```

1: function CONNECT_NEARBY_CONTOURS(contours, max_distance)
2:   contours  $\leftarrow$  LIST(contours)
3:   connected_contours  $\leftarrow$  [ ]
4:   while contours do
5:     contour  $\leftarrow$  POP(contours, 0)
6:      $(x, y, w, h) \leftarrow$  BOUNDINGRECT(contour)
7:     center_x  $\leftarrow x + \frac{w}{2}$ 
8:     center_y  $\leftarrow y + \frac{h}{2}$ 
9:     nearby_contours  $\leftarrow$  [contour]
10:    nearby_indices  $\leftarrow$  [0]
11:    for each (idx, other_contour) in ENUMERATE(contours) do
12:       $(x2, y2, w2, h2) \leftarrow$  BOUNDINGRECT(other_contour)
13:      center2_x  $\leftarrow x2 + \frac{w2}{2}$ 
14:      center2_y  $\leftarrow y2 + \frac{h2}{2}$ 
15:      distance_squared  $\leftarrow (center\_x - center2\_x)^2 + (center\_y - center2\_y)^2$ 
16:      if distance_squared < max_distance2 then
17:        APPEND(nearby_contours, other_contour)
18:        APPEND(nearby_indices, idx)
19:      end if
20:    end for
21:    stacked_contours  $\leftarrow$  VSTACK(nearby_contours)
22:    hull  $\leftarrow$  CONVEXHULL(stacked_contours)
23:    APPEND(connected_contours, hull)
24:    contours  $\leftarrow$  [contours[i] for i in range(LEN(contours)) if i not in nearby_indices]
25:  end while
26:  return connected_contours
27: end function

```

This additional function worked as expected, but it wasn't used further. It was better to find an alternative by enhancing the image so that the detected contours were already joined together after passing through the Canny Edge Detector.

Resizing

Resizing the image as the final step in pre-processing is crucial for maintaining overall detail while ensuring consistency and compatibility with analysis algorithms. By standardizing images to a consistent size, we ensure all input images match in dimensions, which is key for uniform processing. This step is particularly vital for algorithms requiring fixed-size inputs, like Convolutional Neural Networks (CNNs).

By keeping resizing last, we first address high-level noise and irrelevant details, allowing for a more focused pre-processing on significant features. This sequence helps preserve the spatial relationships within the image by maintaining the aspect ratio, which is essential for accurate feature extraction. Additionally, resizing reduces computational demands, leading to quicker processing times and more efficient memory use. While, in this case, it does not enhance the performance of the pre-processing methods, it does enhance the performance of the feature extraction algorithms and CNNs, which work better on images of reduced dimensions.

This approach improves the accuracy of feature extraction by focusing on essential elements and ensures that preceding steps, such as Region of Interest (ROI) extraction and contour filtering, are more precise. So, resizing as the last step is key to preparing the ROI while keeping the original features' integrity intact.

4.2.2.2 Feature Extraction

Feature extraction is crucial for obtaining useful information from raw image data, facilitating effective analysis and classification. Selecting feature extractors, such as simpler extraction methods like SIFT or deep learning models like VGG16, EfficientNet, and ResNet, depends on various factors tailored to the studied dataset. The results using the different extractors will be evaluated.

For software libraries and tools, feature extraction is performed using popular deep learning frameworks like TensorFlow or PyTorch, which provide comprehensive support for implementing advanced models such as VGG16, EfficientNet, and ResNet. Additionally, traditional computer vision libraries like OpenCV are used to extract SIFT features, demonstrating a hybrid approach that combines classical and deep learning techniques. This integration ensures a comprehensive feature extraction process that meets the diverse requirements of the dataset while leveraging the latest developments in machine learning research.

Histograms

Histograms are a useful technique for visualizing the distribution of pixel intensities in an image. In the context of image processing, a histogram is a graphical representation that shows

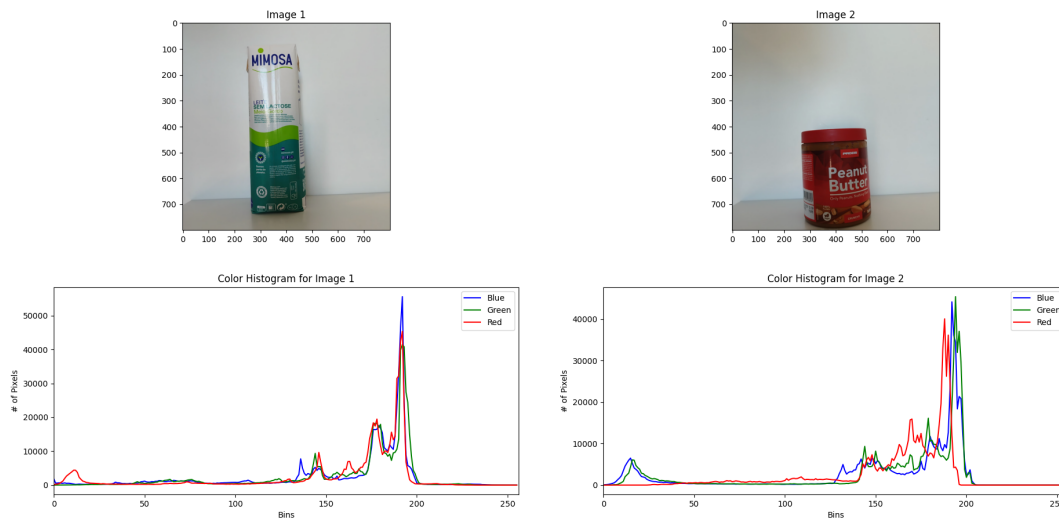


Figure 4.5: Example images and respective histograms

how often each intensity value occurs. For an RGB image, there are three separate histograms, each representing the distribution of pixel intensities for one of the color channels: Red, Green, and Blue. The intensity of each channel can range from 0 to 255, where 0 corresponds to no intensity (black), and 255 represents the maximum intensity for the color of the respective channel.

Similarly, there are histograms for images represented in the HSV (Hue, Saturation, Value) color space for each of the three channels. The Hue histogram represents the distribution of colors, Saturation shows the distribution of color intensity, and Value depicts the brightness levels.

Histograms are powerful visualization tools and feature extraction tools. The histogram values for each channel can be used as features to describe the distribution of colors in images, which is ideal for detecting changes in images, comparison, or object detection. They can also be used to ensure that certain features are on a comparable scale, which is important when comparing features.

In the image 4.5, we can see how the histograms reflect the colors present in the images. For both images, the respective histogram shows peaks of the three colors near 200 bins, corresponding to the background, since it is white and therefore consists of the three colors. In the case of image 1, there are also peaks of the three channels in the area of 180 bins, corresponding to the white part of the milk carton, and then there is a green and red peak near 150 bins and a blue peak near 140 bins, corresponding to green and its tonalities. In the case of image 2, the red peak

near 175 bins stands out, which highlights the importance of red in the product, although there are some shorter peaks in the green and blue channels, as there are product colors corresponding to different shades of red and grey.

SIFT

Using SIFT as a feature extractor offers some advantages over other simpler extractors since it offers scale and rotation invariance, making it possible to detect the same features regardless of the scale or rotation of the image. Furthermore, SIFT is robust to light variations. This algorithm returns a set of descriptors, which allows effective matching of key points. Some of the parameters used in this system are as follows:

- *nFeatures*: The number of features to retain. The best ones are always chosen. Increasing this number can improve the location of keypoints, but it also increases computing time and can detect irrelevant keypoints;
- *contrastThreshold*: The contrast threshold used to filter out weaker features in low contrast zones;
- *edgeThreshold*: The contrast threshold used to filter out features in edge zones;



Figure 4.6: Extraction of SIFT Features

The extraction of SIFT features must be moderate, as seen in image 4.6 since extracting all the descriptors detected can lead to the detection of features that do not belong to the desired object. In the methodology of this work, the parameters have been set fixed: *nFeatures* has been set to 100, *contrastThreshold* to 0.04, and *edgeThreshold* to 10.

VGG16

Whenever convolutional networks, like VGG16, are used for feature extraction, a network that has been pre-trained on large datasets, like ImageNet, must be used. This methodology guarantees the achievement of a resilient and superior feature representation. The aforementioned photos must first be downsized to 224 by 224 pixels in order to comply with the network's required input resolution and this feature extraction process. In order to provide the best possible processing, the images must also be normalized, which is defined as having an average and a standard deviation of 0.5.

After the preparatory steps, the images are then inputted into the pre-trained network. It is critical to note that, in this context, the network's utility as a feature extractor necessitates the removal of the last two fully connected layers. By ensuring that the extracted features accurately reflect the output from the first fully connected layer, this change maximizes the relevance and usefulness of the recovered features for further analytical pursuits.

EfficientNet

The efficientNet network is designed for efficient scaling with better accuracy and fewer parameters than VGG16. However, it is utilized similarly to the pre-trained VGG16 network at the implementation level. The input images undergo similar transformations, resizing, and normalization. However, in the former, the mean value is [0.485, 0.456, 0.406], and the sigma value is [0.229, 0.224, 0.225].

For the use of EfficientNet, the existing model in PyTorch was not used directly but rather a re-implementation provided by the "efficientnet_pytorch" package. This re-implementation of the network is completely consistent with the original TensorFlow implementation in the paper. Because of that, there's no need to remove any layers from the network, as the used library already has a built-in function for extracting features with the exact specifications and collecting them from the desired layer.

ResNet

ResNet networks incorporate residual connections, which introduce shortcut connections to skip one or more layers. These connections facilitate smoother gradient flow during backpropagation, effectively addressing the vanishing gradient problem and enabling the construction of significantly deeper networks.

Their implementation in this project methodology closely resembles previous networks, involving resizing and normalizing transformations for input images and initializing the network with weights pre-trained on a large dataset. Additionally, the final fully connected layer is omitted to enable feature extraction.

4.2.2.3 Normalization

Before starting the clustering process, it is essential to pre-process all the extracted features. One of the most important and crucial steps is scaling, which adjusts the range and distribution of the features in the dataset. Some of the advantages offered by this processing are as follows:

- **Consistent range of values:** Since CNNs produce feature maps with different values and, which can vary greatly depending on the extraction layer, the initial images, the activation functions, among other factors, scaling the values helps to keep the values consistent and easier to analyze and compare;
- **Improved Convergence:** If the data used in distance-based models, such as KNN or SVM, are on the same scale, its possible to speed up the convergence process and prevent them from getting stuck in a local minimum;
- **Balanced Influence:** In the type of algorithms in which the data will be processed, the distance-based ones, scaling also prevents features from impacting it disproportionately, which could lead to biased results. Without scaling, larger values would have a greater influence;
- **Improved Interpretability:** Because the features are on the same scale, the results and models become easier to visualize and interpret, which also helps to understand the influence of each feature on the final result.

There are three types of normalization most commonly used in ML. The Standard Scaler transforms the data to have a mean of 0 and a standard deviation of 1. The MinMax Scaler scales the values to a fixed range, usually between 0 and 1. The Robust Scaler uses the mean and the interquartile range (IQR), making it very robust to outliers. All three scalers were tested, and a combination of Robust Scaler and MinMax Scaler was eventually created. The Robust Scaler 4.1 was chosen because it deals better with the numerous outliers created by the CNN kernels and the inherent characteristics of the images that introduce strong noise and reduce their influence. Then, the MinMax Scaler 4.2 was added to ensure that all the features are on the same scale, guaranteeing equal influence of the data.

$$X' = \frac{X - \text{median}(X)}{Q_3(X) - Q_1(X)} \quad (4.1)$$

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (4.2)$$

4.2.2.4 Clustering

K-Means

The K-Means algorithm was chosen as the first method for clustering the dataset due to its simplicity and effectiveness in grouping feature vectors with similarities. K-Means iteratively refines the centroids of the clusters in order to minimize the variance within each cluster, making it suitable for discovering inherent groupings in the image feature data.

This algorithm facilitates the categorization of images into pre-determined classes. The initialization of centroids plays a crucial role in determining the efficacy of the K-Means clustering process. Notably, the selection of initial centroid positions markedly influences the ultimate outcomes of clustering. It influences the cluster quality, and proper initialization can accelerate convergence by starting closer to an optimal solution. The k-means++ method was initially utilized, which spreads out the initial centroids to improve clustering performance. This method helps achieve better initial cluster centroids, leading to more optimal clustering results. In the finalized methodology, the initial centroids were initialized with features extracted from selected images, which became the centroids. These images were chosen as reference points to evaluate similarity with other images, effectively customizing centroid placement based on the dataset's specific characteristics. The number of clusters was manually defined, as the number of classes for each dataset was known.

Although this algorithm works well, it is somewhat limited as it only considers spherical clusters and is very sensitive to the initial position of the centroids. We tested the separation of features using this algorithm, but to try to overcome its limitations, we moved on to other alternatives.

Gaussian Mixture Models

One alternative was the Gaussian Mixture Models, which can use different shapes and sizes to create clusters. These probabilistic models assume that all data points are generated by mixing a finite number of Gaussian distributions with unknown parameters.

In the application of this model, the number of components is defined, which is the number of classes over which the images are to be separated. The type of covariance is also indicated, in which it is defined as whether each component has its own general covariance matrix, whether all the components share the same diagonal, whether each component has its own individual diagonal, or whether each component has a singular variance. In this case, the “full” flag was used, which indicates that each component should have its own covariance matrix. Finally, it was indicated how the model initializes the components. It can initialize randomly based on a selection of points made by the user or use k-means or k-means++. The latter method chooses

the first centers randomly, but the following ones are chosen from a balanced data distribution, favoring points further away from existing centers. A "random_state" was also specified, in this case, 42, to ensure that the initialized centroids are always the same and to guarantee that the comparisons performed are not dependent on the points selected initially.

Support Vector Machine

Support vector machines were the last option considered, given their supervised nature. Here, however, the model was used in an unsupervised way. The main argument that has to be defined in SVMs is the kernel function. This function defines how the data will be mapped in high-dimensional space, where a linear separator, named a hyperplane, can be used to separate the data.

There are several kernels that can be used as stated in the subsection [2.3.4](#). In this case, the "Linear" kernel was always used, as it showed the best results during initial testing.

4.2.2.5 Testing Method

In order to establish an effective clustering system, an iterative and adaptive approach was adopted. Due to the absence of a predefined dataset and established rules, it was imperative to employ a flexible methodology. This evolutionary approach enabled the capacity to dynamically evaluate and implement decisions as the project progressed, ensuring continuous refinement based on empirical results.

The initial pivotal step involved defining the clustering system. The K-Means algorithm was selected for its proven reliability and widespread application in similar contexts. Various configurations were explored by combining K-Means with the previously mentioned feature extractors to enhance performance. This process entailed experimenting with techniques such as equalization, image resizing, cropping, the application of blur filters, and the use of a sharpening mask. The blur detector test was also tried to exclude images that were too blurry. Each parameter was meticulously fine-tuned through multiple tests to determine the optimal settings for each scenario.

