UNIVERSIDADE D
COIMBRA

Daniel Roque Pereira

TOWARDS AN AUTOMATIC TRAINING SYSTEM USING FEW-SHOT OBJECT DETECTION

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor João Nuno Gonçalves Costa Cavaleiro Correia and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

January of 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Daniel Roque Pereira

Towards an automatic training system using Few-shot Object Detection

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Professor João Nuno Gonçalves Costa Cavaleiro Correia, Rui Mendes and Pedro Miguel Felizardo Antunes presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Daniel Roque Pereira

Em direção a um sistema de treino automático para deteção de objetos usando poucos exemplos

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes, orientada pelo Professor João Nuno Gonçalves Costa Cavaleiro Correia,Rui Mendes e Pedro Miguel Felizardo Antunes e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro 2023

Acknowledgements

Gostaria de agradecer, primeiramente, aos meus orientadores neste projeto: ao Professor João Nuno Correira, pela sua paciência, disponibilidade e grande ajuda nesta etapa final; ao Rui Mendes, por toda a sua atenciosidade e paciência também. Sem ele, o meu trabalho no último mês não teria sido possível. Queria também estender este agradecimento ao Pedro Miguel Felizardo Antunes e à Redlight por me acolherem e terem sido inacreditavelmente simpáticos e amigáveis durante a minha estadia.

Um agradecimento do fundo do meu coração aos meus amigos por todo o apoio dado neste processo. Amigos estes que não só me encorajaram a fazer pausas do trabalho intenso, mas que também respeitavam a minha decisão de ficar em casa caso necessário, sem me pressionar.

Ultimamente, mas não menos importante, gostaria de expressar com todo o meu ser a minha gratidão para com a minha família pelo tremendo apoio e afeto que recebi, não apenas para este trabalho específico, mas desde o início da minha vida. Um obrigado especial aos meus pais, que sempre estiveram aqui para mim e que me apoiaram por este árduo caminho e respeitaram as minhas decisões. São eles a maior chave do meu sucesso. Ter uma mãe que prepara refeições independentemente da hora, seja porque estava completamente desgastado ou porque fiz uma noitada, fez o impossível parecer mais fácil. Obrigado a todos vocês que me impediram de me ir abaixo quando tudo parecia perdido!

Há ainda um reconhecimento especial que gostaria de fazer para esta tese. Há indivíduos, que embora já não estejam presentes, continuam a viver nos nossos corações. Ao meu querido cão, Benny, que infelizmente morreu no dia da minha apresentação intermédia, gostava de ter passado mais tempo de qualidade contigo no teu último suspiro. Espero que estejas num lugar melhor...

Finalmente, aos meus avós, com os quais cresci e que fizeram de mim um homem melhor. Ao meu avô "Zé", que nos seus últimos dias expressou o seu arrependimento de não estar cá para me ver na universidade, e ao meu avô "Tonito", que olhava por mim mais do que ninguém. Onde quer que estejam, espero que estejam orgulhosos de mim. Estou no final de terminar o meu progresso, e depois de toda a alegria, tristeza, sofrimento, euforia que passei nestes últimos anos, é com todo o meu coração e alma que dedico esta tese a **VOCÊS**.

Abstract

Companies supplying large global supermarket networks face challenges in managing inventory and ensuring timely shipments. These difficulties can arise from delays in restocking products, management issues, or problems with the timely delivery of shipments. Given the inefficiency of assigning a company worker to monitor each store's processes, *Brands&Ninjas* (B&N) has proposed a framework for detecting objects on shelves through photography. However, the current framework demands a substantial number of images for accurate object detection and incurs significant costs in terms of human labor for annotation, which can also introduce errors.

In order to streamline the onboarding process, reduce human labor, and improve scalability, this dissertation exploits the evolving field of Few-shot Object Detection (FSOD) and explores the Automatic Annotation (AA) field. AA employs computer algorithms for efficient and accurate object labeling, while FSOD, particularly meta-learning FSOD, focuses on detecting objects with limited training examples. By combining these techniques, we aim to create a scalable, time-efficient, and error-free tool suitable for automating onboarding processes. This endeavor finds relevance in addressing challenges faced by companies managing global supermarket networks, particularly in inventory management and shipment tracking. The proposed framework involves object detection through photography, accompanied by automated annotation and model training, enhancing the efficiency of the existing system. After an in-depth review of the State of the Art, we proposed an architecture to incorporate both of these techniques into a unified process. However, we soon discovered that the process was not feasible, as FSOD and AA are processes that demonstrate significant inherent incompatibility which complicates their coexistence. Consequently, we focused entirely on creating an FSOD system for automatic training.

We evaluated the capabilities of the YOLOv8 and *Detectron2* (d2)'s *Faster-RCNN* (FRCNN) models to develop a model that could meet our expectations without significant drawbacks in performance. In this dissertation, we implemented the FSOD technique across 2 different object detection State of the Art (SOTA) models in order to be able to detect objects on shelves with few training instances. Based on our results, we concluded that the approach is viable and the proof concept ends up working, but there is still significant room for FSOD to evolve and we leave key points to be explored for further progression and enhancement on it.

Keywords

Computer Vision, Machine Learning, Artificial Intelligence, Deep Learning, Neural Networks, Object Detection, Automatic Image Annotation, Few-shot Object Detection

Resumo

As empresas que fornecem redes globais de supermercados enfrentam desafios na gestão de inventário e na garantia de entregas pontuais. Estas dificuldades podem surgir devido a atrasos na reposição de produtos, problemas de gestão ou questões relacionadas com a entrega atempada de encomendas.

Dada a ineficiência de atribuir um trabalhador da empresa para monitorizar os processos de cada loja, a B&N propôs uma ferramenta para detetar objetos em prateleiras através de fotografias. No entanto, a ferramenta atual exige um número substancial de imagens para deteção precisa de objetos e acarreta custos significativos em termos de trabalho humano para anotação, o que também pode introduzir erros.

Com o objetivo de otimizar o processo de integração, reduzir a mão de obra humana e melhorar a escalabilidade, esta dissertação explora o campo em evolução da Detecção de Objetos com Poucos Exemplos (FSOD) e investiga o campo de Anotação Automática(AA). AA utiliza algoritmos de computador para uma etiquetagem eficiente e precisa de objetos, enquanto o FSOD, em particular o FSOD baseado em meta-aprendizagem, concentra-se na deteção de objetos com um número limitado de exemplos de treino. Ao combinar estas técnicas, temos como objetivo criar uma ferramenta escalável, eficiente em termos de tempo e sem erros, adequada para automatizar os processos de integração. Este empreendimento é relevante para enfrentar os desafios enfrentados pelas empresas que gerem redes de supermercados globais, especialmente na gestão de inventário e no acompanhamento de envios. A ferramenta proposta envolve a deteção de objetos através de imagem, acompanhada pela anotação automática e treino do modelo, melhorando a eficiência do sistema existente.

Após uma revisão aprofundada do Estado da Arte(SOTA), propusemos uma arquitetura para incorporar ambas as técnicas num processo unificado. No entanto, rapidamente descobrimos que o processo não era viável, uma vez que o FSOD e o AA são processos que demonstram uma incompatibilidade significativa, o que complica a sua coexistência. Como resultado, concentramo-nos inteiramente na criação de um sistema FSOD para treino automático.

Para este fim, investigamos as capacidades dos modelos YOLOv8 e FRCNN do d2 para desenvolver um modelo que pudesse corresponder às nossas expetativas sem desvantagens significativas em termos de desempenho. Nesta dissertação, implementamos a técnica FSOD em dois diferentes modelos SOTA de deteção de

objetos, a fim de ser capaz de detetar objetos em prateleiras com poucas instâncias de treino. Com base nos nossos resultados, concluímos que a abordagem é viável e que a prova de conceito funciona, mas que ainda há espaço significativo para o FSOD evoluir, e deixamos pontos chave a serem explorados para uma maior progressão e aprimoramento do mesmo.

Palavras-Chave

Visão por computador, Aprendizagem máquina, Inteligência Artificial, Rede Neuronal, Aprendizagem profunda, Deteção de objetos, Anotação automática de imagem, Deteção de objetos few-shot

Contents

Acronyms

AA Automatic Annotation.

B&N *Brands&Ninjas*.

CNN Convolutional Neural Networks.

CV Computer Vision.

d2 *Detectron2*.

DA Data Augmentation.

DNNs Deep Neural Networks.

FRCNN *Faster-RCNN*.

FSOD Few-shot Object Detection.

IoU Intersection over Union.

mAP Mean Average Precision.

ML Machine Learning.

OD Object Detection.

RoI Region of Interest.

RPN Region Proposal Network.

SNNs Shallow Neural Networks.

SOTA State of the Art.

List of Figures

List of Tables

Chapter 1

Introduction

Companies from all over the world that supply big commercial networks struggle with stock management and shipment. Either the product is not restocked in time due to mishaps or management flaws or the shipments are not delivered in time.

Brands&Ninjas (B&N) is a company that hired *Redlight Software* to create a software product that can help minimizing these problems. The proposed solution is a crowdsourcing platform for brands to manage stock, monitor prices, and scan shelves to ensure inventory is restocked according to schedule. The platform utilizes an app and its users, referred to as *ninjas*, who are sent on missions to complete these tasks. Currently, the data collected from these missions is manually validated. It is crucial to consider that *ninjas* may fail to deliver the correct answer to questions asked for the mission. The goal is subsequently to make a system that automatically validates all the images taken by the *ninjas*.

The current study focused on the specific case of missions assigned by the brand *Delta Cafés*, having already developed a system capable of detecting 5 different types of coffee from the *Delta Cafés* brand: *Qalidus*,*Qaracter*, *EpiQ*, *Portugal* and *Angola*

However, the current system still has the disadvantage of forcing the developers in charge of training the model into the burden of manually annotating a high quantity of images. Ninjas are also responsible for part of the validation of the framework's detection results by inspecting the taken photo and providing information as following:

- The *ninja* confirms that the product is on the shelf and corresponds to the correct answer or depicts that the product is on the shelf while it is not.
- The *ninja* states that the product is not on the shelf when it actually is or

denies that the product is not on the shelf and it corresponds to the correct answer.

1.1 Context

For the purpose of training a model with fewer images and to deprive the *Ninjas* from the burden of manually annotating there are two key concepts: FSOD and Automatic Annotation (AA).

AA is the process of using computer algorithms to automatically identify and label objects or features within an image. The State of the Art (SOTA) in AA is constantly evolving which means there are several commercial tools available that can achieve high-accuracy results.

FSOD is a task that aims to detect objects within an image or video with a very limited number of training examples. FSOD can be divided into two main categories: transfer FSOD and meta-learning FSOD. Transfer FSOD is the process of adapting a pre-trained object detector to new classes with limited examples, while meta-learning FSOD is the process of training a model to quickly adapt to new objects with very few examples by learning a meta-representation or an adaptive optimizer.

When using transfer FSOD, it is important to note that Deep Neural Networks (DNNs) are often used, which can lead to overfitting if a sufficient number of examples are not provided which points us to prefer the usage of meta-learning FSOD.

We will use the combination of both these two techniques to create a more scalable, less time consuming and human error free tool that is suitable for an automatic onboarding process that any user can experiment.

1.2 Objectives

The objective is to develop a process of image annotation and automatic training for object detection where the system is automated in a way that any client can input a few examples of their desired object and the system will instantly start to learn how to recognize it in pictures captured by the *ninjas* without any manual effort.

An automatic onboarding system is to be developed by recognizing the images via the combination of tasks like AA and FSOD. Thus,we intend to replace the previously used YOLOv4 by a model capable of training accurately with less examples using the *Detectron2* framework and find ourselves a software of automatic annotation that is suitable to integrate in our pipeline.

1.3 Methodology and Planning

1.3.1 First iteration - September 2022 to May 2023

We plan on using an iterative waterfall in order to combine the sequential process of the traditional waterfall model with the flexibility of iterative development. It follows a structured process of planning, design, development, testing, and deployment, with a focus on continuous testing and refinement throughout the development cycle. This way, we can allow for feedback to be incorporated and changes to be made throughout the project, resulting in a higher-quality final product.

The *Gantt* chart in Figure 1.1 demonstrates the timeline that the project is supposed to follow.

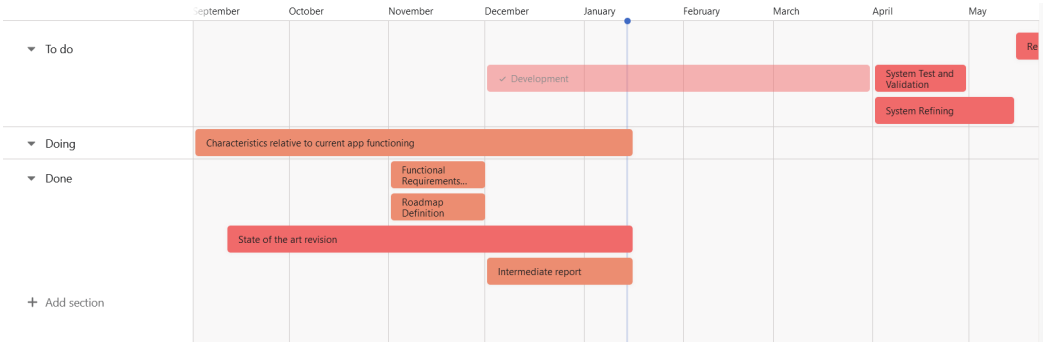


Figure 1.1: Project timeline

- **Characteristics relative to current app functioning** - Consists of understanding and reading about the work done on the previous framework. 11 Sep. - 16 Jan. 2023
- **Functional Requirements and Architecture** - Definition of the system's requirements and architecture 1 Nov. 2022 - 1 Dec. 2022
- **Roadmap definition** - This timeline itself 1 Nov. 2022 - 1 Dec. 2022

- **State of the art revision** - Research about key concepts and their SOTA 11 Sep. 2022 - 16 Jan. 2023
- **Intermediate report**. 15 Dec, 2022 - 16 Jan. 2023
- **Development phase of the system based on the current framework**. 1 Dec. 2022 - 31 Mar. 2023
- **System test and validation after the development phase**. 1 Apr. 2023 - 1 May 2023
- **Result refining after concluding the system testing**. 1 Apr. 2023 - 15 May 2023
- **Final report writing**. 15 May 2022 - 31 Jun. 2022

1.3.2 Second and final iteration - May 2022 to September 2023

As explained in previous chapters, this work suffered an abrupt turn of events and we ended up focusing full time on FSOD once we learned the impossibility of doing AA for our purpose. The process was therefore simplified into the following events:

- **Replication of the current Framework**
- **Few Shot Object Detection with *Detectron2's Faster-RCNN***
- **Few Shot Object Detection with *YOLOv8***
- **Standard Model training for *Faster-RCNN***
- **Result refining**
- **Final report writing**

1.4 Outline

The rest of this document is split between 6 different chapters:

2: The background chapter where we explain the main concepts for the reader to be able to understand chapter 3.

3: The State of the Art chapter where we introduce the most advanced concepts in order to develop our work.

4: The Framework chapter where we display the previously existing framework and the prototype of the framework to be created.

5: The Experimentation chapter where we talk about our dataset, display and discuss results.

6: The conclusion and future work section where we finalize with a resume of the previous chapters,take some final conclusions and project what more there is to be done in the future.

Chapter 2

Background

In this chapter, we delve into the foundational concepts of machine learning, Computer Vision (CV), and object detection. We trace the evolution of these fields and highlight their significance as the foundation of our research.

2.1 Machine Learning

This section will give an overview about the definition of Machine Learning (ML) as well as the critical subcategories and techniques necessary for the better understanding of this dissertation.

ML is a subfield of computer science that focuses on the development of systems capable of learning and improving from data. It leverages algorithms and statistical models to enable computers to recognize patterns, make decisions, and perform tasks without explicit programming. ML has evolved into a diverse field with various subcategories, including supervised and unsupervised learning, reinforcement learning, and more[21].

Supervised learning is a core subcategory of ML. In supervised learning, the algorithm is provided with a dataset that includes input data and corresponding target labels or class labels. The primary goal of supervised learning is to learn a mapping or function that can accurately classify or predict new, unseen data based on the patterns it has identified in the training data[21]:. To facilitate supervised learning, datasets are typically divided into three subsets[8]:

- **Training Set:** This set is used to train the machine learning model. The model learns from the input data and their associated target labels, gaining the knowledge needed for classification or prediction.

- **Validation Set:** During the training process, a portion of the dataset is reserved for validation. The validation set helps estimate the model's performance, including its accuracy and generalization capabilities, by assessing its predictions against known target labels.
- **Test Set:** The test set serves as an independent dataset used to evaluate the model's performance after it has been trained and validated. It allows for an unbiased assessment of how well the model generalizes to new, unseen data.