Upon reaching a stagnation in progress, alternative clustering methods were investigated. First, the Gaussian Mixture Model (GMM) was tested, and subsequently, a Support Vector Machine (SVM) was configured for clustering rather than classification. This progression facilitated a comparison of the efficacy of different algorithms in accurately grouping the images.

In addition, when confronted with the limitations of the current dataset, a new dataset was created with distinct characteristics. Each new dataset was developed with the objective of realizing greater potential and achieving improved clustering results. Through ongoing analysis

of outcomes and iterative refinement of the applied methods, adept navigation through the exploratory nature of this project was achieved.

4.2.2.6 Interface

The graphical interface for the clustering system was developed to enhance productivity and simplify interaction with the system. It also provides visual feedback on the clustering process quickly and intuitively. The Dash framework was chosen for its flexibility, ease of implementation, and integration with Python.

The dashboard has three main functions:

- Clustering System
- Visualization of the features of the VGG16 network
- Access to training a classification network to obtain percentage results based on the dataset generated.

For clustering, users need to define which centroids they want in the folder where the dashboard will run. They should insert one or more photographs into the "Centroids" folder, from which they want features to be extracted to calculate the initial centroid position. Each image file must be named "label_xx.png", where "label" is the desired label for that image and xx is the image number for that label so that there are no images with the same name. After that, the user should enter the name of the new cluster model in a text field or select one of the existing models. In case he chooses this last option, the existing model will be used, and the new images passed for clustering will be clustered according to the clusters of the existing model. The user must then select the feature extractor (just valid for a new model) to use and load the images to be clustered either by dragging them to the indicated place or through a dialog window that opens when they click on it. Clicking the cluster button will run the algorithm, and boxes corresponding to each cluster will be displayed in a grid. Each box contains the name of the cluster (based on the name of the centroids indicated) and the images assigned to that cluster.

Dash

The interface was created using Dash, a framework for building analytical web applications. The Dash layout was designed to include multiple tabs, with one for clustering and another for classification. Dash provides a variety of pre-designed objects with customizable flags, which accelerates the application design process. User interaction with the application is achieved through callbacks, ensuring that the interface is dynamically updated in response to parameter

changes and results. The design when initializing the application, as well as the result of a test of the clustering system, can be seen in appendix B.

As Dash is implemented in Python, it seamlessly integrates with other libraries, especially machine learning libraries like Sci-kit Learn for clustering algorithms and PyTorch for deep learning models. It also integrates easily with the OpenCV library used for image processing.

Unfortunately, Dash began to show its limitations when dealing with a large amount of data. Uploading images took a long time to process, often causing the application to crash before moving on to the clustering process. After processing, the system also took a while to load the images, but this has been improved by reducing their display resolution.

These issues significantly impacted the computer's RAM usage, making it sometimes difficult to use and necessitating frequent restarts.

A simple solution, without altering the application's structure, is to deploy the system on a scalable infrastructure. By containerizing the application beforehand, you can easily deploy it on a cloud service and ensure compatibility between systems by managing dependencies and using a common kernel for all systems.

4.2.3 Dataset Testing

The Dash interface for the clustering system has a dedicated tab for a classification system, as shown in the image B.3. This tab allows users to choose a dataset within the system for training with the same network as the one used for the dataset background characteristics analysis or a ResNet. The architecture for this network is evidenced in subsection 4.1.

It is essential to conduct testing on the datasets using Convolutional Neural Networks (CNNs). The datasets generated through clustering are primarily intended for training neural networks, which will be utilized for image classification. Solely relying on clustering methods for image classification poses certain limitations, particularly in discerning features within complex backgrounds. Clustering algorithms categorize images based on overall similarity, encompassing both the main subjects and the backgrounds. This underlines the rationale for minimizing the background features. However, in image classification, the goal is to only classify the object. CNNs have the capability to address this distinction.

For CNN testing, the loss is computed using CrossEntropyLoss, and the model parameters are optimized using the Adam optimizer with a learning rate of 0.001%. Additionally, a StepLR scheduler is implemented with a step size of 10 and a gamma value of 0.1 to adjust the learning rate periodically.

The interface allows users to choose the network they wanna train from scratch with any datasets created via clustering, stored in a predefined directory, or with images uploaded directly

through the interface. It is also possible to keep training with new images in a network that was already trained in another iteration via a clustering dataset. This flexibility enables rapid evaluation and comparison of model performance on various datasets.

The interface provides users with key performance metrics, including accuracy, loss, and F1-score on the test set for training new and pre-trained models. These metrics clearly indicate the model's performance and are essential for evaluating the effectiveness of the dataset and the model.

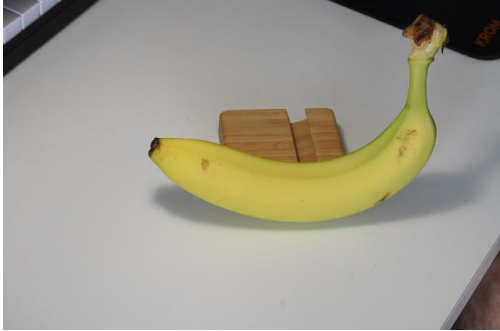
4.3 Synthetic Dataset Generation

The clustering system did not efficiently generate the dataset, as is possible to verify in the following chapter. While most images were well classified, some outliers were misclassified, rendering the dataset unusable. Incorrect classifications significantly impact the accuracy of the classification networks. Additionally, the method failed to generate the necessary bounding box metadata for training detection networks. It became essential to find an alternative method capable of rapidly generating a large volume of data with the ability to create metadata, including bounding boxes and masks for object filtering if needed. To address these issues, an advanced approach involving Mask R-CNN was employed to detect and segment the objects. This method stands in contrast to the previously used cut-and-paste approach with a shadow synthesis system by [51]. Utilizing Mask R-CNN, the process benefits from more precise object extraction and the ability to handle odd-shaped objects effectively. This improved methodology ensures a fast and accurate generation of datasets, complete with the necessary metadata allowing the use in training object detection networks.

4.3.1 Generating the Objects Data

The first step was to find a way of obtaining images of the objects that would make up the dataset. Since these objects are focused on the Portuguese market, there is no data source, so they must be fully captured. The intention was to insert them into a background in order to generate the new images, so it was important that only the object was captured and not the environment in which it was found at the place of capture. The images could be captured, and then the background removed manually using an image editing program, but this process would be inefficient and would take too long to achieve the desired number of images. It would also make the system inaccessible to any user, making it difficult to expand the dataset in implementation for a specific supermarket since it would need technical expertise. The solution was to use the Mask R-CNN neural network, which is capable of performing instance segmentation, returning

a mask for the detected objects, making it possible to filter out the background of the objects easily.



(a) Image before passing through the Mask-RCNN network



(b) Image resulting from the application of the mask generated by the Mask-RCNN network

Figure 4.7: Application example of the segmentation network

The first step involved capturing the feed from an external webcam to allow for better object positioning and a wider range of background options during image capture, which is not easily achievable with a laptop webcam. When prompted by the user, the current frame was isolated from the webcam feed and sent to the network.

The *maskrcnn_resnet50_fpn_v2* model, an enhanced version of the original Mask R-CNN with a ResNet-50-FPN backbone, was used. This network, pre-trained on the COCO-val2017 dataset, provides a mask that separates the object, a bounding box, a label, and a score, indicating the probability that the detected item matches the specified label. It provides the mask that separates the object, a bounding box, a label, and the probability or score that the detected item matches the indicated label. When requesting a prediction from the model, it is also possible to specify the *object_threshold* and *edge_threshold*, indicating the minimum required probability for the model to indicate a certain object and the value defining the edge to detect whether a certain part of the image belongs to the object or not.

Every time the system is prompted to capture an object, each image is saved in a folder, along with the image of the object only (i.e., the complete image with the mask applied). Additionally, a metadata file is created, indicating the original image's location, the object's image, the object, the bounding box of the object in the original image, and the label. The label in this metadata is the one entered by the user in the interface.

4.3.2 Generating the new images

After obtaining images of the objects, it's also essential to gather background images. This can be accomplished by taking photos in supermarkets and obtaining permission to use them through a formal process. Once these two components are secured, the system will have fulfilled its requirements for generating images. Some images were taken to test the system.

Image Augmentation

These enhancements involve random horizontal and vertical flips, perspective distortions, affine transformations, and rotations. These transformations are vital for simulating various real-world scenarios and enhancing the model's robustness.

The `augment_image` function applies these transformations using the PyTorch transforms module, which offers a flexible and efficient method for performing image augmentations.

Object Placement on Backgrounds

After augmenting the images of the objects, they must be inserted into the background. The initial step involves resizing the object image, ensuring that its size never exceeds half of the background image to maintain a realistic scale. Larger sizes are avoided to prevent objects from being too close together, which could affect the camera's focus. Once the scaling is addressed, a random position in the background is chosen for inserting the object, ensuring randomness and variability in the resulting image. This process involves utilizing OpenCV for tasks such as resizing, masking, and blending the object image onto the background. The output is a composite image with the object seamlessly integrated into the background, with variations in position and size in each generated image.

The images produced are then stored in a newly created directory. Additionally, a fresh metadata file is generated. This file contains the path to each image, along with the bounding boxes outlining the objects within the images and their respective labels. This methodology offers a significant advantage, as the positioning of the images is automatically handled by the system, allowing for the precise knowledge of image location and size. Consequently, including the bounding box as metadata is effortless. This approach facilitates the training of classification networks and enables the training of detection networks, which was not feasible with the images initially generated by the proposed clustering system. Moreover, the use of detection networks allows for the detection of multiple objects within the same image, even if they bear different labels. In addition to the default metadata created in JSON, an extra one is created in the format requested by the YOLO network to facilitate the process of training a new network.

4.3.3 Converting the metadata

The image generation system stores metadata in a custom JSON format. However, it's necessary to modify the metadata files to ensure compatibility with the YOLO network.

A dictionary assigns unique numbers to different classes since YOLO doesn't support string-based classes. Additionally, YOLO requires normalized coordinates, so this conversion is essential. The following code illustrates how this is accomplished:

Algorithm 2 Normalize Bounding Box Coordinates

Require: *image_path*, *bounding_box*

Ensure: Normalized coordinates: *x_center*, *y_center*, *width*, *height*

```

1: image  $\leftarrow$  Open image using image_path
2: image_width, image_height  $\leftarrow$  Get dimensions of image
3: x_min, y_min, x_max, y_max  $\leftarrow$  bounding_box coordinates
4: x_center_absolute  $\leftarrow \frac{x_{min} + x_{max}}{2}$ 
5: y_center_absolute  $\leftarrow \frac{y_{min} + y_{max}}{2}$ 
6: bbox_width  $\leftarrow x_{max} - x_{min}$ 
7: bbox_height  $\leftarrow y_{max} - y_{min}$ 
8: x_center  $\leftarrow \frac{x_{center\_absolute}}{image\_width}$ 
9: y_center  $\leftarrow \frac{y_{center\_absolute}}{image\_height}$ 
10: width  $\leftarrow \frac{bbox\_width}{image\_width}$ 
11: height  $\leftarrow \frac{bbox\_height}{image\_height}$ 
12: if  $0 \leq x\_center \leq 1$  and  $0 \leq y\_center \leq 1$  and  $0 \leq width \leq 1$  and  $0 \leq height \leq 1$  then
13:   return x_center, y_center, width, height
14: else
15:   return Error: normalized coordinates out of bounds
16: end if

```

Subsequently, the updated metadata is saved. This is achieved by generating a separate file for each object with a ".txt" extension. The content is saved as "class_id x_center y_center width height."

4.3.4 Synthetic Dataset Testing

When working with synthesized images, we divide them into training and validation datasets, with a split of 80% for training and 20% for validation. The validation dataset evaluates the network's performance on previously unseen images, providing valuable metrics for assessing the model's ability to generalize.

The synthesized images contain bounding box information, which is crucial for training detection networks. The YOLOv5 model was selected for this study due to its efficiency and

effectiveness in object detection tasks. Specifically, we chose to train the YOLOv5s model from scratch, as its smaller size makes it more suitable for training on laptops.

The training process was executed using the following command, which utilizes the default settings for the loss function, optimizer, and scheduler provided by the YOLOv5 framework:

```
python src/yolov5/train.py --img 640 --batch 16 --epochs 50
--data src/dataset.yaml --cfg src/yolov5/models/yolov5s.yaml
--weights '' --name yolov5_first
```

Key parameters included:

- `-img 640`: Specifies the input image size.
- `-batch 16`: Sets the batch size to 16.
- `-epochs 50`: Indicates that the model will be trained for the specified epochs.
- `-data src/dataset.yaml`: Points to the dataset configuration file.
- `-cfg src/yolov5/models/yolov5s.yaml`: Specifies the model configuration file.
- `-weights ''`: Ensures that the model is trained from scratch without any pre-trained weights.
- `-name yolov5_first`: Names the training run for easy identification.

This setup guarantees that the YOLOv5s model is trained on the synthesized dataset, providing a solid framework to evaluate the effectiveness of our dataset creation process. The results obtained from the validation dataset will be crucial in determining the performance and accuracy of the trained model.

4.4 Evaluation Metrics

The metrics chosen to analyze the performance of the different clustering tests and the YOLO network trained on synthetic images were decided on the basis of the collection made in chapter 2.5.

The silhouette coefficient was included as it provides insight into the density and separation of the clusters, quantifying how well each sample fits with the cluster assigned to it.

On the other hand, the Davies-Bouldin Index was employed to gauge the dissimilarity between clusters. This index assesses the uniqueness of each cluster, emphasizing the importance of distinct cluster boundaries to minimize the risk of erroneous groupings.

In addition to the above metrics, all wrong cluster assignments were counted manually to assess the system's accuracy. A misclusterization refers to an instance where data is wrongly grouped to another cluster that does not represent the correct image category. A percentage is also calculated to allow comparison by dividing the number of misclusterizations by the total number of images.

These three metrics allow for a quantitative and qualitative assessment of the system to serve as an evaluation guide for the ongoing refinement and optimization of the clustering methodology.

Other metrics were used to evaluate the YOLO object detection models trained with the synthetic images. Accuracy was used to measure the accuracy of positive predictions made by the model. Recall assessed the model's ability to find all relevant instances in the dataset. Mean Average Precision (mAP) provides a comprehensive measure that calculates the average precision for each class and the average between these values. On the other hand, intersection over Union (IoU) is a suitable metric to guarantee the model's effectiveness in detecting the object and calculating its bounding box, since the metric measures the overlap between the ground truth bounding box and the one predicted by the model. We also use mAP_{0.5}, which is the mAP for an IoU of 0.5.

Together, these measures provide a strong foundation for assessing the effectiveness of the YOLO system, guaranteeing that the completeness and accuracy of the detections were carefully considered.