In the pursuit of enhanced accuracy and performance in machine learning tasks, raw data often undergoes preprocessing. Preprocessing encompasses several essential techniques:

- **Data Cleaning:** Data cleaning involves the identification and removal of outliers, erroneous data points, and missing values. Ensuring that the data is of high quality is critical for training robust models.
- **Data Transformation:** Data transformation techniques, such as normalization and scaling, are employed to bring the data into a consistent range of values. This ensures that the machine learning algorithm converges efficiently during training.
- **Data Augmentation (DA):** DA is a pivotal component, especially in deep learning model training. DA techniques aim to increase the amount of available training data by applying various transformations to the existing dataset. These transformations can include random cropping, flipping, rotation, and scaling.

Data augmentation broadly falls into two categories[36]:

- **Data Warping:** Data warping techniques modify the appearance of data without altering their class labels. These techniques, such as random cropping, flipping, rotation, and scaling, enhance the model's robustness to changes in position, orientation, and scale.
- **Oversampling:** Oversampling techniques aim to increase the number of training samples, particularly for minority classes. Methods like bootstrapping and Synthetic Minority Over-sampling Technique (SMOTE)[2] generate additional examples, balancing the class distribution and improving model performance.

The range of techniques for DA can go from easily explainable techniques such as horizontal flipping, random cropping, and color augmentation (Fig. 2.1) to more complex ones like mixing pixels or using Generative Adversarial Networks(GANs) [27].

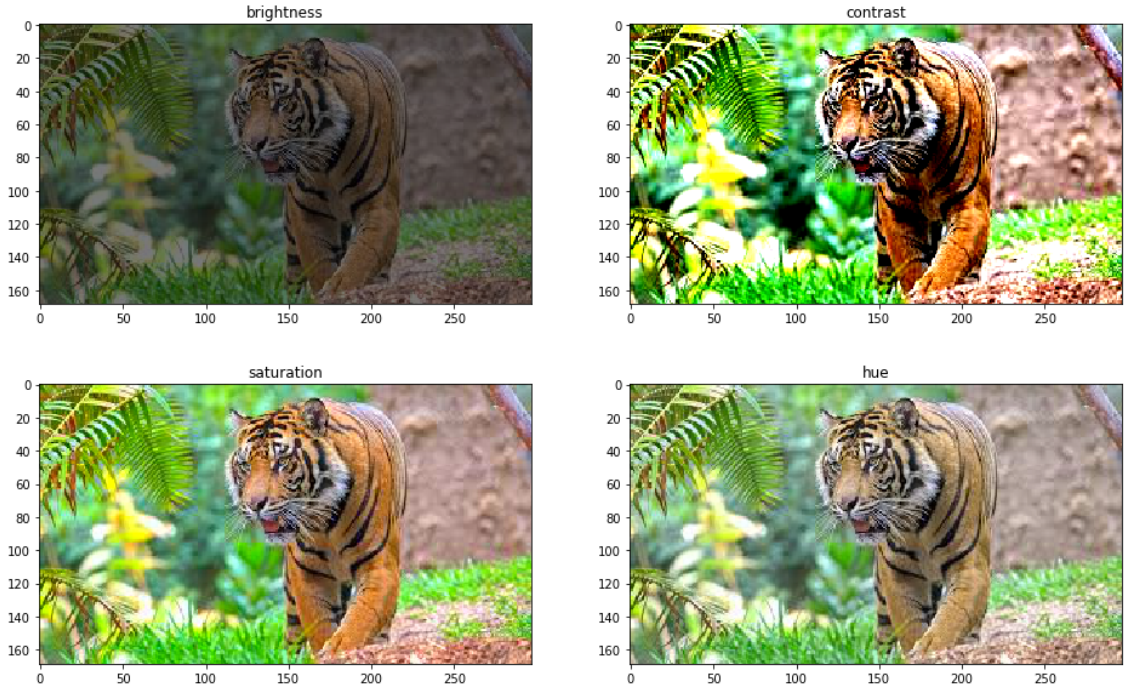


Figure 2.1: Color DA (Brightness, Contrast, Saturation) [30].

The selection of data augmentation techniques typically relies on the nature of the machine learning task at hand and the inherent characteristics of the dataset. This consideration brings us to our particular task: object detection Object Detection (OD), a computer vision CV task,both to be described in the next chapters.

2.2 Computer Vision

CV, a field nestled at the intersection of computer science and artificial intelligence, has undergone a remarkable evolution since its inception in the 1950s. This means the field has had various definitions throughout the years. We will delve into a brief moment of its story in order to elucidate the readers on the macro-scale of our work.

- **Early Concepts (1950s-1960s):** The foundations of CV were laid by visionaries like Oliver Selfridge[26] and Frank Rosenblatt[25]. They explored the

uncharted territory of pattern recognition and self-learning systems, setting the stage for the development of computational vision.

- **Block World (1960s):** In the 1960s,the father of CV Lawrence Roberts[24] and B. K. P. Horn embarked on a mission to create CV systems capable of recognizing objects in a simplified "block world" This marked a significant stride towards object recognition.

- **The Hough Transform (1962):** Paul Hough introduced the Hough Transform, a mathematical technique for detecting lines and shapes in images. This innovation provided a fundamental tool for image analysis and object recognition.

- **Pioneering Research (1970s):** During this era, Richard Duda and Peter Hart[5] unveiled the "Princeton Recognition Engine" a groundbreaking project that brought CV into sharper focus. It laid the groundwork for further exploration in the field.

- **Machine Learning Revolution (2000s):** The dawn of the 21st century witnessed a paradigm shift in CV. Yann LeCun, Geoffrey Hinton, and Yoshua Bengio[16] championed the rise of machine learning, particularly deep learning and Convolutional Neural Networks (CNN), which became the bedrock of modern CV.

- **The Scale-Invariant Feature Transform(2004):** David G. Lowe's SIFT algorithm[20], introduced in 2004, revolutionized feature-based object recognition and image matching. Its scale-invariant properties made it a cornerstone in CV applications.

- **ImageNet and Deep Learning Breakthroughs (2010s):** In 2010, Fei-Fei Li and her team initiated the ImageNet Large Scale Visual Recognition Challenge[4], catapulting deep learning into the spotlight. Notable contributions include AlexNet by Alex Krizhevsky, VGGNet by Karen Simonyan and Andrew Zisserman, and ResNet by Kaiming He, which pushed the boundaries of image classification.

- **Autonomous Vehicles and Robotics (2010s):** The 2010s also witnessed substantial strides in CV for autonomous vehicles and robotics. Researchers like Sebastian Thrun[17] made groundbreaking contributions, enabling machines to perceive and navigate their environments with unprecedented precision.

- **Augmented Reality (AR) and Virtual Reality (VR):** AR and VR technologies, immersive and transformative by nature, lean heavily on CV. A multitude of authors and researchers, often affiliated with companies like Magic Leap and Oculus, have played pivotal roles in advancing these captivating domains.

- **Current Trends (2020s and Beyond):** As we step into the 2020s and beyond, CV continues to evolve. Real-time object tracking, semantic segmentation, and human pose estimation represent some of the current frontiers. A diverse community of researchers and engineers is tirelessly working to unlock new horizons in this dynamic field.

CV is then a specialized field dedicated to enabling machines to interpret and comprehend visual data from the real world through the utilization of methodologies derived from computer science, physics, mathematics, and engineering. Within the realm of CV, the primary objectives encompass Image Detection, Image Segmentation, Object Detection, and Semantic Segmentation. For the scope of our research, we will be placing particular emphasis on OD as its central focus.

For this task, there are some several CV concepts that need to be introduced firstly for a better understading of our work like CNN and Region Proposal Network (RPN)

A CNN is a multi-layer Deep Neural Networks (DNNs) designed to exclusively process image data composed of convolutional layers, pooling layers, and fully-connected layers [16]:

- The **convolutional layers** are used to learn features from the image data by applying a set of filters to the input image.

- The **filters** are designed to detect specific patterns in the image, such as edges, corners, and textures.

- The **pooling layers** are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers while maintaining the most important information.

- The **fully-connected layers** are used to classify the image into a specific category or label.

This type of architecture can be observed in 2.2.

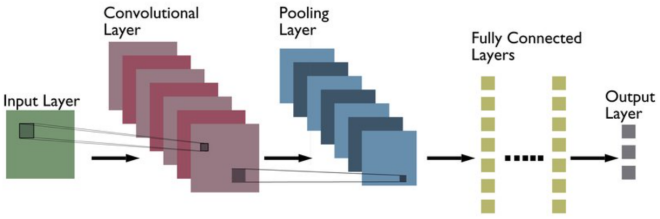


Figure 2.2: CNN typical architecture [14]

A RPN is a CNN that predicts object boundaries and objectness scores at every position in an image. It is trained to generate high-quality region proposals and can be integrated into a single network with algorithms like Fast R-CNN [6] by sharing their convolutional features. The RPN component guides the unified network to relevant areas of the image using attention mechanisms. RPNs are designed to predict region proposals effectively across a range of scales and aspect ratios using anchor boxes as references at different scales and ratios. This method can be thought of as a pyramid of regression references, which avoids the need to enumerate images or filters with various scales or aspect ratios [23].

2.2.1 Object Detection

Object detection is a field of CV that aims to locate and identify objects within an image or video. It is a technique that allows a model to understand objects' presence, location, and shape in an image or video. Object detection aims to predict a bounding box around the objects of interest in the image, along with a class label that describes the object.

There are different techniques for object detection, but the most common ones are based on CNN. These techniques can be broadly categorized into two groups: two-stage methods and single-stage methods (Figure 2.3):

- Two-stage methods, such as R-CNN[7], Fast R-CNN[6], and Faster R-CNN[23], first generate region proposals and then classify each proposal to check whether it contains an object or not.

- On the other hand, one-stage methods, such as *YOLOv7* [31] and SSD [19], predict class scores and bounding boxes directly from the feature maps of the CNN. Single-stage methods are typically faster than two-stage methods but may have lower accuracy.

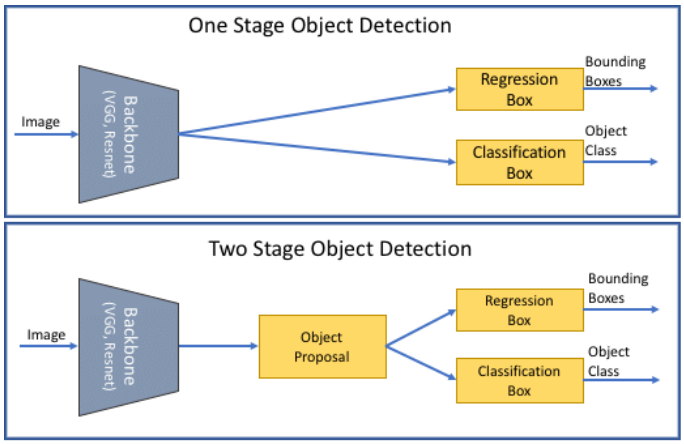


Figure 2.3: One Stage vs Two Stage Object Detectors

For further details on object detection, the interested reader is referred to this survey on object detection [37].

2.3 Summary

In this chapter, we have introduced the foundational concepts of ML,CV and OD.

ML is a computer science field that empowers computers to learn from data and make predictions or decisions without explicit programming. It involves crucial steps like data preprocessing and data augmentation (DA) to enhance the quality of training data.

CV, a specialized field at the intersection of computer science, physics, mathematics, and engineering, enables computers to interpret and understand visual information from the real world.

OD) is a specific CV technique focused on locating and identifying objects within images or videos. It accomplishes this task by predicting bounding boxes around objects and assigning class labels to them.

With this foundational understanding of ML, CV, and OD, we are now prepared to delve deeper into more complex concepts in the upcoming chapters.

Chapter 3

State of the Art

With the foundational concepts in place, we can now delve into the key concepts central to our work, namely Automatic Annotation (AA) and Few-shot Object Detection (FSOD).

In the context of our investigation, which aims to develop an AA and training system, a primary objective is to minimize the quantity of annotated examples necessary to train the model. This approach is geared towards reducing project requirements and costs while enhancing scalability.

The subsequent sections of this chapter provide comprehensive insights into crucial concepts and domains that are pivotal for our approach. These definitions predominantly concentrate on image-related aspects, even though many of the techniques mentioned also have applications in text and video domains.

3.1 Object Automatic Annotation for Images

AA is the process of automatically assigning metadata to images or videos. This metadata, generated through CV techniques, includes text or keywords used for object detection, including bounding box coordinates and labels. Bounding boxes are rectangular frames drawn around objects to define the Region of Interest (RoI).

Standard annotation methods often rely on manual human labor to annotate a large number of images, which is time-consuming and expensive. Additionally, manual labeling can introduce subjectivity, as annotations may vary from person to person. To address these challenges, automatic image annotation, also known as auto-annotation or linguistic indexing, has emerged[18].

Most AA algorithms involve extracting features from training and test images, creating annotation models based on the training data, and then generating annotations for test images[3]. The quality of these annotations depends on both the images provided for training and testing and the model used. Crowdsourcing has played a role in collecting a large volume of annotations through platforms like Amazon Mechanical Turk, significantly reducing the time required for dataset collection by involving numerous annotators[12].

Despite its advantages, AA faces several challenges:

- Algorithms may struggle to determine which region of an image corresponds to a specific tag, as a tag can apply to a region or the entire image.
- Low-level features may not capture the precise meaning of an image, leading to errors in measuring similarity between two images.
- Training datasets may contain outliers or redundant tags, affecting model performance.
- The "semantic gap" between low-level image features and high-level concepts or objects complicates annotation[18].

Historically, AA has relied on three primary models:

- Generative Model: These statistical models aim to learn the underlying probability distribution of a dataset and generate new data samples that resemble the original dataset. Early AA models, such as tr-mmLDA, employed generative models to capture correlations between image features and annotation texts. Today, Generative Adversarial Networks (GANs) often represent generative models.
- Discriminative Model: Discriminative models model the dependence between unobserved variables using conditional probability distributions. Unlike generative models, they do not generate joint samples of variables X and Y. Discriminative models excel in tasks like classification and regression. In AA, each concept becomes a classification problem, with hierarchical classification used to bridge the semantic gap. Some methods combine generative and discriminative models to leverage their advantages.
- Graph Model: Graph-based image annotation methods, while computationally complex, have gained popularity in AA due to advancements in computational capabilities[18].

The field of AA is ever-evolving, with new techniques and approaches constantly emerging. While certain methods may be considered state-of-the-art at a given time, ongoing development means that newer methods may surpass them. Researchers and practitioners must stay up-to-date and assess the relative effectiveness of different methods for specific tasks.

Although termed "automatic," many AA approaches still require human involvement, particularly for annotating a small number of images. Recent advancements in End-to-End AA aim to eliminate human intervention from start to finish[9].

The state-of-the-art in image annotation continues to evolve with ongoing research and innovation:

Xiao et al.[13] introduced an "End-to-End Automatic Image Annotation Based on Deep CNN and Multi-Label Data Augmentation" approach. This method treats feature extraction and annotation as a multi-labeling problem, effectively reducing overfitting and enhancing model generalization using a multi-label Domain Adaptation (DA) based on ML-WGAN. While this method outperforms other image annotation approaches, it may require additional implementation work, as no associated frameworks are provided.

Wu et al.[34] explored a novel task called Diverse Image Annotation, which aims to annotate images with a limited number of tags that cover maximum semantic information. They introduced semantic metrics to evaluate the quality of tag lists, resulting in more human-consistent annotations compared to traditional metrics and achieving a state-of-the-art performance.

Subsequently, Wu et al. presented "D^2IA," which extended their previous work by incorporating a generative adversarial network and policy gradient algorithm to handle training challenges. This approach outperformed its predecessor in several metrics and offered more comprehensive image content descriptions[33].

The dynamic field of image annotation ensures that new developments will continue to shape its landscape, offering improved techniques and tools for various applications.

To facilitate the AA process, various semi-automatic annotation frameworks and commercial tools have been developed. These tools combine annotation techniques and machine learning algorithms to improve efficiency:

LOST (Label Object Suggestion Tool)[12] is a versatile semi-automated framework for annotating images and videos. LOST enables the creation of custom annotation pipelines, supports Python scripting, and can be configured as a cloud

application for distributed computation. It offers features such as point annotation, point supervision, and bounding box proposals, with a threshold of 0.5 for bounding box acceptance. LOST's flexible pipeline divides tasks into object localization and class label assignment, supporting multi-iterative annotation processes.

Bylabel[22] presents an innovative approach to automatic image annotation. It allows human selection of automatic boundary fragment proposals, eliminating the need for annotators to click on multiple boundary points. This framework incorporates edge detection and splitting algorithms to enhance annotation accuracy and efficiency.

Numerous commercial tools, including V7[29], Labelbox, and SuperAnnotate, provide advanced image annotation capabilities as described in 3.1. These tools often include features like model-assisted labeling, annotator statistics, and vector segmentations.

Most popular image annotation tools

	V7	Labelbox	Scale AI	SuperAnnotate	Playment	DataLoop	Supervise.ly	Hive Data
Model-assisted Labeling	✓	✓	✓	✓	✓	✓	✓	✗
Model training	✓	✗	✗	✓	✗	✗	✓	✗
Model inference	✓	✗	✗	✗	✗	✓	✓	✗
Annotator statistics	✓	✓	✓	✓	✓	✓	✓	✗
Vector Segmentations	✓	✓	✓	✓	✗	✓	✗	✗

Table 3.1: Annotation tool comparison provided by V7

3.2 Few Shot Object Detection

FSOD is a ML task that involves detecting objects using a limited number of labeled examples,making it possible for models to be trained with less data,therefore being important to our objective. To address the data limitation issue, FSOD employs techniques such as transfer learning and meta-learning (Fig 3.1), which enable the model to adapt quickly to new object classes.