Chapter 5

Results and Discussion

This chapter presents the outcomes of the procedures outlined in the methodology. Along with showcasing the results, this chapter delves into an analysis, elucidating the reasoning that guided the decisions regarding the subsequent tests.

Image clustering is a fundamental technique across various domains, from computer vision to pattern recognition, offering substantial depth in dealing with complex visual data.

The conducted analysis ensures an assessment based on both quantitative and qualitative metrics. Techniques such as the silhouette coefficient and the Davies-Bouldin index allow for a deep understanding of the efficiency and coherence of the clustering applied methodology. These metrics yield valuable quantitative measures, shedding light on the quality and compactness of the generated clusters. Additionally, a manual inspection of the distribution of images within the clusters ensures a qualitative evaluation of the process. This enables an assessment of the semantic coherence and interpretability of the clusters that extend beyond the scope of statistical measures.

This chapter also explores the performance of an object detection network, YOLO, trained with synthetically generated images. The use of synthetic data offers the advantage of obtaining a vast and diverse dataset, which is essential for ensuring good model robustness. This chapter will look at the results of the synthetic generation and testing of the trained YOLO network.

Metrics such as precision, recall, mean average precision (mAP), and Intersection over Union (IoU) were used to analyze the network's performance.

These detailed evaluations of both the clustering methodology and the YOLOv5 object detection system aim to establish the most efficient dataset and algorithm. With these results, this study aims to attain a stage where the most efficient dataset and algorithm can serve as robust training data and a solid foundation for classification in unfamiliar scenarios.

5.1 Dataset Background Characteristics

Training the models as indicated in the subchapter 4.1, provided the following results.

Table 5.1: Metrics obtained from background comparison testing

Training Dataset	Training Epochs	Accuracy	Loss	F1 Score
White Background	40	45.833%	2.710	33.333%
Random Background	60	18.333%	4.452	31.646%
Cropped Random Background	60	5.000%	12.112	1.613%

By analyzing the 5.1 table, we observe that the model trained on images with a white background obtained a higher accuracy than the models trained with a random background. This increased accuracy can be attributed to the model’s ability to effectively extract features from the product itself when the background is uniform and white. In contrast, with random backgrounds, the model may inadvertently extract features from the background, complicating the learning process for product features.

Observing the confusion matrices 5.1 can confirm this. The model trained on images with a white background can accurately identify some products in the test images with a random background. Meanwhile, models trained with random backgrounds tend to overfit to a certain class and identify all the products in the test dataset to that class due to the reliance on background features.

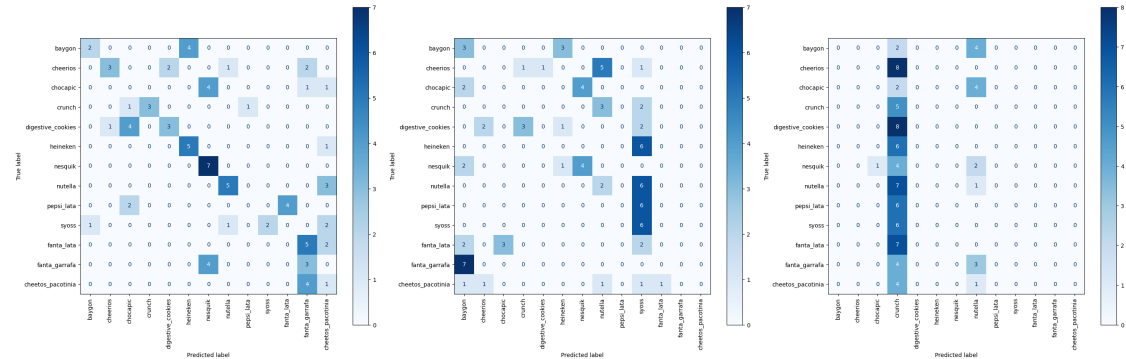


Figure 5.1: Confusion matrices obtained from testing

The particularly poor performance of the model trained with cropped random backgrounds, as indicated by its significantly lower accuracy and F1 score, suggests that the cropping process introduces additional challenges. Although the entire product is shown, cropping can eliminate contextual information that helps distinguish different products. The loss of this information can

lead the model to focus on less distinctive features of the product itself, resulting in poorer generalization and performance. Consequently, the model trained on cropped random backgrounds performs the worst among the tested configurations.

5.2 Generated Datasets

Given the need to obtain our own data, we created some datasets to test different aspects. The training datasets were numbered in order of creation.

The first dataset that contained only cereals had the following labels: "Chocapic", "Corn-Flakes", "CornFlakesEquilibrio", "EquilibrioChocolate", "EquilibrioFrutos", "Estrelitas", "Fitness", "FitnessChocolate", "Kellogs", "KellogsCornFlakes", "KellogsMaxi", "KitKat" and "Lion".

Starting from the Original Dataset 2, the datasets contained all the same products with the following labels: "amendoim", "cafe", "canaela", "cha", "leite", "pedras" and "sumol".

This was important to guarantee that the metrics used to compare the created clusters were not influenced by a different number of clusters.

Cereals - Original Dataset 1

The initial dataset was obtained at a supermarket using a smartphone equipped with a high-resolution 64MP camera capable of recording Full HD video at a frame rate of 120fps. This high frame rate was crucial for capturing specific frames amidst constant movement. The smartphone was moved horizontally to capture the desired footage to capture the shelves in rows. The video frames were then isolated, resulting in numerous images, some depicting multiple types of cereal in the same frame. This can be seen in the figure 5.2.



Figure 5.2: Frames extracted from the original video



Figure 5.3: Cropped video frames with single types of cereal.

As the goal was to categorize the images, those containing multiple types of cereal were discarded, leaving only images featuring a single type of cereal. This is necessary because the purpose of the dataset we are trying to create is to train a classification network that can only give one label per image. Nevertheless, some images still included parts of the adjacent top and bottom cereal boxes. Consequently, the original video was carefully edited, and irrelevant areas were removed. The video was rendered and subsequently separated into frames, as can be seen in figure 5.3. Images featuring multiple types of cereal were once again eliminated to produce the final dataset images. One notable advantage of this dataset is the absence of a cluttered background since the major part of the image contains the cereal box, ensuring a more effective feature extraction for analysis. Additionally, the high 120Hz capture rate minimizes image blurring, facilitating the extraction of features.

Photos with hands - Original Dataset 2

Captured in a smartphone with a 64MP camera and with a frame rate of 30fps because of hardware limitations. The capture resolution was 1080 per 1080 pixels to maintain a 1:1 aspect



Figure 5.4: Frames extracted from the Original Dataset 2 video

ratio and ease the processing that would be made. The aim of this dataset was to obtain a more neutral background in which there weren't so many features that could be detected. Despite this objective, a completely plain white background was not used. The objects were presented on camera suspended and held only by hands. The user holding the objects moved them around to display the different faces.

The capture method used was video, and frames were extracted from these videos. This approach led to some frames being shaky due to object manipulation and some frames not containing any objects at all, as illustrated in Figure 5.4.

After filtering out images containing manipulated objects and those without any objects, the remaining images can be viewed in the reference image 5.5. However, despite the filtering, some images still exhibit a slight blur, potentially due to the inherent instability of objects being held by hands.

Photos with full white background - Original Dataset 3 e 4

This dataset aims to address the shortcomings of the previous one by ensuring that only the objects' features are captured. To do this, the objects were captured with a completely white background, minimizing distractions from the surroundings. The objects were placed on a surface to ensure they were not partially covered by hands and remained effectively static. This dataset was also captured on a smartphone with a 64MP camera and a frame rate of 30fps because of hardware limitations. The capture resolution was 1080 per 1080 pixels



Figure 5.5: Frames filtered for the Original Dataset 2

We also positioned the objects as close as possible to the phone used to capture them to maximize the detail in the images. As in the previous cases, in the original video, there are frames in which hands are manipulating objects or frames in which there are no objects. These frames were subsequently removed.

During the first attempt to create this dataset, some images contained shadows and a visible line between the wall and the background, introducing unwanted elements. This can be observed in figure 5.6.

To try to combat these problems, a new capture was made, which produced better results, as can be seen in the image 5.7.

Photos with Camera - Testing Dataset

Finally, another dataset was created to assess the system's capability to focus on the unique features of various products while disregarding background variations. In this regard, we assembled a range of products from Portuguese supermarkets. Unlike the first dataset, the images in this collection were taken individually using a 24MP sensor camera without utilizing the video frame extraction system.

The intention was to create a dataset that would test the resilience of the later classification system by capturing each product from multiple perspectives, offering a comprehensive representation of the different aspects of the items. Furthermore, we intentionally incorporated diverse backgrounds in the images to test the network's capacity to separate the product features



Figure 5.6: Original Dataset 3

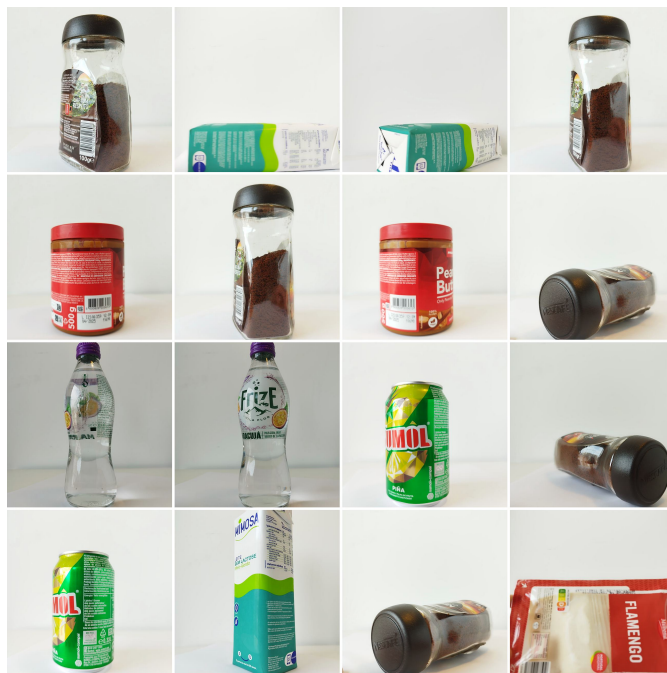


Figure 5.7: Original Dataset 4



Figure 5.8: Testing Dataset

from the surrounding context. This approach was designed to assess the system's capability to learn to identify and distinguish the products based on their inherent characteristics rather than their backgrounds.

5.3 Dataset Clustering

The K-Means algorithm stands out as one of the most widely recognized clustering techniques, making it an ideal starting point for our experimentation. Given our intention to manually define centroids, initializing clusters with features extracted from the selected images was imperative. The simplicity of analyzing images was exploited to facilitate this process based on their color histograms. This approach enabled us to efficiently extract features from the data, facilitating its organization for subsequent analysis.

Recognizing the significance of isolating relevant image content for accurate clustering, we performed a "Center_Crop" operation to focus specifically on the product while filtering out extraneous background information. This step aimed to enhance the specificity of our image analysis.

Starting with the RGB Histogram as the features extractor, the clustering performance evaluation yielded promising results, with a silhouette coefficient of 0.46263 and a Davies-Bouldin Index of 0.89921. These metrics provide quantitative insights into the cohesion and separation of the generated clusters, reflecting the effectiveness of our approach. A total of 203 images

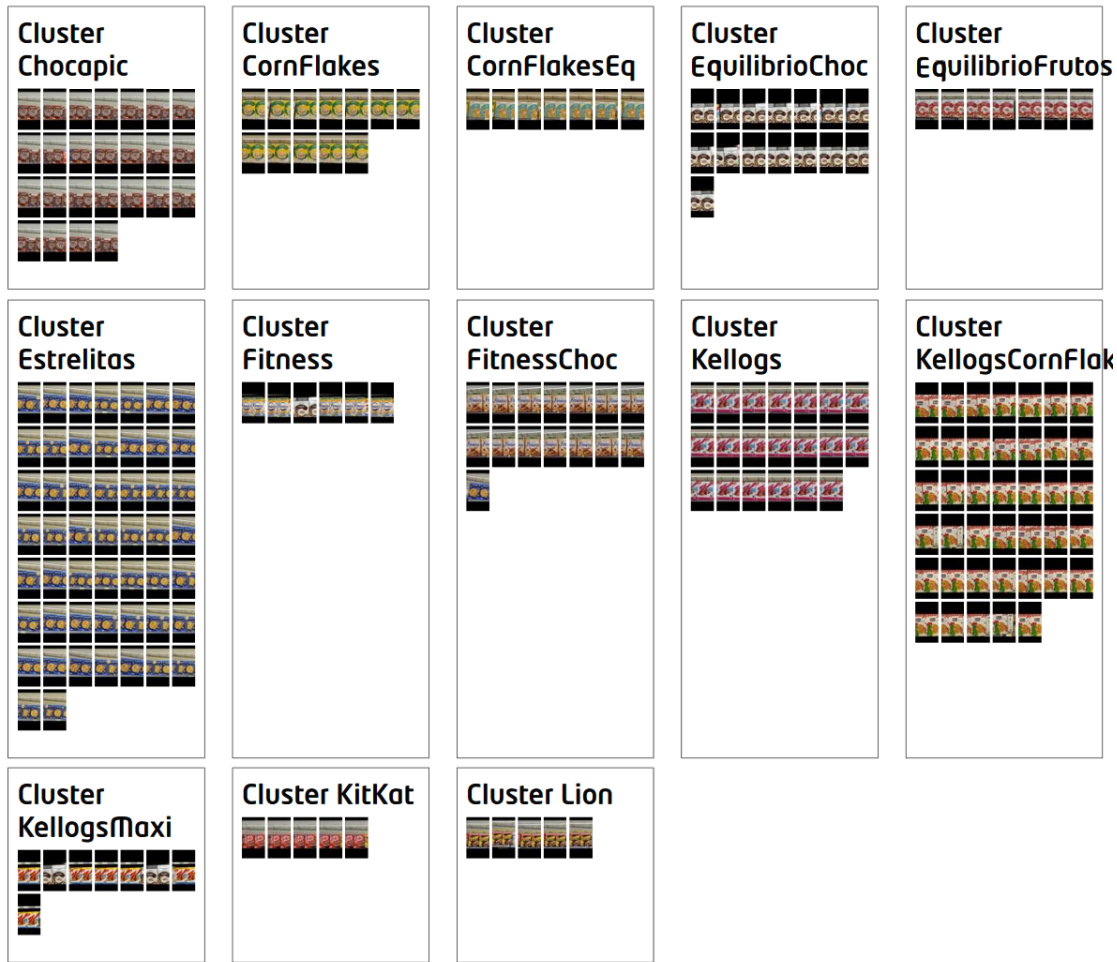


Figure 5.9: Original Dataset 1 clusters created by K-Means algorithm with histogram RGB features

underwent classification processes. Among these, four instances were inaccurately categorized, culminating in an impressive clustering efficacy rate of 98.03%, as shown in figure 5.9. Such a high level of accuracy significantly highlights the robustness of the adopted clustering methodology, effectively demonstrating its capability to accurately assemble images according to their inherent characteristics.

The findings are conveyed graphically as shown in figure 5.9.

After running some tests, it was confirmed that the video extractor's cropping highlights the product, which improves the extraction of features from the histogram. However, there are still a few cases where incorrect clustering occurs. These may be due to the black bars introduced by the cropping of the frames, which introduces a bias in the RGB histogram. Although stretching

Table 5.2: Clustering performance on Original Dataset 1 using single image per centroid

Clustering Algorithm	Feature Extractor	Image Resize	Image Crop	Filter	Unsharpening Mask	Silhouette Coefficient	Davies-Bouldin Index	Misclusterizations
K-Means	RGB Histogram	500x500	Center_Crop	No	No	0.463	0.8992	4/203
K-Means	HSV Histogram	500x500	Center_Crop	No	No	0.498	0.713	0/203
K-Means	SIFT	500x500	No	Bilateral Filter (d = 17 , sigmaColor = 5, sigmaSpace = 5)	No	-0.00365	2.116	161/203
K-Means	VGG-16	224x224	No	Bilateral Filter (d = 9 , sigmaColor = 5, sigmaSpace = 75)	No	0.374	0.952	3/203
GMM	SIFT	500x500	No	Bilateral Filter (d = 9 , sigmaColor = 5, sigmaSpace = 75)	No	-0.020	3.815	142/203
GMM	VGG-16	224x224	No	Bilateral Filter (d = 9 , sigmaColor = 5, sigmaSpace = 75)	No	0.374	0.954	3/203

the image could have eliminated these bars, an alternative approach was taken by utilizing a different color space.

Specifically, the HSV (Hue, Saturation, Value) color space was employed to tackle these issues. This color space offers the advantage of more accurately representing human color perception. Separating chromatic content (Hue and Saturation) from intensity (Value) makes the HSV space more resilient to light variations. The Hue component, dedicated exclusively to color, enhances the effectiveness of color detection and comparison.

Applying the HSV color space under the same testing conditions as those for the RGB space resulted in flawless clustering with zero misclusterizations. This outcome is supported by clustering effectiveness metrics: the silhouette coefficient increased to 0.498, and the Davies-Bouldin index decreased to 0.713. These values indicate that the features extracted from the images provided superior information for the clustering system, ensuring enhanced feature aggregation. The higher silhouette coefficient signifies that the images assigned to their respective clusters are more similar and closer to the cluster center.

Additionally, table 5.2 presents the results obtained using other feature extractors that are not solely based on color. Unfortunately, SIFT, which does not utilize color in its descriptor calculations, loses significant information by performing the analysis in grayscale. As a result, it often captures small background details, leading to the incorrect aggregation of images with different products. On the other hand, VGG-16, which incorporates color, achieved much more promising results, misclustering only three products. However, it is notable that the performance metrics for VGG-16 were worse compared to the model that used the RGB histogram as a feature extractor. This discrepancy can be attributed to the higher dimensionality of features in methods like SIFT or VGG-16, where the distance between points becomes less meaningful.

The application of Gaussian Mixture Models intended to capture non-linearities from the feature extractors, but it proved to be ineffective, leading to the decision to discard this clustering method.

After analyzing the results, we tested the clustering algorithm using typical supermarket items with the "Original Dataset 2". This posed challenges as the objects were not the sole focus

Table 5.3: Clustering performance on Original Dataset 2 using single image per centroid

Clustering Algorithm	Feature Extractor	Image Resize	Image Crop	Filter	Unsharpening Mask	Silhouette Coefficient	Davies-Bouldin Index	Misclusterizations
K-Means	RGB Histogram	500x500	Center_Crop	No	No	0.00962	1.799	218/363
K-Means	HSV Histogram	500x500	Center_Crop	No	No	0.0505	1.352	238/363
K-Means	VGG-16	224x224	Center_Crop	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace = 5)	No	0.129	2.145	68/363

of the images, making it difficult to extract relevant features from the background. As shown in Table 5.3, the clustering performance fell short across all scenarios. The highest silhouette coefficient achieved was 0.129, with 68 misclustered images out of 363. SIFT was ruled out as a feature extractor in this phase due to its previously observed poor performance in conjunction with K-Means.

The table further breaks down the comparison of several feature extractors and preprocessing methods. The VGG-16 feature extractor achieved the highest Silhouette Coefficient (0.129) and fewer misclusterizations (68 out of 363) than the RGB and HSV histogram extractors while having a higher Davies-Bouldin Index (2.145). In contrast, the RGB and HSV histogram extractors resulted in higher misclusterizations (218 and 238, respectively) and lower Silhouette Coefficients (0.0975 and 0.0505, respectively), indicating their ineffectiveness for this dataset and task.

All tests utilized Center_Crop and did not employ the Unsharp Mask filter. The presence of a Bilateral Filter in the VGG-16 configuration contributed to its superior performance in reducing misclusterizations, since not using this filter produced worse results with this extractor.

Overall, the K-Means clustering algorithm displayed varying degrees of success depending on the feature extractor and preprocessing method used; however, none of the configurations achieved satisfactory results based on the Silhouette Coefficient and Davies-Bouldin Index. This underscores the need for further refinement in clustering methodology, potentially through exploring other feature extractors, clustering algorithms, or additional preprocessing steps.

In order to improve these outcomes, we opted to utilize a Support Vector Machine (SVM) for clustering purposes. Despite SVM's conventional usage for classification, we adapted it for clustering in this context. Additionally, we augmented the centroid definition by using two images to account for multiple perspectives of the products.

Upon analyzing the results from table 5.4, several observations can be made.

The RGB Histogram extractor demonstrated the highest silhouette coefficient of 0.145 and a Davies-Bouldin Index (DBI) of 1.531, resulting in 189 misclusterizations out of 363. This suggests well-defined clusters compared to other methods. On the other hand, the HSV Histogram showed marginally fewer misclusters (182) and a lower silhouette coefficient of 0.045 with the

Table 5.4: Clustering performance on Original Dataset 2 using two images per centroid

Clustering Algorithm	Feature Extractor	Image Resize	Image Crop	Filter	Unsharpening Mask	Silhouette Coefficient	Davies-Bouldin Index	Misclusterizations
SVM Linear	RGB Histogram	500x500	Center_Crop	No	No	0.145	1.531	189/363
SVM Linear	HSV Histogram	500x500	find_roi + Center_Crop	No	No	0.045	1.886	182/363
SVM Linear	SIFT	500x500	Center_Crop	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace = 5)	No	0.00167	11.0456	334/363
SVM Linear	VGG	224x224	No	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace = 75)	No	0.0783	2.694	93/363
SVM Linear	VGG	224x224	No	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace = 5)	No	0.112	2.263	80/363

extra step of identifying the area of interest (ROI) and center cropping.

When employing a bilateral filter, the SIFT feature extractor kept performing poorly, with a silhouette coefficient of 0.00167 and a high DBI of 11.0456, resulting in 334 misclusters, indicating its ineffectiveness for this specific clustering task.

In contrast, the VGG feature extractor with varying bilateral filter parameters displayed more promising results. With the use of a Bilateral Filter with a *sigmaSpace* parameter of 75, VGG achieved a silhouette coefficient of 0.0783 and a DBI of 2.694, with 93 misclusters. This particular run analyzed images in the HSV color space. In the case of a *sigmaSpace* parameter of 5 and analyzed in RGB space, the silhouette coefficient improved to 0.112, the DBI decreased to 2.263, and the misclusters were reduced to 80. This suggests that VGG features can enhance clustering performance, especially when combined with tuned filtering parameters.

Therefore, out of all the techniques examined for this dataset, the RGB Histogram produced the most distinct clusters despite VGG features with optimal filtering parameters that also showed noteworthy performance. Although clustering results were improved by switching to a Support Vector Machine (SVM) and using two photos per centroid, more improvements are required for more accurate clustering of supermarket objects.

To confirm that the problem in clusterization came from the background, a feature display was created in the dash interface that allows the displaying of the features as shown on image [5.10](#).

As can be seen, there are some filters that put a lot of emphasis on the edges, and so, in some of the feature maps, any contour is emphasized, either from the background or from the product. However, in the filters that don't emphasize the contours so much, some contours of the product are not highlighted, and even so, background details are visible. This means that despite the network weights, these features are assumed to be from the product, making the clustering process more difficult.

Given the potential complications introduced by the irregular background inherent within Original Dataset 2, the investigation was advanced through experiments utilizing Original Dataset

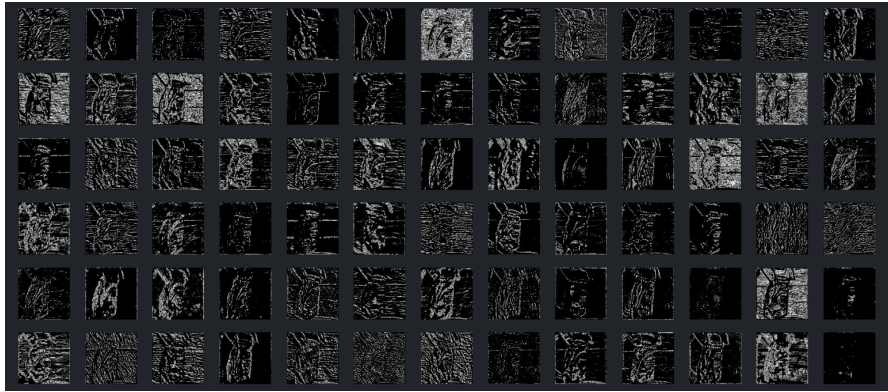


Figure 5.10: Features extracted by the VGG-16 network of an image of the Original Dataset 2

3, as delineated in Table 5.5.

The experimental methodology using the VGG-16 model for feature extraction and a Support Vector Machine (SVM) with a linear kernel was shown to be the best setup. This configuration showed the best outcomes, achieving a silhouette coefficient of 0.355 and the lowest Davies-Bouldin Index at 1.245, without misclusters. These results show an excellent clustering efficiency with distinct and coherent clusters.

On the other hand, the combination of the EfficientNet model for feature extraction and an SVM with a linear kernel resulted in less-than-ideal performance. With this arrangement, the silhouette coefficient was significantly lower at 0.065, and the Davies-Bouldin Index was higher at 3.183. There were also 6 cases of misclustering. Due to the substantial intra-cluster variability, these signs imply that this design did not promote unique cluster forms.

A moderate level of performance was attained by utilizing an SVM paired with the ResNet model as the feature extractor without implementing cropping (at a resolution of 256x256). This arrangement limited misclustering to two occurrences, obtaining a silhouette coefficient of 0.203 and a Davies-Bouldin Index of 1.692. This result suggests a really respectable degree of cluster assignment accuracy and a passable degree of clustering quality.

Applying center cropping to the ResNet feature extractor (256x256 resolution) resulted in an

Table 5.5: Clustering performance on Original Dataset 3 using two images per centroid

Clustering Algorithm	Feature Extractor	Image Resize	Image Crop	Filter	Unsharpening Mask	Silhouette Coefficient	Davies-Bouldin Index	Misclusterizations
SVM	VGG-16	224x224	No	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace = 5)	Yes	0.355	1.245	0/189
Linear	VGG-16	224x224	No	Bilateral Filter (d = 19, sigmaColor = 75, sigmaSpace = 150)	Yes	0.065	3.183	6/189
SVM	EfficientNet	224x224	No	Guided Filter (radius = 9, eps = 0.1)	Yes	0.203	1.692	3/189
Linear	EfficientNet	224x224	No	Guided Filter (radius = 9, eps = 0.1)	Yes	0.249	1.519	0/189
SVM	ResNet	256x256	No	Guided Filter (radius = 9, eps = 0.1)	Yes	0.203	1.692	3/189
Linear	ResNet	256x256	Center_Crop	Guided Filter (radius = 9, eps = 0.1)	Yes	0.249	1.519	0/189

Table 5.6: Clustering performance on Original Dataset 4 using multiple images per centroid

Clustering Algorithm	Feature Extractor	Image Resize	Image Crop	Filter	Unsharpening Mask	Silhouette Coefficient	Davies-Bouldin Index	Misclusterizations
Four Images Per Centroid								
SVM Linear	VGG-16	224x224	No	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace= 5)	Yes	0.170	1.887	64/1438
SVM Linear	ResNet	256x256	No	Guided Filter (radius = 9, eps = 0.1)	Yes	0.129	2.114	244/1438
Multiple Images Per Centroid								
SVM Linear	VGG-16	224x224	No	Bilateral Filter (d = 9, sigmaColor = 5, sigmaSpace= 5)	Yes	0.122	1.770	16/1438
SVM Linear	ResNet	256x256	No	Guided Filter (radius = 9, eps = 0.1)	Yes	0.112	2.429	30/1438

improvement. With no evidence of misclusters, this modified configuration produced a Davies-Bouldin Index of 1.519 and a silhouette coefficient of 0.249. The configuration's results indicate an ideal balance between intra- and inter-cluster compactness, separation, and perfect cluster assignment accuracy.

Finally, with a view to extending the results further, Original Dataset 4 was tested, which, despite maintaining the white background, seeks to maximize the area of the image in which the products are displayed. Although this change improves the clustering process when histograms are used as feature extractors, it was not guaranteed when the extractor was a CNN. Table 5.6 shows the tests carried out.

We started the tests using just 4 images per centroid. The first configuration used an SVM with a linear kernel and a VGG-16 network as an extractor. This combination obtained moderate performance, with a silhouette coefficient of 0.170 and a Davies-Bouldin Index of 1.887 with 64 misclusterizations of 1438 total images. These results suggest that the clusters formed are of moderate quality with relatively well-separated clusters. The second combination swapped the VGG for a ResNet, which resulted in worse performance. This set obtained a silhouette coefficient of 0.129 and a higher Davies-Bouldin index of 2.114 with a total of 244 misclusterizations. These metrics reiterate the poor clustering performance, with less distinct clusters and higher intra-cluster variance.

In the second set of tests, multiple images were used per centroid in order to highlight different faces and positions with which the SVM could compare. The number of images per centroid was adjusted for each product according to the need to expose more angles of each one, but the number of images per centroid was between 9 and 12. The tests were the same as those carried out with 4 images per centroid. In the case of the SVM with the VGG-16 network as an extractor, there was an improvement in performance with a silhouette coefficient of 0.122 and a relatively low Davies-Bouldin index of 1.770. This configuration achieved far fewer misclusters, with only 16 out of a total of 1438 images. When SVM and ResNet were used as extractors, performance was also higher than when only 4 images per centroid were used. However, this performance

is still lacking against the one that used multiple images and VGG as a feature extractor. With a silhouette coefficient of 0.112 and a Davies-Bouldin index of 2.429, this combination had 30 misclusterizations.