There are methods like few-shot by finetuning that use a fully supervised dataset to train the model with a smaller set of examples or meta-learning techniques, where the model is trained on a variety of tasks with different classes and a different number of examples[23].

In FSOD, commonly, the training dataset is divided into a base dataset containing base categories and a novel dataset containing novel categories. A base dataset that contains categories from a pre-trained dataset and serves as the base for

the model capabilities to generalize and relevant feature extraction and a novel dataset that contains the categories/classes to be learned, this dataset forms the crucible for the model's detection capabilities.

If an object detector is trained using only the novel dataset, it is likely to overfit and have poor generalization due to the limited training data. However, if the combined dataset is heavily imbalanced, the detector will be biased towards the base categories[15].

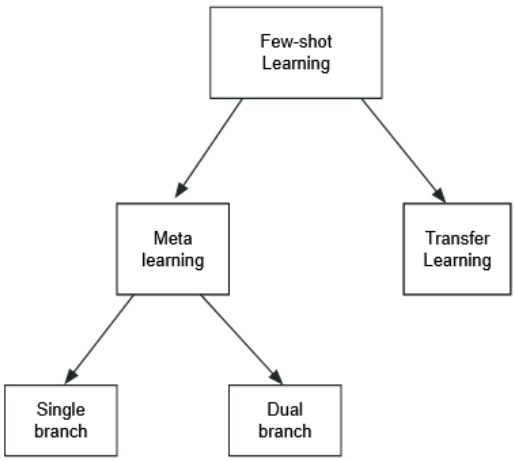


Figure 3.1: Few shot object detection taxonomy

The meta-learning literature has been producing a variety of interesting and performing approaches in recent years, thanks to breakthroughs in deep learning. However, there is still a lack of methods that can generalize knowledge on tasks with new, unseen data domains, hindering the application of deep learning solutions when the amount of available training data is low. Popular approaches also struggle when scaling to the complexity of the learner, resulting in poor performance on complex tasks. The heterogeneity of tasks can also cause counterproductive transfer learning, where irrelevant knowledge is reused. The objective is to determine whether current techniques can be extended to generalize knowledge on new tasks with unseen data domains. The desired model would be able to transfer knowledge among different data domains and tasks, potentially solving the lack of training samples in certain data domains[1]. Below are some of the best meta-learning approaches as of the present.

3.2.1 Meta-learning approaches

Meta-learning approaches are developed to enable a model to adapt rapidly to new tasks or environments by learning how to learn. This is especially useful in few-shot learning, where the model must learn to classify inputs into categories that may not have been fixed during training. In other words, the model must learn how to learn about the features and relationships that define each category, so that it can apply this knowledge to novel categories using only a few examples. To achieve this, meta-learning approaches often involve training the model on a diverse range of tasks and environments, allowing it to learn to generalize and adapt effectively to new situations.
Meta-learning for FSOD is split into single-branch and dual-branch approaches.

The **single-branch** approach is a type of architecture for object detection that has fewer learnable parameters, similar to generic detectors. Nevertheless, this approach will not be discussed further in this context.

The **dual-branch** approach, on the other hand, involves the use of a two-stream architecture consisting of a query branch and a support branch. In this approach, an input image is passed through the query branch, where the model detects object instances. The support branch receives a support set, which consists of support images with a single labeled object each. These objects can be presented in various ways: they can be cropped to the designated object using the ground-truth bounding box, the full image with a binary mask specifying the object location can be used, or the whole image can be used with the relevant features extracted using RoI Align.

The support branch is responsible for extracting relevant features from the support image and aggregating them with the query branch features, enabling the detector to find object instances from the support image in the query image[15].

Guangxing Han et al. propose a meta-learning based FSOD model, called *Meta Faster R-CNN*, which consists of a lightweight, coarse-grained prototype matching network for generating effective and efficient proposals for general few-shot classes, and a fine-grained prototype matching network with attentive feature alignment to address the spatial misalignment between the noisy proposals and few-shot classes[10]. This approach utilizes meta-learning for DA using the model's ability to learn how to learn and generate new examples that are similar to the ones in the limited training data. This model demonstrates superior performance

compared to fine-tuning when using very few examples (e.g. 1/2 shot), as fine-tuning is prone to overfitting with a small number of examples.

In their latest proposal, Guangxing Han et al. introduce a method called *Fully cross-transformer* that integrates a cross-transformer into the feature backbone and detection head. This approach employs asymmetric-batched cross-attention to combine K-V pairs from the query and support branch that have different batch sizes[11].

Both these approaches achieve State of the Art (SOTA) across multiple shots with the latter having a slight edge over the former.

3.2.2 Transfer Learning

Transfer learning is a technique in ML that involves using a model that has been trained on one dataset to improve the performance of a model on a different, related dataset. It is often used when there is a limited amount of data available for the new task and can help the model generalize better to the new domain[32]. In the context of few-shot learning and object detection, transfer learning involves using a model that has been trained on a larger, baseline dataset to recognize and classify novel object categories in the new domain. These novel categories may have more instances available than in few-shot learning, but may still not have as many as the baseline dataset. Therefore, techniques for learning with limited data may need to be incorporated in order to effectively use transfer learning for few-shot object detection[15].

Some modifications that may be applied to the transfer learning approach for FSOD include:

- Adapting the RPN weights during finetuning on the novel dataset and increasing the number of non-maximum suppression in order to reduce the number of missed detections;
- Adapting the feature pyramidal network weights during finetuning;
- Increasing the variance of training from novel categories via DA or pseudo-labeling to improve detection accuracy, especially for cases where the number of training examples is really low;
- Knowledge from the semantically most similar base category should be transferred in order to initialize the weights of components from each novel category;

- The angle between gradients of novel and base categories must be taken into account to prevent catastrophic forgetting and keep the performance on base categories;

- The loss should be modified regarding optimized gradient flow and inter-class separability. In an auxiliary branch, a contrastive loss can help to improve the discriminative power of features, like in two-branch meta-learning;

- On *Faster R-CNN based* detectors, a score refinement can help to reduce false positive classifications. Single-stage detectors can profit from an auxiliary branch in order to enable DA.

Ross Girshick proposed *Fast R-CNN* which is based on *R-CNN*, a CNN extension. The method takes as input the entire image and a set of object proposals, processes the image with multiple convolutional and max pooling in order to generate a convolutional feature map, and extracts a feature vector with a RoI pooling layer for each object proposal. The vector is then split into a softmax layer over K object classes and another that outputs four bounding box position values for each of the K object classes. For fine-tuning *Fast R-CNN* uses hierarchical sampling and a streamlined training process with a fine-tuning stage to train the softmax classifier and bounding-box regressors together instead of separately[6].

Shaoqing Ren et al. achieved state of art performance with *Faster R-CNN*, combining a deep fully convolutional network and *Fast R-CNN* detector into a unified network for object detection. *Faster R-CNN* improves the previous *Fast R-CNN* by making the region proposal step almost cost-free allowing for real-time runs. The learned RPN also improves region proposal quality and thus the overall object detection accuracy [23].

There is, however, an approach that tries to combine the strengths of both meta-learning and transfer learning.

Meta-Transfer Learning for FSOD combines the DNNs that are usually utilized in transfer learning and implements them into meta-learning that typically uses Shallow Neural Networks (SNNs). This approach proved to be highly efficient for adapting learning experiences to unseen tasks, especially for baselines and ablative models. The design is not dependent on any model thus having the advantage of being adapted for whatever task we choose to[28].

However, in our opinion, The other approaches have experimented on more challenging benchmarks meaning that this approach is still not comparable to them. Nevertheless, it is worthy of note due to its latent potential.

3.3 Summary

After the reflections done in the previous sections, we acquired major knowledge about the two key tasks necessary for the success of this project: Automatic Annotation and Few-shot Learning.

AA is based on 3 different models: generative, discriminative, and graph. These models approach the task of image annotation in different ways, but all aim to accurately assign labels or annotations to images without the need for human intervention. As a field in constant evolution, AA approaches and techniques are constantly being developed, that being said there are multiple tools that have already been developed and commercialized that can be used for the task.

FSOD is based on 2 different approaches: meta-learning and transfer learning. Since transfer learning is trained on DNNs and is likely to overfit with a low K-shot, dual-branch meta-learning is better designed for our main goal.

As for now, this indicates that tools like *V7*, *Superannotate*, or *Supervise.ly* are good candidates to be used as the automatic annotation framework and that *Meta Faster-RCNN* or *FCT* leading to the use of frameworks like *Facebook's Detectron2*[35] or *TensorFlow* Object Detection API for *Meta Faster-RCNN*. We did not find any tools that implement *FCT*.

However, we must acknowledge a fundamental challenge in the integration of AA with few-shot annotation tasks. AA fundamentally relies on pre-trained object detection models to perform annotations efficiently. In essence, without a robust detector capable of accurately localizing and identifying objects within images, the automatic annotation process encounters substantial difficulties,and thus,we believe that both cannot coexist.

Nonetheless, it remains imperative to investigate the practicality of combining automatic annotation with few-shot annotation tasks. This exploration is driven by the possibility of devising innovative strategies or methodologies to address this challenge effectively, even in the absence of an ideal detector. By undertaking this research, we aim to uncover potential solutions and approaches that can facilitate the coexistence of these two annotation by means of the frameworks and software mentioned above.

Chapter 4

Framework

In this chapter, we initiate a comparison between two distinct frameworks: the legacy version and the current version of the object detection system. The legacy version was initially engineered to recognize five specific classes: *Qalidus*, *Qaracter*, *EpiQ*, *Portugal*, and *Angola*. This prior iteration has demonstrated remarkable performance and serves as a solid foundation for our ongoing work.

However, the contemporary framework still relies on resource-intensive processes, such as manual annotation of extensive datasets. This practice not only introduces the potential for errors but also consumes significant time and effort.

Our objective is to leverage the strengths of the legacy version to facilitate an insightful comparison with the refined approach we are developing. Through this comparative analysis, we aim to address the challenges presented by the current framework, particularly those associated with data annotation and training with less data. This comparison enables us to evaluate the effectiveness and efficiency of our advancements in object detection.

The utilization of the legacy version as a benchmark allows us to quantify the improvements achieved in the current framework. This evaluation is pivotal in determining whether our modifications result in substantial enhancements in accuracy, efficiency, and overall performance. By undertaking this comparative analysis, our objective is to emphasize the merits of our novel approach, rooted in few-shot object detection. This method has been meticulously designed to alleviate resource-intensive processes and potential errors, all the while preserving or only slightly worsening the detection capabilities of the legacy version.

Through this evaluation, we aim to shed light on the effectiveness of our new approach, which harnesses the power of few-shot learning to enhance object de-

tection performance.

4.1 Existing Framework

The earlier version of this framework was built upon a model trained using the one-stage detector *YOLOv4*, leveraging a dataset comprising 1500 training images and 750 test images.

In contrast, the initial phase of the previous work concentrated solely on the Qalidus object, serving as the foundational investigation focus.

The latest iteration, which also serves as the foundation for this dissertation, introduces four additional classes. These classes are precisely defined based on annotated data, and their specifications are outlined in Table 4.1.

Table 4.1: Original Dataset Information

Class	**Training Instances**	**Training Images**	**Test Instances**	**Test Images**
Qalidus	2128	374	283	92
Qaracter	1527	377	251	90
EpiQ	1137	336	249	84
Angola	410	120	78	26
Portugal	239	89	58	26

Additionally, during this replication, we identified several annotation errors within the training dataset. As a result, 56 images were removed, resulting in a revised training dataset comprising 1454 images. The adjusted instance distribution is presented in Table 4.2.

Table 4.2: Dataset Adjustment

Class	**Training Instances**	**Training Images**	**Test Instances**	**Test Images**
Qalidus	1680	344	283	92
Qaracter	1401	329	251	90
EpiQ	1066	313	249	84
Angola	409	119	78	26
Portugal	238	88	58	26

4.2 Proposed Framework

Within this proposed framework, our primary objective revolves around the seamless creation of an automated onboarding platform. This pivotal advancement

ensures the swift integration of novel object classes into the detection process, enabling immediate recognition upon introduction.

To illustrate this capability, consider the scenario where a company like "*Twix*" seeks our expertise. Our focus lies in optimizing the utilization of "*Twix*" chocolate bar examples for shelf identification. We are dedicated to minimizing the requisite number of instances necessary for precise identification, thereby enhancing operational efficiency. Additionally, a core facet of our framework involves the development of an automatic onboarding procedure for our clients. By simply providing positive object examples, clients can initiate the process. Our sophisticated system takes charge of subsequent steps, encompassing both object identification and testing against a comprehensive dataset.

Through these initiatives, our prepared framework is poised to revolutionize the object detection landscape by delivering an unparalleled combination of efficiency, adaptability, and accuracy. The architecture for the system is presumed to comprise the following pipeline:

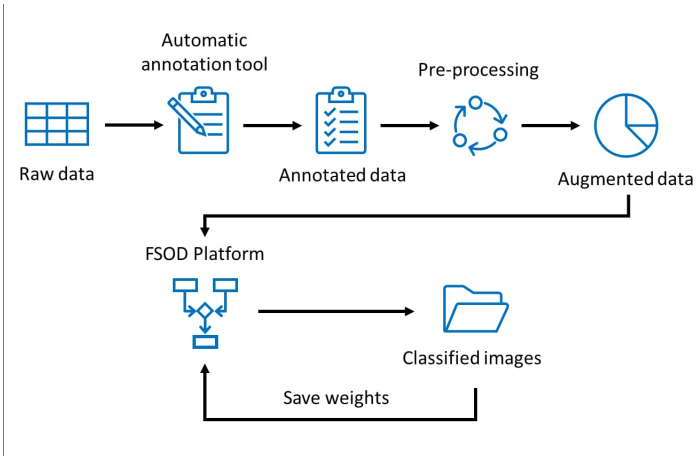


Figure 4.1: System Architecture

Our overarching strategy revolves around harnessing the potential of automated annotation tools to process and enhance the raw data through augmentation techniques. Following this data preparation phase, we will employ the cutting-edge *Detectron2 (d2)* platform, utilizing the *Meta Faster R-CNN* algorithm, to facilitate the prediction of product classes within images.

For the annotation tool, options such as V7 and Supervise.ly, both of which offer trial versions, present promising avenues for experimentation. Notably, *Meta Faster R-CNN*, as previously mentioned, stands out for its state-of-the-art performance in meta-learning few-shot object detection. Hence, it stands as a strong

contender for achieving optimal results among the various models considered.

Chapter 5

Experimentation

In this chapter, we embark on a comprehensive journey through the experimental analysis of our proposed framework for few-shot object detection. This chapter encapsulates the essence of our research, as it provides a deep dive into the empirical evaluation of the framework's capabilities. We begin by elucidating the dataset used for assessment, shedding light on its composition, intricacies, and significance in evaluating the framework's performance.

The chapter unfolds with the replication of the base model, a crucial step that establishes a benchmark for comparison. By doing so, we ensure a consistent starting point and verify the seamless integration of the legacy model into the novel framework.

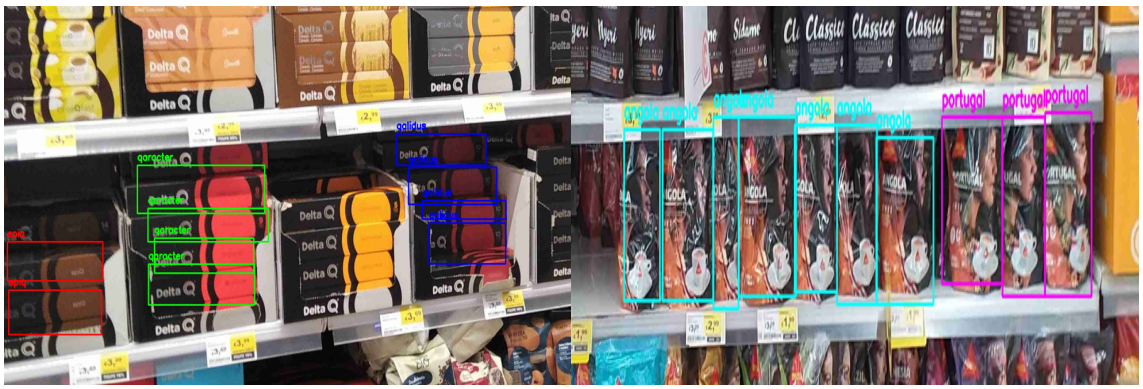
Central to this chapter is the execution of the FSOD experimentation itself. We delve into the *Faster-RCNN* (FRCNN) algorithm as the previously indicated *Meta-Faster RCNN* was not found in the given configurations for FSOD in the d2 repository despite their documentation affirming its existence.