By analyzing these latest tests, it's possible to confirm the impact of the number of images per centroid on the performance of the clustering process and the reduction in misclusterizations. This works better since merging numerous photos per centroid yields a richer feature representation. With a more extensive feature set to characterize each cluster, this modification may be able to capture a greater variety of variances within each category. It also continues to be confirmed that the VGG-16 network has the best performance for feature extraction, providing better clustering metrics and a lower number of misclusterizations.

Introducing a blur detector was also tested, as mentioned in the sub-section 4.2.1.1, to remove images that had blur so as not to corrupt the feature extraction process. However, this detector detected blur even in irrelevant areas, such as the background, or even in images with little to almost no blur. Despite the different attempts to adjust the threshold, in all the tests, the two detectors tested always flagged too many images positively for blur, and the resulting dataset ended up being very small, making it not very relevant for training a classification network.

5.4 Dataset Testing

By training models with the datasets created, some evaluation metrics were drawn to draw conclusions about the effectiveness of the datasets in training the model and identifying products in more challenging situations, such as those provided by the test dataset. The models were trained on the *Original Network* architecture since it has fewer parameters (8,504,460 parameters) against the ResNet Model (11,182,668 parameters), and the training performance was a concern on the low-end hardware that was accessible. To provide a more detailed view of the model performance, the loss, accuracy, and F1-Score were plotted. These graphs offer insight into the evolution of training and help comprehend how the model generalizes unseen data.

5.4.1 Datasets created for clustering testing

5.4.1.1 Original Dataset 2

We started this analysis with Original Dataset 2, which has a background that isn't completely smooth and some hands-on parts of the objects. In figure 5.11, we can see the evolution of the training loss and accuracy, and in figure 5.12, the F1-Score. The graphs show a steady

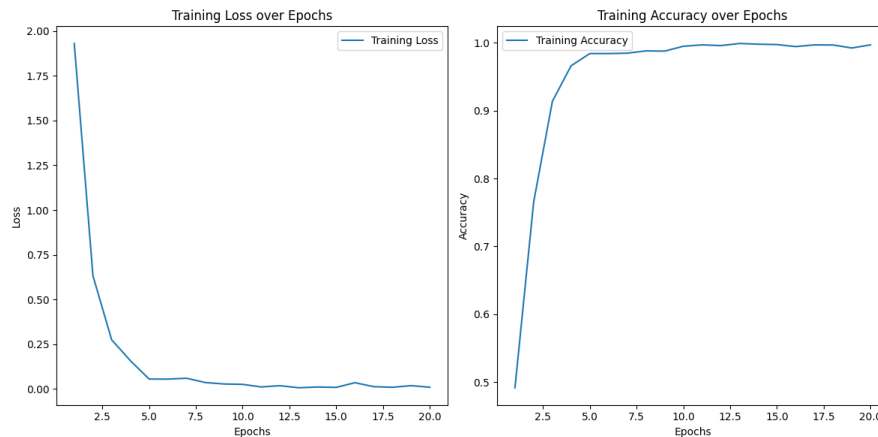


Figure 5.11: Training loss and accuracy for Original Network trained with Original Dataset 2

decrease in loss and a steady increase in accuracy, until they reach almost 0 and 100%, respectively. The training was stopped when they were already reaching a point of stability to avoid overfitting.

The graphs show a rapid drop in loss within the first five epochs, which indicates that the model learns quickly in the first phases. After this initial drop, the loss continues to fall but at a lower rate until it stabilizes around the tenth epoch. Accuracy increases rapidly in the first five epochs, getting close to 95%. It then fluctuates until it reaches a plateau close to 100% around the tenth season. The behavior of F1-Score is similar to that of accuracy.

The model learns quickly and efficiently, as indicated by the training accuracy and the F1-Score. It also converges quickly, as can be seen by the stabilization of the metrics in the first few training epochs.

Although one might suspect that the model was overfitting, this is not the case, as training sessions were run with other training epoch values and twenty epochs were considered the ideal point.

The confusion matrix in image 5.12 shows both continuous difficulties and dubious performance. 19 times, the prediction "amendoim" is accurate, with a few misclassifications to "cafe" and "canaela". There are 5 accurate predictions for "cafe", however a lost of other instances were poorly predicted for another class. This shows that "cafe" and these other classes have some characteristics in common. Although the "canaela" class has 2 correct predictions, 14 cases are still incorrectly classified as "amendoim" suggesting that there is still work to be done in differentiating between "canaela" and "amendoim". There's no correct prediction for "Cha" as it is still confused with other labels, meaning many background features are being extracted, which confuses the model. Although "leite" has 12 accurate predictions, there is still confusion be-

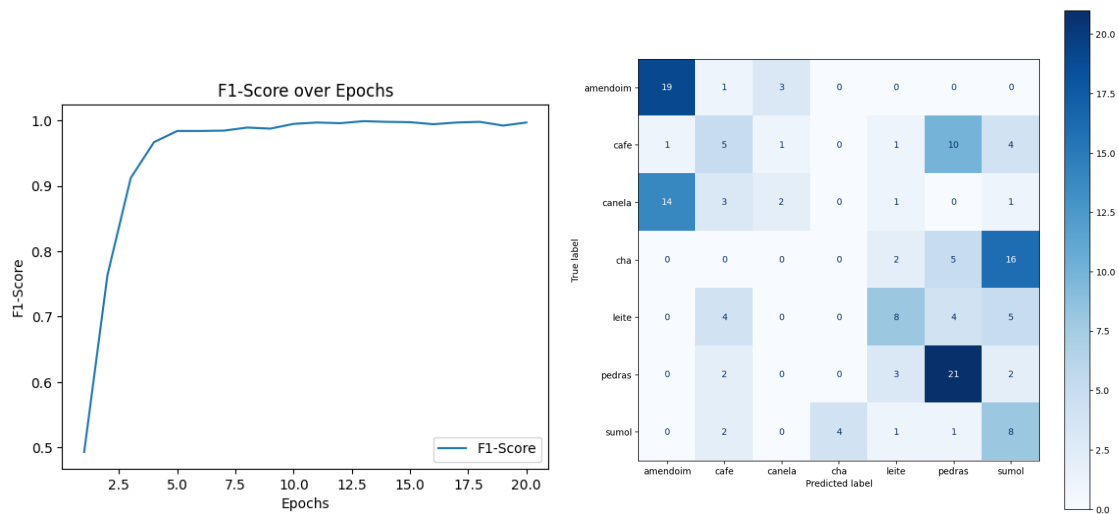


Figure 5.12: F1-Score and confusion matrix resultant from testing for ResNet network trained with Original Dataset 2

tween "cafe", "pedras", and "sumol", indicating that further separation is necessary. 21 of the identifications of "pedras" are accurate, and there are fewer misclassifications, indicating improved performance for this class. Even with 8 accurate predictions, "sumol" also exhibits some misclassifications, especially regarding "cha", suggesting that more differentiation is required.

5.4.1.2 Original Dataset 3

In the case of Original Dataset 3, we are already looking for a white background, although there is a small separation between the floor and the wall. In figure 5.13, we can see the evolution of the training loss and accuracy, and in figure 5.14, the F1-Score.

The graphs show an evolution very similar to that of the model trained with Original Dataset 2, although it is slightly slower, as can be seen from the slightly lower slope of the performance metric curves. However, in this case, we can also confirm a steady decrease in loss and an equally steady increase in accuracy. The curves begin to plateau from the eight season onwards. This model was also stopped early to avoid overfitting the model.

This model also learns efficiently, as the accuracy and the F1-Score indicate. It converges a little more slowly than the previous model but is also able to reach convergence relatively quickly.

Analyzing the confusion matrix in image 5.14, we can see a great improve in performance. It's possible to conclude that the model performs well for the "amendoim" class, correctly predicting 20 instances with 3 misclassifications. This suggests that the model does a good job of

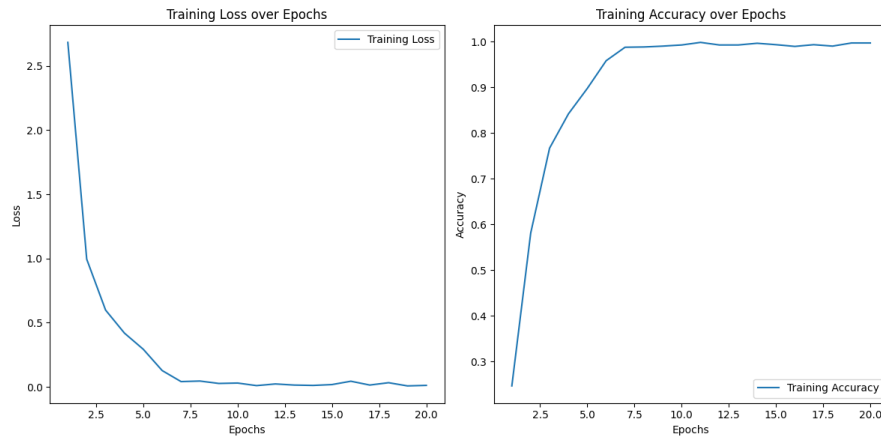


Figure 5.13: Training loss and accuracy for original network trained with Original Dataset 3

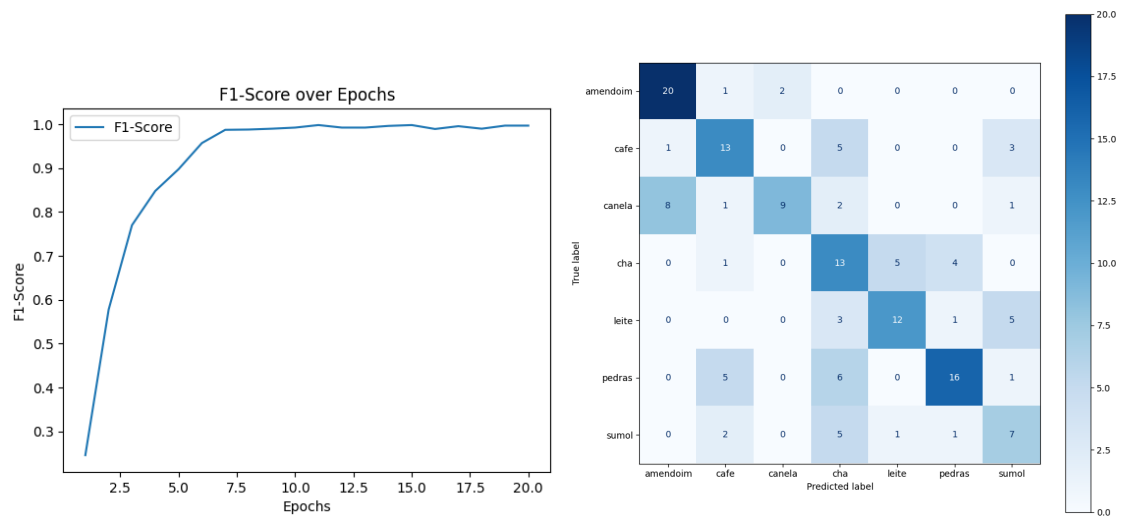


Figure 5.14: F1-Score and confusion matrix resultant from testing for Original Network trained with Original Dataset 3

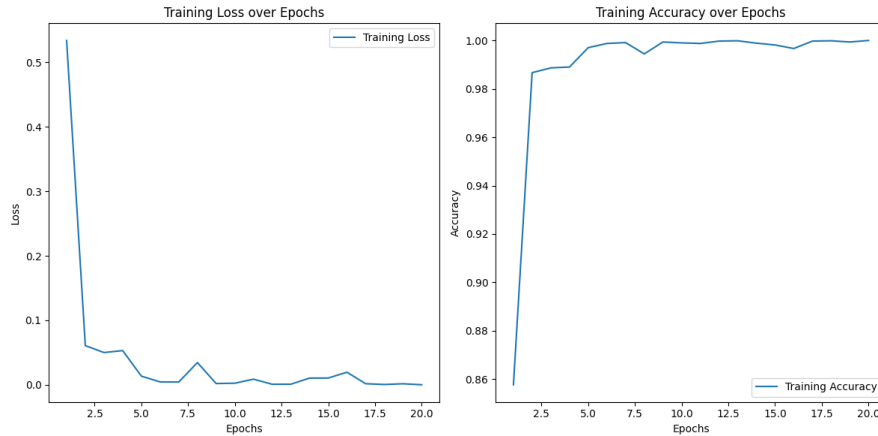


Figure 5.15: Training loss and accuracy for original network trained with Original Dataset 4

capturing the characteristics that set "amendoim" apart. The algorithm properly detects 13 instances of "cafe" but misclassifies 5 as "cha", 3 as "sumol", 1 as "amendoim", and 1 as "pedras". This suggests that "cafe" shares features with these other classes, leading to some confusion and indicating a need for further refinement. The "canaela" class poses a significant challenge for the model. Out of 21 true instances, only 9 are correctly predicted, while 8 are misclassified as "amendoim", 2 as "cha", 1 as "cafe", and 1 as "sumol". This high level of misclassification indicates substantial feature overlap with "amendoim", necessitating improved feature extraction or additional training data to enhance differentiation. For "cha", "leite" and "pedras", there is an average classification, being the biggest part of the instances well classified, with some misclassification between other two to three classes. On the "sumol" class, the model performs less accurately. It predicts seven cases accurately, but incorrectly labels five as "cha," five as "leite," one as "cafe," and one as "pedras." This suggests that "sumol" has features in common with multiple other classes, which causes misclassifications to occur frequently.

5.4.1.3 Original Dataset 4

Finally, we have the graphs of the evolution of the loss and accuracy and the F1-Score during training in the images 5.15 and 5.16 respectively.

The training loss starts at around 0.45 and begins to drop rapidly until around 0.05 where it begins to oscillate while approaching zero, close to the tenth epoch. From then on, the loss remains close to zero. Accuracy starts at 86% and rises rapidly in the first three training seasons, reaching around 99% by then. From then on it stabilizes at 100%, albeit with occasional fluctuations. The F1-Score, like the other models, shows behavior equivalent to accuracy. This indicates that the model has achieved good precision and recall.

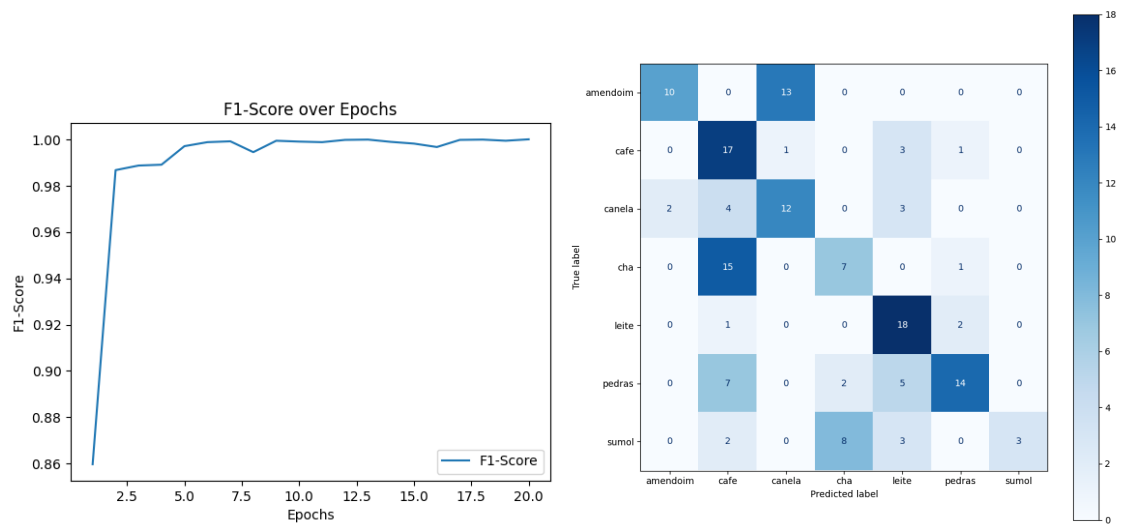


Figure 5.16: F1-Score and confusion matrix resultant from testing for Original Network trained with Original Dataset 4

The rapid convergence and stabilization of the model indicate that it, like the others, evolved appropriately according to the training set. Although, at first glance, the high accuracy values could indicate a situation of overfitting, training was carried out with different epoch values to ensure that this was ideal.

There have a slight setback, as evidenced by the confusion matrix. The "amendoim" class is properly predicted only 10 times, with 13 instances being wrongly labeled as "canela". This might occur because the products are positioned close to the camera, resulting in the perception of smaller details. Additionally, the similarity in color of the products makes it harder for the model to distinguish between them. The model properly detects 17 instances of "cafe," but incorrectly labels 3 examples as "leite", 1 as "amendoim", and 1 as "pedras", suggesting ongoing confusion between these classes. The "canela" class is still difficult to forecast, with only 12 correct predictions out of 21 true instances. Misclassifications include 4 as "cafe," 3 as "pedras", 2 as "amendoim", and 1 as "sumol". The model has trouble with the "cha" class; it only predicts seven cases correctly and incorrectly labels fifteen as "cafe," one as "pedras," and one as "sumol". This suggests that there are a lot of similarities between "cha" and "cafe," which causes misclassifications to happen frequently. With 18 accurate predictions, "Leite," and only two slight misclassifications, these results demonstrate the ability of the model to recognize this class. For "pedras," the model correctly predicts 14 instances but shows significant misclassifications into "cafe," suggesting a need for more distinct features between these two classes. Lastly, the model struggles with the "sumol" class, only making three accurate predictions and

misclassifying multiple times.

The model's overall performance is good for some classes, such as "cafe" and "leite", but it struggles with others, especially "cha" and "sumol".

Table 5.7: Metrics obtained from the dataset classification testing

Training Dataset	Training Epochs	Accuracy	Loss	F1 Score
Original Dataset 2	20	41.140%	4.708	35.801%
Original Dataset 3	20	58.677%	5.418	58.661%
Original Dataset 4	20	52.610%	8.546	51.478%

Analyzing the results of the 5.7 table, it's possible to confirm that the presence of hands and the slightly raised background introduce variability and additional features unrelated to the product. These are factors that contribute to the lower accuracy (41.140%) and F1-Score (35.801%) of this model. The hands, being visually complex and changing position between frames, can be confused with the object and distracting for the model.

In the case of the model trained with Original Dataset 3, the white background offers a substantial improvement over the previous model, achieving an accuracy of 58.667% and an F1-Score of 58.661%. The presence of a visible line on the wall and shadows introduce some noise, which does not allow the model to perform better. A greater number of images per class could also be a factor that would help this model evolve.

Finally, the model trained with Original Dataset 4 was evaluated. This one has the most controlled environment of all the datasets, with a white background and close proximity of the objects to the camera. With an accuracy of 52.610% and an F1-score of 51.478%, this dataset unexpectedly underperformed the previous one. This may be due to the distance of the objects from the camera since, being closer, more irrelevant details, flaws in the product, and even reliefs are captured, making the task of classifying the model more difficult.

Concluding the analysis of the datasets classification, there is a definite trend of increased model performance in the transition from a cluttered background with hands to a crisp white background. The model can more effectively isolate and pick up on the unique characteristics of the objects when background noise and other distractions are removed. The findings underscore the significance of backdrop selection in picture classification tasks and stress the need to exclude extraneous information in order to improve model resilience and accuracy. The findings also underline the problem with excessive information, such as when the products have too many details.

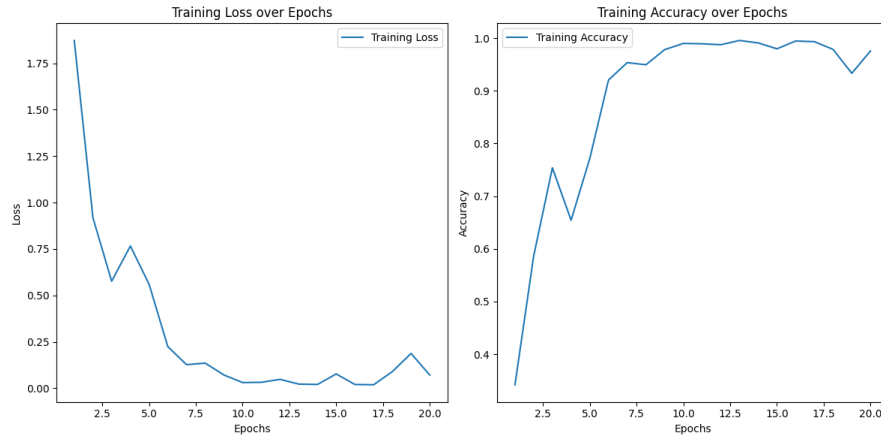


Figure 5.17: Training loss and accuracy for ResNet network trained with Original Dataset 2 modified

5.4.2 Dataset with metadata created via clustering

An additional test was carried out to assess the impact of bad clustering that could result from using the automatic clustering system. To do this, the dataset with the best classification results has six labels changed to simulate a simple misclustering. The model was trained, and the following results were obtained:

The model's performance was highly compromised, with only the six labels modified. From the graphs of the image 5.17, it's possible to see that the decrease in loss and increase in accuracy was much slower and with some instability during the process. It is also possible to confirm that the model has great difficulty in maintaining a stable accuracy during training, constantly oscillating to the point of worsening in some periods. The same is true of F1-Score.

These results show their impact in the test results, where the model only obtained an accuracy of 27.132%, a very low figure compared to the 54.191% obtained when the same dataset was completely well classified. In addition, the model obtained a loss of 4.45 and an F1-Score of 27.633%. The impact of the small change in the dataset is reflected in these values and the confusion matrix shown in fig 5.18 where a much more dispersed confusion matrix is shown, given the misclassification.

5.5 Synthetic Dataset Generation

In order to combat the poor performance of the clustering system and train a system that could also detect the position of objects to detect more than one in the same image.

By synthesizing images, the following examples were obtained:

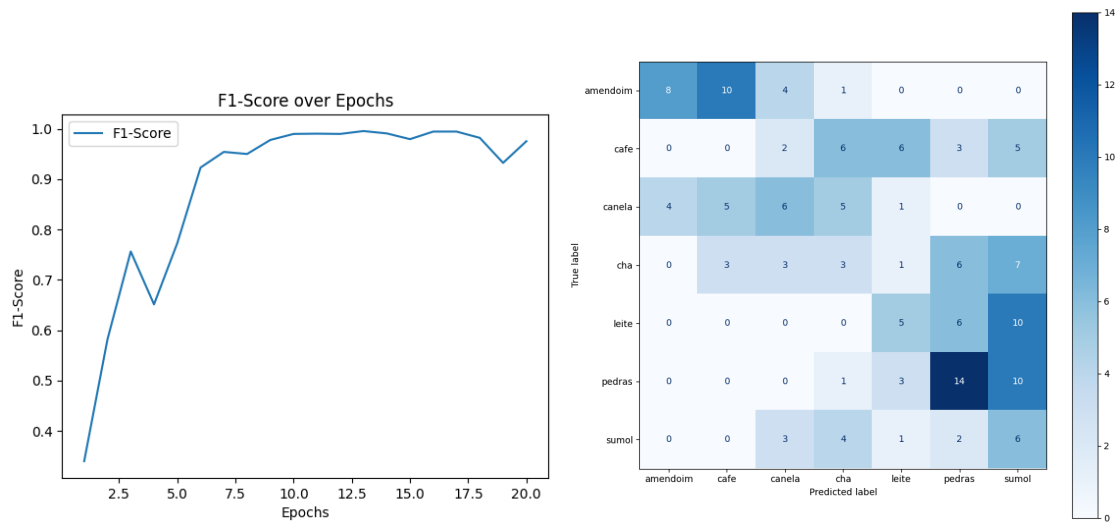


Figure 5.18: F1-Score and confusion matrix resultant from testing for ResNet Network trained with Original Dataset 2 modified



Figure 5.19: Synthetic images generated

The random positioning of the product in the image is unpredictable, given that there is no way of knowing the position or direction in which the consumer will insert the product when it is placed in the shopping cart. The synthetic dataset contained 400 images for training and 100 images for validation, annotated with bounding boxes and product labels.

Using the images obtained to train the YOLOv5 model, the results shown in image 5.20 were obtained from the validation dataset.

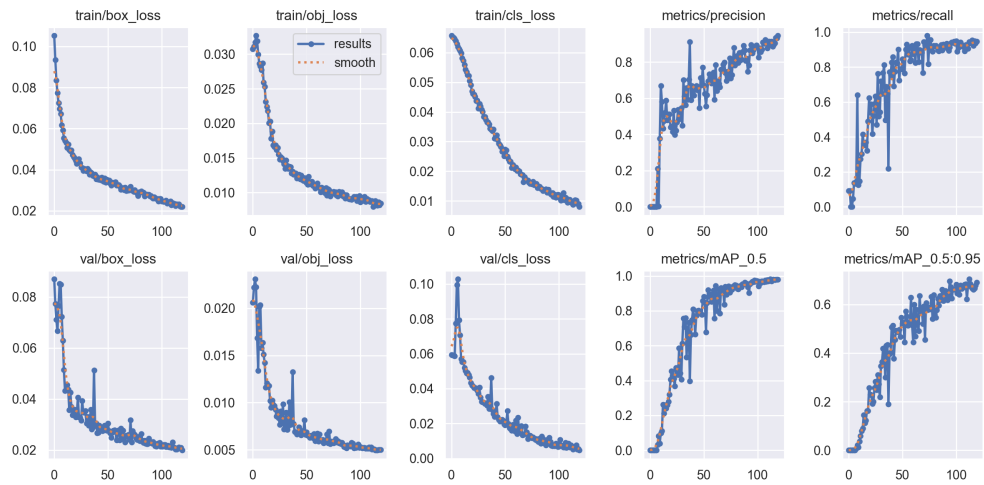


Figure 5.20: Metrics graphs resultant from training and validation on the synthetic images dataset

These metrics include the evolution of precision, which, although somewhat volatile, increases over the seasons until it evolves less and less, ending at 89.8%. The same goes for recall, which reaches 94.5% at the end of training. The loss of the box in training shows a very constant and rapid evolution at the beginning, starting to evolve more slowly around the thirtieth season. From then on, it continues to decrease, reaching a value of 0.02195 at the end of training. The evolution of the box loss in validation is very similar to that in training but with some oscillation. This is to be expected, given that the images in the validation dataset are new, and therefore, the expected performance is worse in this case. The mAP_0.5 increases rapidly until around epoch 50, where it begins to increase at a lower rate, and by the end of the training, it begins to stabilize at around 98.2%.

In the image 5.21, the confusion matrix shows that at the end of the training, the model is able to detect and classify the products with great reliability, although it has great difficulty in distinguishing the green apple from the normal pear. This can be confirmed in the image 5.22, where it's possible to confirm that in the two images where the green apple appears (the first and last in the grid), the model classifies the product twice, with both classifications. In that image, on the left are shown the validation images with Ground-Truth bounding boxes and labels, while on the left are the same images, but with the predicted bounding boxes and classifications.

These promising results demonstrate that with a small amount of images, which can be easily obtained, it is possible to create a complex dataset with complete metadata. It would not be possible to manually create a dataset with all the classifications and bounding boxes with this level of ease. However, the system's effectiveness has yet to be proven in a real environment to

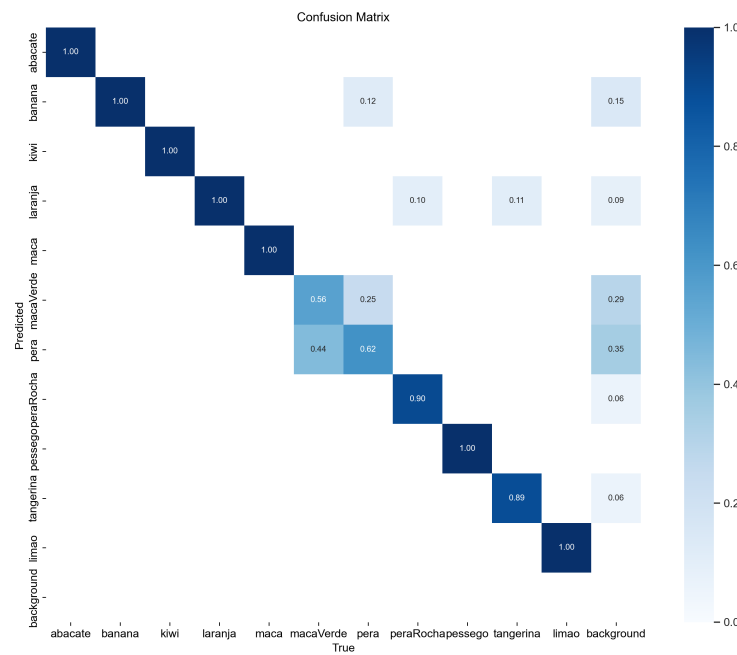


Figure 5.21: Confusion matrix generated from the validation dataset



Figure 5.22: Validation results

ensure this is the way forward.

Moreover, the integration of Mask R-CNN overcomes the limitations of the previous system by producing thorough datasets with bounding boxes. This improved approach not only streamlines training and validation processes but also highlights the potential of automated data generation methods in addressing the challenges of manual dataset creation. After validating the model's capability to classify and detect objects in controlled settings accurately, the next steps involve assessing its performance in real-world scenarios to confirm its reliability and practical usefulness.

Chapter 6

Conclusions

With this thesis, the necessary development began for a system that could integrate an intelligent shopping cart, where products could be inserted, detected, and classified using only computer vision methods.