The core of this chapter revolves around unveiling the outcomes yielded by our experimentation efforts. Within this section, we unveil the performance results of our framework when subjected on novel data. To dissect its efficacy, we rely on precision-recall curves, Mean Average Precision (mAP) scores and other pertinent metrics. mAP is a metric that comes from calculating the Average precision across multiple classes which assesses the performance of a model for a single object class and considers various confidence thresholds. For our particular experiments, we consider the threshold of 0.5 Intersection over Union (IoU) which is usually the common criterion to whether a detection is considered correct or not.

This chapter is then organized into several sections, commencing with a description of the dataset employed. Following with the experimental setup and the results. The chapter culminates with a comprehensive discussion of these results, from which we draw meaningful conclusions.

5.1 Dataset

Before starting our actual practical recreation it is important to present the dataset in its full detail. The dataset is composed of 1454 training images and 750 test images. We will shortly present a description of how each class is supposed to look in order to better distinguish them in further shown experiences.



(a) Qalidus, Qaracter, and Epiq (b) Angola and Portugal

Figure 5.1: Visual examples of the classes

The distinguishability of various classes in figure 5.1 is evident; for instance, the *EpiQ* class stands out due to its concise packaging format and distinctive brown color. In contrast, the *Qalidus* and *Qaracter* classes share a similar color palette, differing primarily in brightness. Notably, the *Portugal* and *Angola* classes are easily discernible, owing to their distinct letter combinations and unique packaging imagery.

However, these numerical values are not suited for few-shot object detection, nor do they lend themselves to an optimal experience in the realm of Few-Shot Object Detection (FSOD). A few-shot dataset typically encompasses three distinct dataset types.

The **base dataset** that serves as the bedrock, encompassing the foundational base classes. It consists of an extensive collection of data, brimming with valuable information while remaining independent of any insights related to the novel

classes. The primary objective of this dataset is to facilitate the detector in extracting pertinent features that are relevant to the novel classes from the pre-existing data features. An essential constraint is that this dataset must not contain any unannotated objects belonging to the novel set. Notably, our base dataset is derived from the widely acknowledged COCO dataset, which stands as a benchmark in the realm of object detection, containing 90 classes from which 60 are rotated as base depending on the metadata definition.

The **novel dataset** that comprises a limited number of instances from each class we want to detect, this dataset forms the crucible for the model's detection capabilities. Ideally, the dataset's size should align with the product of the shot count and the number of classes, such that for instance, with a shot count of 30 and 5 classes, the dataset would ideally encompass 150 images. Within each image, 30 instances of each class are annotated. Our novel dataset is meticulously curated from our *Delta* dataset images, arranged into batches that adhere to the aforementioned criteria. It is important to note that one particular scenario diverges from the norm, where 78 images were specifically chosen to accommodate 30 instances for each class. This value represents the maximum number of images attainable for constructing a novel dataset from the *Delta* dataset. This term accounts for the exhaustion of images with only one object annotated first, followed by those with two, three, and so forth, until reaching the dataset's requisite number of shots.

Following the completion of training with the novel dataset, the **mixed dataset** incorporates objects from both novel and base classes. This dataset plays a pivotal role in assimilating the acquired knowledge from combining the previously trained models.

With a full insight about the dataset we are now ready to explore the totality of the experiences realized. Further sections will delve into these experiences with more detail.

5.2 Base Framework Model Recreation

In this section, a replication of the base model was made as a comparison benchmark for future experiences. In order for the interested reader to be able to recreate the experiences there are some critical steps to take in preparing the framework:

For the purpose of enabling interested readers to replicate the conducted experiments, it is imperative to meticulously follow a series of critical steps to properly

configure YOLOv8:

- **Create and Organize Directories:** Begin by establishing a directory exclusively named *yolov8* to house the entire program. Within this directory, create a subfolder for the datasets. Organize this folder structure to encompass separate *test* and *train* folders. These folders, in turn, should each contain two subfolders—*labels* and *images*—for the storage of annotations and images, respectively.

- **Install Ultralytics:** Utilize the command `pip install ultralytics` to install the *Ultralytics* library, which will be employed in the forthcoming steps.

- **Create YAML Configuration:** Craft a YAML configuration file where crucial parameters are defined. Specify the dataset folder path along with the respective paths to the *test* and *train* folders. Additionally, enumerate the classes to be employed, assigning numerical labels (ranging from 0 to $n - 1$, where n is the number of classes) and their corresponding names. Notably, adjustments such as augmentation, learning rate, and supplementary settings can be tailored within this YAML configuration or via command line.

- **Training:** Initiate the training process by executing the command:

  ```
  yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml
  epochs=25 imgsz=800 plots=True
  ```

  Here, *yolov8n.pt* represents the pre-trained model utilized for knowledge transfer into the model undergoing training.

- **Validation:** To assess the model's performance, execute:

  ```
  yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt
  data={dataset.location}/data.yaml
  ```

  Similar to the training process, this step relies on the *best weights* obtained during the training phase.

- **Predictions:** For generating predictions, execute:

  ```
  yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt
  conf=0.25 source={dataset.location}/test/images save=True
  ```

  In this context, the confidence threshold for predictions can be defined, ensuring that only predictions surpassing this threshold are made.

5.3 Few-shot Object Detection Experimentation

This section begins with a brief explanation about our setup,with a recreation of the environment followed by a description of the actual parameters used for each experience

5.3.1 Experimental Setup

To establish the credibility and robustness of the data utilized for few-shot object detection (FSOD) within the Detectron2 (d2) framework, an array of diverse few-shot datasets was curated, using 1, 2 ,5 , 10 and 30 shots which were generated via a randomized selection of images until the number of instances was achieved . This approach aimed to mitigate any potential misattributions of the model's performance solely to biases originating from annotation quality. With this goal in mind, a set of compact datasets was generated to cater to different few-shot scenarios. Specifically, we generated ten unique instances of larger shot sizes (10 and 30) using seed values ranging from 1 to 10 and were aiming to generate more samples for each existant shot. However, due to time constraints, the seed generation process was not exhaustive. It is important to acknowledge that a more comprehensive exploration of seed values with at least 30 data points per group,executing multiple seeds in our case, would have been ideal for recreating the experience. Thus, seeds were not a step to be considered in our experimentation. Instead, we will evaluate the results of our experiences based on the shot and training dataset used.

As for the remaining parameters for the experiences,they were trained throughout 50000 total iterations with the learning rate fixed at 0.01 and images subjected to the Random Flip and Resize Shortest Edge techniques.

Table 5.1: Training Parameters

Parameter	Value
Epochs	50000
Learning rate	0.01
Image Size	640
Data augmentations	Random Flip and Resize Shortest Edge

Using Detectron2 Faster-RCNN proved to be more intricate to implement. This framework revolves around COCO and other benchmark datasets like PASCAL

VOC and LVIS, requiring custom coding for a specialized dataset environment. Follow the steps below to set it up:

1. **Environment Setup:**
   - Clone the repository: `https://github.com/ucbdrive/few-shot-object-detection`.
   - Build Detectron2: Clone and build `https://github.com/facebookresearch/detectron2` to establish the necessary engine.
2. **Dataset Configuration:**
   - You can skip cloning the first dataset repository as the changed code is already provided to the reader.
   - Modifications in the code involve handling novel classes, metadata registration, evaluation processes, and the few-shot data generator.
   - Adjust the data generator:
     - Annotations are aligned with the COCO dataset categories list.
     - COCO's list contains 90 classes, with 60 acting as the base dataset depending on metadata.
     - This forces the current class IDs to be offset by 90, so if original IDs were 1 to 5, new IDs become 91 to 95.
   - Prepare the dataset:

     ```
     python datasets/prepare_coco_fews_shot.py [0-10]
     ```

     The dataset naming convention plays a critical role in proper metadata registration, facilitating seamless integration into the framework. A stringent format is required for dataset naming to ensure accurate metadata handling and effective utilization. Specifically, the dataset must be named following the structure: `foldername_trainval_[novel,base,all]_[1,2,5,10,30]shot[_seed]`, where `[novel,base,all]` represents the type of dataset (novel, base, or combined), and `[1,2,5,10,30]shot` signifies the number of shots used in the dataset. The `[_seed]` component is optional and should only be included if a seed was used during data creation.
3. **Configuration Files:**
   - Edit YAML files (`configs/COCO-detection/faster_rcnn_R_101_FPN_` $\times$ `.yaml`):

- Modify datasets, epochs, and batch size; refrain from altering other settings already optimized for few-shot.
- Changing learning rates in configuration files yielded significant result degradation.