The main challenge was the lack of a dataset incorporating products from the Portuguese domestic market and much of the development involved understanding how best to create one. Given the market volatility, it was also essential to ensure that the system used to create the dataset could be easily integrated into a system that could be accessed by a store assistant so that products could be updated and added.

The first step was to study the best alternative for the style of the dataset, whether it should have a white or random background, as the datasets found offered both alternatives. The results showed that a white background allows the classification system to learn the features more easily, particularly in situations where similar backgrounds are used and the algorithm can start to learn the background's features and not the desired product, substantially reducing performance.

With this in mind, three datasets were created, one with a simple background and the other two with plain white backgrounds, to test a cluster system that was created to aggregate all the similar images and thus facilitate the process of annotating them to create the metadata. An additional dataset with random backgrounds was also created to be used as a test dataset for the models trained with the dataset from the clustering system.

The clustering system did not prove to be efficient, even when different clustering algorithms and different feature extractors were used, with only one case in all the tests in which all the images were correctly aggregated. There were also a few cases with a small amount of misattribution. The only way to guarantee the effectiveness of this method is through human review, and to implement it, a user has to correct the clusters manually, making the process more time-consuming. After testing a dataset with poor labeling to train a CNN, it turned out that

even this small amount of misattribution is enough to seriously compromise the performance of a classification network since it uses wrongly annotated images, which leads to the weights being out of balance.

In any case, the datasets created were tested, with manual labeling, to ascertain the performance of the three and draw conclusions regarding their individual performance and comparisons between them. The Original Dataset 3 was revealed to have the best performance out of the three because of its simple white background and good distance to the camera, which allowed the model to extract good features to analyze.

Given the limitations, it was decided to test a system that generated the images synthetically rather than using human-captured ones. To do this, a dataset of fruits was created, given its complex morphology, which adds to the system's difficulty. Using a Mask R-CNN network, the different faces of some fruits were captured, and after segmenting the images, they were inserted into the background of supermarket scenes. This methodology improved over other studies, since it allowed the segmentation of objects with odd shapes. In this process, some augmentation was used to give the fruit different sizes and rotations, as well as some changes in perspective and color. A major advantage of this methodology is that, as the position of the product is chosen by the system and its dimensions are known, it is easy to register the bounding box to annotate the image and help with training. With the resulting images, part was used to train a YOLOv5 network and another small portion to validate the model, obtaining very promising results.

Throughout this study, various challenges emerged, particularly at the beginning, due to the absence of a labeled dataset. Regrettably, the clustering and classification approach employed proved to be inefficient and time-consuming. While it was feasible to organize and categorize the data using the clustering method, it did not yield consistent results suitable for training a model. Nonetheless, the volume and diversity required for strong model training might be supplied by synthetic dataset generation. The results demonstrated significant potential for achieving high classification accuracy by utilizing a synthetic dataset with an advanced neural network such as YOLO.

Based on these findings, the subsequent steps in the research to fulfill the stated project objective will involve expanding the synthetic dataset by capturing and segmenting a wide array of products available in the Portuguese market. Subsequently, this data will be augmented, followed by training a relatively lightweight object detection model that can be easily integrated into a shopping cart and tested in a real-world setting to ensure its efficacy in commercial environments.

Appendix A

Request for Photographic Record with Consent

The following four pages contain the document developed to request people to collect images to create the dataset, instructions for photographic capture and consent for the storage of the files.

CONSENTIMENTO INFORMADO, LIVRE E ESCLARECIDO

PARA UTILIZAÇÃO DE IMAGENS EM PESQUISA

Por favor, leia com atenção a seguinte informação. Se tiver alguma dúvida não hesite em solicitar mais informações. Se concorda com a proposta que lhe foi feita, queira assinar este documento.

Título da tese: “Reconhecimento automático de produtos para compras inteligentes”

Enquadramento: Este projeto enquadra-se numa dissertação de mestrado que está a ser desenvolvida orientada pela Faculdade de Engenharia da Universidade do Porto para a conclusão do Mestrado de Engenharia Electrotécnica e de Computadores, em âmbito empresarial na empresa “Retail Consult”.

Explicação sucinta do estudo: As compras inteligentes são uma área em crescimento que visa criar ferramentas para revolucionar a experiência de compra tanto de clientes quanto de vendedores. As vantagens para os clientes centram-se numa melhor experiência para uma compra mais rápida e com um acesso facilitado às melhores promoções; bem como recomendações de produtos para substituir os que estejam em falta. Para os vendedores, estes avanços prometem acesso às mais diversas estatísticas sobre os produtos que vendem e o local de venda, permitindo-lhes aumentar as suas vendas através da análise destes dados.

A visão computacional é uma dessas ferramentas. Utiliza inteligência artificial para analisar imagens, o que promete revolucionar a indústria ao reduzir as dificuldades presentes no mercado atual e ajudar nas tarefas monótonas e repetitivas. Embora seja muito promissora, também apresenta múltiplos desafios, nomeadamente no que diz respeito à identificação de produtos, dada a grande diversidade de embalagens e designs; podendo ser diferentes para o mesmo produto. Procurando formular um conjunto de dados de treino sólido e bem estruturado, aliado a uma rede neuronal aprimorada para analisar as características obtidas através das imagens capturadas, este trabalho procura obter um sistema capaz de rivalizar com as soluções já existentes.

Assim, o objetivo desta recolha é obter material fotográfico relativo a produtos vendidos em mercados de retalho, para poderem ser usados no treino deste novo sistema.

Condições: A sua participação é totalmente voluntária e isenta de qualquer tipo de prejuízo caso decida ou não participar ou desista do mesmo.

Confidencialidade e anonimato: As imagens, obtidas neste estudo, serão de acesso exclusivo do estudante e orientadores do projeto. No processo de tratamento de dados não

será possível identificar os respondentes, em virtude dos dados serem armazenados de forma agregada e sem identificação individual (garantido deste modo o anonimato);

Para qualquer esclarecimento de dúvidas, não hesite em contactar o estudante - Manuel Fernandes Sousa através do telemóvel: 925664787 ou através do e-mail: up201904610@up.pt.

Assinatura:

Declaro ter lido e compreendido este documento, bem como as informações verbais que me possam ter sido fornecidas pela pessoa que acima assina. Foi-me garantida a possibilidade de, em qualquer altura, recusar participar neste estudo sem qualquer tipo de consequências.

Desta forma, aceito participar neste estudo e permito a utilização dos dados referidos que, de forma voluntária e gratuita, forneço confiando que apenas serão utilizados para o presente estudo, tendo sido garantidas confidencialidade e anonimato.

Nome:

Assinatura:

Data: ____ / ____ / ____

Indicações para a Recolha Fotográfica

Procura-se obter fotografias de produtos comprados em qualquer supermercado localizado em **Portugal**.

Procure fotografar produtos em embalagens com diferentes características: cartão rígido, plástico rígido/maleável, vidro transparente/fosco.

É importante que as fotografias capturadas cumpram os seguintes requisitos:

- Encontrem-se num local bem iluminado;
- O fundo envolvente da fotografia seja de cor branca;
- A superfícies onde o produto se encontra pousado seja de cor branca;
- Uso primordial de um smartphone.

Cumprindo estes requisitos, solicitam-se **dois tipos** de fotografias:

- **Produto Individual:** as duas primeiras fotografias têm que ser correspondentes ao lado frontal e ao lado traseiro do produto, tirado à altura do mesmo. As restantes fotografias (entre 5 a 10) podem ser capturadas de qualquer ângulo, devendo apenas se garantir que a cor dos objetos envolventes ao produto na fotografia seja a cor branca.
- **Conjunto de Produtos:** Fotografias capturadas com as mesmas características supracitadas, em que se encontram apresentados alguns dos produtos selecionados para as fotografias de produtos individuais.

Como enviar as fotografias?

- Una as fotografias todas numa única pasta no computador;
- Aceda à pasta partilhada do google drive no seguinte link: <https://shorturl.at/jmvU8> ;
- Clique no botão carregar no canto superior esquerdo e selecione a opção carregar pasta;
- Selecione a pasta que contém as fotos;
- Aguarde o fim do carregamento - pode verificar o progresso na janela que aparece no canto inferior direito.

Obrigado!



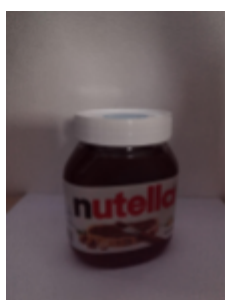
1ª Imagem



2ª Imagem



Exemplo de Restantes Imagens



Exemplo de Casos a Evitar



Exemplos de Fotografia com Múltiplos Produtos

Appendix B

Dash Interface

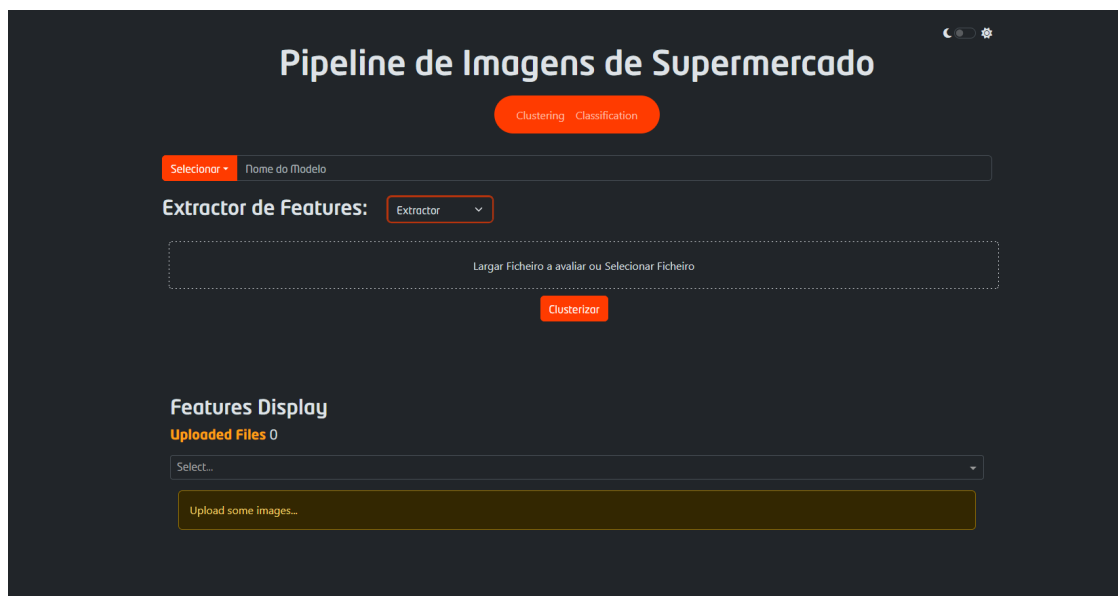


Figure B.1: Main dashboard page

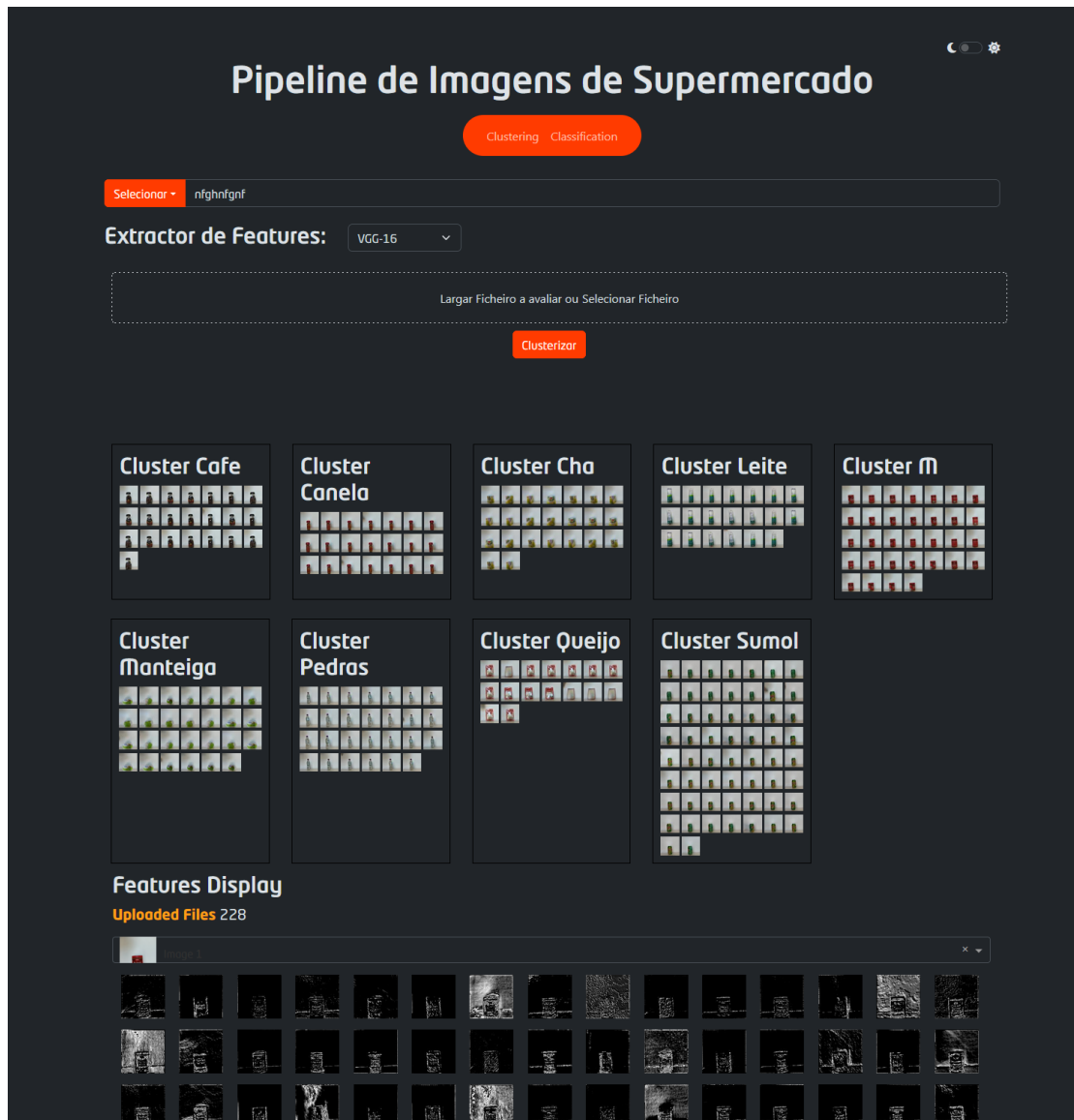


Figure B.2: Main dashboard page displaying results from a clustering test

The screenshot shows a web interface titled "Pipeline de Imagens de Supermercado". At the top right, there are icons for a moon (dark mode), a sun (light mode), and a gear (settings). Below the title, there is a horizontal bar with two tabs: "Clustering" (active) and "Classification". Below this bar, there is a form with several fields and buttons:

- A field labeled "Nome do Grupo de Clusters" with a "Selecionar" button on the left and an "Augment" button on the right.
- A section labeled "Rede de Classificação:" with a "Selecionar" dropdown menu and a "Classificar" button.
- A section labeled "Rede de Teste:" with a "Selecionar" dropdown menu.
- A section labeled "Dataset a Testar:" with a "Selecionar" dropdown menu.
- A dashed box containing the text "Largar Ficheiro a avaliar ou Selecionar Ficheiro" and a "Testar" button below it.

Figure B.3: Second dashboard page

References

- [1] Sofia Coloma, Carol Martinez, Bars Can Yalcin, and Miguel A. Olivares-Mendez. Enhancing Rover Teleoperation on the Moon With Proprioceptive Sensors and Machine Learning Techniques. *IEEE Robotics and Automation Letters*, 7(4):11434–11441, October 2022. URL: <https://ieeexplore.ieee.org/document/9857576/>, doi: 10.1109/LRA.2022.3198794.
- [2] Pengzhou Tang, Yu Guo, Han Li, Zhen Wei, Guanguan Zheng, and Jun Pu. Image dataset creation and networks improvement method based on CAD model and edge operator for object detection in the manufacturing industry. *Machine Vision & Applications*, 32(5):1–18, September 2021. Publisher: Springer Nature. URL: <https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib,uid&db=asn&AN=152088400&lang=pt-pt&site=ehost-live&scope=site,doi:10.1007/s00138-021-01237-y>.
- [3] Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, and V. Pattabiraman. Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, 8(1):66, December 2021. URL: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00434-w>, doi:10.1186/s40537-021-00434-w.
- [4] Zhengyu He. Deep Learning in Image Classification: A Survey Report. In *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pages 174–177, Guangzhou, China, December 2020. IEEE. URL: <https://ieeexplore.ieee.org/document/9421942/>, doi:10.1109/ITCA52113.2020.00043.
- [5] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks, January 2018. arXiv:1608.06993 [cs]. URL: <http://arxiv.org/abs/1608.06993>.
- [6] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, September 2020. arXiv:1905.11946 [cs, stat]. URL: <http://arxiv.org/abs/1905.11946>.
- [7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN, January 2018. arXiv:1703.06870 [cs]. URL: <http://arxiv.org/abs/1703.06870>.

- [8] Sanjit K. Roy, M. S. Balaji, and Bang Nguyen. Consumer-computer interaction and in-store smart technology (IST) in the retail industry: the role of motivation, opportunity, and ability. *Journal of Marketing Management*, 36(3-4):299–333, February 2020. URL: <https://www.tandfonline.com/doi/full/10.1080/0267257X.2020.1736130>, doi:10.1080/0267257X.2020.1736130.
- [9] Vatsalya Singhi and Kayalvizhi Jayavel. SMART SHOPPING SYSTEM FOR BILLING AUTOMATION. *Vol. No.*, (13), 2017. URL: https://www.ijaer.com/admin/upload/1508066858_Vatsalya_4.pdf.
- [10] Ceren Gulra Melek, Elena Battini Sonmez, Hakan Ayrar, and Songul Varli. Development of a Hybrid Method for Multi-Stage End-to-End Recognition of Grocery Products in Shelf Images. *Electronics*, 12(17):3640, August 2023. URL: <https://www.mdpi.com/2079-9292/12/17/3640>, doi:10.3390/electronics12173640.
- [11] Yuchen Wei, Son Tran, Shuxiang Xu, Byeong Kang, and Matthew Springer. Deep Learning for Retail Product Recognition: Challenges and Techniques. *Computational Intelligence and Neuroscience*, 2020:1–23, November 2020. URL: <https://www.hindawi.com/journals/cin/2020/8875910/>, doi:10.1155/2020/8875910.
- [12] Vânia Guimarães, Jéssica Nascimento, Paula Viana, and Pedro Carvalho. A Review of Recent Advances and Challenges in Grocery Label Detection and Recognition. *Applied Sciences*, 13(5):2871, February 2023. URL: <https://www.mdpi.com/2076-3417/13/5/2871>, doi:10.3390/app13052871.
- [13] Zhenyu Wu, Daiying Zhou, and Hong Hu. Blur Detection For Video Streams In The Compressed Domain.
- [14] Ayush Senapati, Atul Kumar Karn, Yash Bhardwaj, Jenicka. S, Padma Priya R, Ankit Shukla, and Shivam Arora. Identification of blurred objects in real time Video using deep learning neural networks. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–4, Kharagpur, India, October 2022. IEEE. URL: <https://ieeexplore.ieee.org/document/9984429/>, doi:10.1109/ICCCNT54827.2022.9984429.
- [15] Pramod Gupta and Naresh K. Sehgal. *Introduction to Machine Learning in the Cloud with Python: Concepts and Practices*. Springer International Publishing, Cham, 2021. URL: <https://link.springer.com/10.1007/978-3-030-71270-9>, doi:10.1007/978-3-030-71270-9.
- [16] Bikash Santra and Dipti Prasad Mukherjee. A comprehensive survey on computer vision based approaches for automatic identification of products in retail store. *Image and Vision Computing*, 86:45–63, June 2019. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0262885619300277>, doi:10.1016/j.imavis.2019.03.005.

- [17] Dongkuan Xu and Yingjie Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2):165–193, June 2015. URL: <http://link.springer.com/10.1007/s40745-015-0040-1>, doi:10.1007/s40745-015-0040-1.
- [18] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, April 2023. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0020025522014633>, doi:10.1016/j.ins.2022.11.139.
- [19] K.A. Abdul Nazeer, S.D. Madhu Kumar, and M.P. Sebastian. Enhancing the K-means Clustering Algorithm by Using a $O(n \log n)$ Heuristic Method for Finding Better Initial Centroids. In *2011 Second International Conference on Emerging Applications of Information Technology*, pages 261–264, Kolkata, India, February 2011. IEEE. URL: <http://ieeexplore.ieee.org/document/5734940/>, doi:10.1109/EAIT.2011.57.
- [20] Nahid Gholizadeh, Hamid Saadatfar, and Nooshin Hanafi. K-DBSCAN: An improved DBSCAN algorithm for big data. *The Journal of Supercomputing*, 77(6):6214–6235, June 2021. URL: <https://link.springer.com/10.1007/s11227-020-03524-3>, doi:10.1007/s11227-020-03524-3.
- [21] Dingsheng Deng. DBSCAN Clustering Algorithm Based on Density. In *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pages 949–953, Hefei, China, September 2020. IEEE. URL: <https://ieeexplore.ieee.org/document/9356727/>, doi:10.1109/IFEEA51475.2020.00199.
- [22] Douglas Reynolds. Gaussian Mixture Models. In Stan Z. Li and Anil Jain, editors, *Encyclopedia of Biometrics*, pages 659–663. Springer US, Boston, MA, 2009. URL: https://doi.org/10.1007/978-0-387-73003-5_196, doi:10.1007/978-0-387-73003-5_196.
- [23] Fatma Najar, Sami Bourouis, Nizar Bouguila, and Safiya Belghith. A Comparison Between Different Gaussian-Based Mixture Models. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 704–708, Hammamet, October 2017. IEEE. URL: <http://ieeexplore.ieee.org/document/8308357/>, doi:10.1109/AICCSA.2017.108.
- [24] Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *WIREs Computational Statistics*, 1(3):283–289, 2009. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.49>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.49>, doi:10.1002/wics.49.
- [25] Y. Ireaneus Anna Rejani and S. Thamarai Selvi. Early Detection of Breast Cancer using SVM Classifier Technique. *CoRR*, abs/0912.2314, 2009. arXiv: 0912.2314. URL: <http://arxiv.org/abs/0912.2314>.

- [26] Nick Guenther and Matthias Schonlau. Support Vector Machines. *The Stata Journal*, 16(4):917–937, 2016. _eprint: <https://doi.org/10.1177/1536867X1601600407>. URL: <https://doi.org/10.1177/1536867X1601600407>, doi:10.1177/1536867X1601600407.
- [27] Stephen Winters-Hilt and Sam Merat. SVM clustering. *BMC Bioinformatics*, 8(S7):S18, November 2007. URL: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-S7-S18>, doi:10.1186/1471-2105-8-S7-S18.
- [28] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application, June 2023. arXiv:2302.00487 [cs]. URL: <http://arxiv.org/abs/2302.00487>.
- [29] Niluthpol Chowdhury Mithun, Rameswar Panda, and Amit K. Roy-Chowdhury. Construction of Diverse Image Datasets From Web Collections With Limited Labeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(4):1147–1161, April 2020. URL: <https://ieeexplore.ieee.org/document/8640812/>, doi:10.1109/TCSVT.2019.2898899.
- [30] Seongdeok Bang, Francis Baek, Somin Park, Wontae Kim, and Hyoungkwan Kim. Image augmentation to improve construction resource detection using generative adversarial networks, cut-and-paste, and image transformation techniques. *Automation in Construction*, 115:103198, July 2020. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0926580519311653>, doi:10.1016/j.autcon.2020.103198.
- [31] Maggie Wigness, Bruce A. Draper, and J. Ross Beveridge. Efficient label collection for unlabeled image datasets. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4594–4602, Boston, MA, USA, June 2015. IEEE. URL: <http://ieeexplore.ieee.org/document/7299090/>, doi:10.1109/CVPR.2015.7299090.
- [32] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, April 2018. arXiv:1804.02767 [cs]. URL: <http://arxiv.org/abs/1804.02767>.
- [33] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10, Sukkur, March 2018. IEEE. URL: <https://ieeexplore.ieee.org/document/8346440/>, doi:10.1109/ICOMET.2018.8346440.
- [34] Monika Bansal, Munish Kumar, and Manish Kumar. 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. *Multimedia Tools and Applications*, 80(12):18839–18857, May 2021. URL: <https://link.springer.com/10.1007/s11042-021-10646-0>, doi:10.1007/s11042-021-10646-0.

- [35] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *CoRR*, abs/1803.01164, 2018. arXiv: 1803.01164. URL: <http://arxiv.org/abs/1803.01164>.
- [36] Devvi Sarwinda, Radifa Hilya Paradisa, Alhadi Bustamam, and Pinkie Anggia. Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer. *Procedia Computer Science*, 179:423–431, 2021. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1877050921000284>, doi:10.1016/j.procs.2021.01.025.
- [37] Yi Zhu and Shawn Newsam. DenseNet for dense flow. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 790–794, Beijing, September 2017. IEEE. URL: <http://ieeexplore.ieee.org/document/8296389/>, doi:10.1109/ICIP.2017.8296389.
- [38] Gao Huang, Shichen Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. CondenseNet: An Efficient DenseNet Using Learned Group Convolutions. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, Salt Lake City, UT, June 2018. IEEE. URL: <https://ieeexplore.ieee.org/document/8578389/>, doi:10.1109/CVPR.2018.00291.
- [39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs]. URL: <http://arxiv.org/abs/1505.04597>.
- [40] Miles McCrory and Spencer A. Thomas. Cluster Metric Sensitivity to Irrelevant Features, February 2024. arXiv:2402.12008 [cs, stat]. URL: <http://arxiv.org/abs/2402.12008>.
- [41] Salih Sarp, Murat Kuzlu, Yanxiao Zhao, Mecit Cetin, and Ozgur Guler. A comparison of deep learning algorithms on image data for detecting floodwater on roadways. *Computer Science and Information Systems*, 19(1):397–414, 2022. URL: <https://doiserbia.nb.rs/Article.aspx?ID=1820-02142100058S>, doi:10.2298/CSIS210313058S.
- [42] 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):279, January 2021. URL: <https://www.mdpi.com/2079-9292/10/3/279>, doi:10.3390/electronics10030279.
- [43] Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey, November 2019. arXiv:1911.12116 [cs]. URL: <http://arxiv.org/abs/1911.12116>.

- [44] Kanupriya Mittal and V. Mary Anita Rajam. Computerized retinal image analysis - a survey. *Multimedia Tools and Applications*, 79(31-32):22389–22421, August 2020. URL: <https://link.springer.com/10.1007/s11042-020-09041-y>, doi:10.1007/s11042-020-09041-y.
- [45] Shikha Dubey, Abhijeet Boragule, and Moongu Jeon. 3D ResNet with Ranking Loss Function for Abnormal Activity Detection in Videos, February 2020. arXiv:2002.01132 [cs, eess]. URL: <http://arxiv.org/abs/2002.01132>.
- [46] Alina Matei, Andreea Glavan, and Estefania Talavera. Deep learning for scene recognition from visual data: a survey, July 2020. arXiv:2007.01806 [cs]. URL: <http://arxiv.org/abs/2007.01806>.
- [47] Oliver Rippel, Patrick Mertens, Eike Konig, and Dorit Merhof. Gaussian Anomaly Detection by Modeling the Distribution of Normal Data in Pretrained Deep Features. *IEEE Transactions on Instrumentation and Measurement*, 70:1–13, 2021. URL: <https://ieeexplore.ieee.org/document/9493210/>, doi:10.1109/TIM.2021.3098381.
- [48] Sakorn Mekruksavanich. The Smart Shopping Basket Based on IoT Applications. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 714–717, Beijing, China, October 2019. IEEE. URL: <https://ieeexplore.ieee.org/document/9040750/>, doi:10.1109/ICSESS47205.2019.9040750.
- [49] Marco Leo, Pierluigi Carcagni, and Cosimo Distante. A Systematic Investigation on end-to-end Deep Recognition of Grocery Products in the Wild. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7234–7241, Milan, Italy, January 2021. IEEE. URL: <https://ieeexplore.ieee.org/document/9413250/>, doi:10.1109/ICPR48806.2021.9413250.
- [50] Namitha James. Automated Checkout for Stores: A Computer Vision Approach. *Revista Gestão Inovação e Tecnologias*, 11(3):1830–1841, June 2021. URL: <http://www.revistageintec.net/index.php/revista/article/view/2053>, doi:10.47059/revistageintec.v11i3.2053.
- [51] Ren-Yi Lee, Tong-Yuen Chai, Sing-Yee Chua, Yen-Lung Lai, Wai Sim Yee, and Su-Cheng Haw. Cashierless Checkout Vision System for Smart Retail using Deep Learning. *Journal of System and Management Sciences*, August 2022. URL: <http://www.aasmr.org/jsms/Vol12/JSMS%20august%202022/Vol.12.No.04.15.pdf>, doi:10.33168/JSMS.2022.0415.
- [52] Kostas Georgiadis, Giorgos Kordopatis-Zilos, Fotis P. Kalaganis, Panagiotis Migkotzidis, Elisavet Chatzilari, Valasia Panakidou, Kyriakos Pantouvakis, Savvas Tortopidis, Symeon Papadopoulos, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Products-6K: A Large-Scale Groceries Product Recognition Dataset, February 2021. URL: <https://doi.org/10.5281/zenodo.4428917>, doi:10.5281/zenodo.4428917.