4. **Two-Stage Fine-Tuning (TFA):**

   - **Stage 1 - Base Model Training:** Train the base model:

     ```
     python3 -m tools.train_net --num-gpus 1 \
         --config-file configs/COCO-detection/
         faster_rcnn_R_101_FPN_base.yaml
     ```

   - **Stage 2 - Fine-Tuning:**
     - Remove last layer weights:

       ```
       python3 -m tools.ckpt_surgery \
           --src1 checkpoints/coco/faster_rcnn/
           faster_rcnn_R_101_FPN_base/model_final.pth \
           --method remove \
           --save-dir checkpoints/coco/
           faster_rcnn/faster_rcnn_R_101_FPN_all
       ```

     - Fine-tune predictor on novel set:

       ```
       python3 -m tools.train_net --num-gpus 1 \
           --config-file configs/COCO-detection/
           faster_rcnn_R_101_FPN_ft_novel_1shot.yaml \
           --opts MODEL.WEIGHTS checkpoints/coco/faster_rcnn/
           faster_rcnn_R_101_FPN_all/model_reset_remove.pth
       ```

     - Combine base and novel weights:

       ```
       python3 -m tools.ckpt_surgery \
           --src1 checkpoints/coco/faster_rcnn/
           faster_rcnn_R_101_FPN_base/model_final.pth \
           --src2 checkpoints/coco/faster_rcnn/
           faster_rcnn_R_101_FPN_ft_novel_1shot/model_final.pth
           --method combine \
           --save-dir checkpoints/coco/faster_rcnn/
           faster_rcnn_R_101_FPN_all \
           --coco
       ```

5. **Final Fine-Tuning and Evaluation:**

   - Fine-tune last layer on balanced dataset:

```
python3 -m tools.train_net --num-gpus 1 \
    --config-file configs/COCO-detection/
    faster_rcnn_R_101_FPN_ft_all_1shot.yaml \
    --opts MODEL.WEIGHTS $WEIGHTS_PATH
```

   - For evaluation:

```
python3 -m tools.test_net --num-gpus 1 \
--config-file configs/COCO-detection/
faster_rcnn_R_101_FPN_ft_all1_1shot.yaml \
    --eval-only
```

It is noteworthy that options like `-eval-only`, `-eval-all`, and `-eval-during-train` are provided for evaluation. An evaluation is performed even after the training phase. This command is useful to avoid retraining the entire model, facilitating comparisons. Moreover, the upgraded framework not only enables training and testing but also enhances the analysis process. It automatically generates a dedicated folder within the checkpoint directory containing predictions and annotations drawn over test images. This augmentation provides a more comprehensive basis for comparison and evaluation.

5.3.2 Experimental Results

To establish comprehensive benchmarks and enable fair comparisons, we conducted separate experiments using both the YOLOv8 and Faster R-CNN architectures. The YOLOv8 experiment served as a benchmark within the YOLO framework, providing a baseline for comparison. Similarly, the Faster R-CNN experiment allowed us to evaluate the standalone performance of the Faster R-CNN model. This approach ensures a rigorous comparison by assessing each framework independently in its unique experimental context.

For an in-depth examination of the experimental results, we offer comprehensive visualizations and tabulated data. To ensure accessibility and mitigate potential visibility issues caused by document size limitations, we have thoughtfully

provided a cloud folder containing images depicting predicted classes and corresponding ground truth annotations for each experiment[1].

5.3.2.1 YOLO-v8 Experiments

In this subsection, we unveil the results obtained from our experimental endeavors, commencing with the non-few shot YOLOv8, and subsequently, the few-shot YOLOv8. Both experiments were trained using the parameters listed in Table 5.2.

Table 5.2: Training Parameters

Parameter	Value
Epochs	150
Learning rate	0.1
Image resize	800
Data augmentations	Mosaic and Mixup
Confidence interval	25

It is imperative to note that within the few-shot context, the YOLO framework encountered limitations, specifically, it could not operate effectively with shot counts below 5. Consequently, results for 1 and 2-shot scenarios are omitted due to their infeasibility, as the detector consistently crashed without yielding any results.

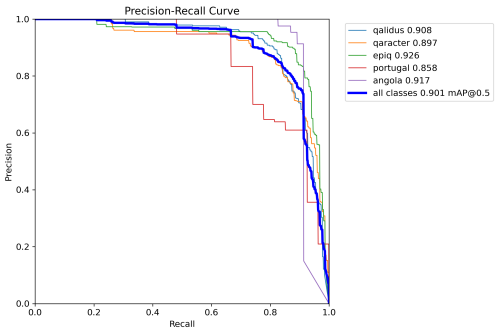


Figure 5.2: Precision Recall for Non-few shot YOLOv8

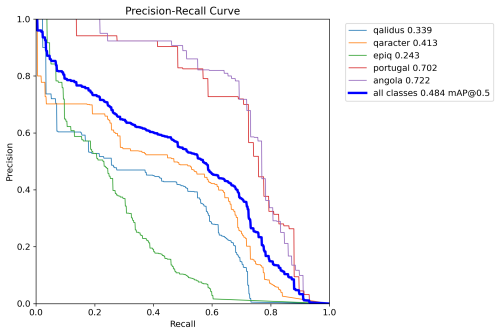


Figure 5.3: Precision Recall for 30-shot YOLOv8

[1]**https://drive.google.com/drive/folders/1LXT4Bc9B61YP0V9N5hvF238cqcO2a-pm?usp=sharing**

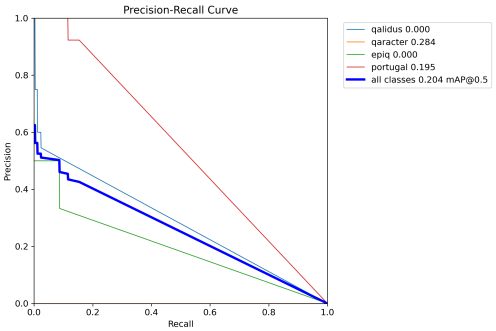


Figure 5.4: Precision Recall for 10-shot YOLOv8

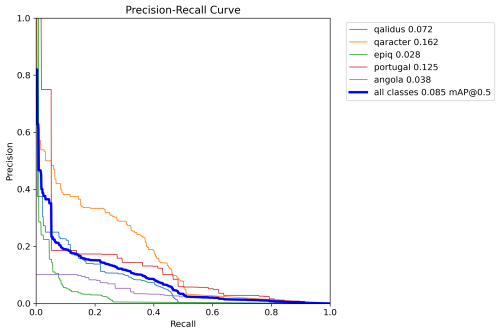


Figure 5.5: Precision Recall for 5-shot YOLOv8

Table 5.3: Performance according to the mAP metric of each and all classes involving training for standard "full" and FSOD in YOLOv8

All classes mAP	*Qalidus*	*Qaracter*	*EpiQ*	*Portugal*	*Angola*	Train	Test	Shots
0.901	0.908	0.897	0.926	0.858	0.917	1454	750	Full
0.484	**0.339**	**0.413**	**0.243**	**0.702**	**0.722**	78	750	30
0.204	0.000	0.284	0.000	0.195	0.541	38	750	10
0.085	0.072	0.162	0.028	0.125	0.038	21	750	5

Figure 5.6: Predictions on standard YOLO

Figure 5.7: Predictions on 30-shot YOLO

We can verify that in figure 5.7 detects some objects incorrectly as being from the *Angola* and *Portugal* classes and the standard model in figure 5.6 detects everything correctly with no errors.

In these experiments, we present mAP results for both the training and validation phases across three distinct shot scenarios: 5-shot, 10-shot, and 30-shot.

Notably, these experiments produced results that exhibited more pronounced disparities in metrics compared to our previous trials. As a consequence, each experiment warrants an individual examination.

In the case of the 30-shot scenario, a noteworthy observation is there was relative consistency in metrics between the training and validation phases. This indicates that the model's performance doesn't degrade significantly when transitioning from training to validation.

Furthermore, it's evident that the model excels in distinguishing objects such as Portugal and *Angola*. These specific product classes are characterized by a significant number of images, each containing only one instance of the object. This finding underscores the pivotal role of dataset quality in few-shot object detection. The presence of more images with isolated objects allows the model to gain a deeper understanding of these classes. Notably, both *Portugal* and *Angola* significantly contribute to the overall *map* by achieving scores that are approximately twice as high as those of the other categories. It's important to note that these

objects are larger in size compared to their counterparts, which likely contributes to their enhanced detection performance.

When we examine lower shot values, specifically 10 and 5, we observe a consistent trend where the model tends to reduce the mAP, a behavior well-documented in transfer learning models.

Additionally, it is crucial to emphasize that the validation results exhibit notable disparities when compared to what were our training results. This discrepancy serves as a strong indicator of overfitting and/or poor model complexity.

5.3.2.2 Faster RCNN Experiments

Moving on to the few-shot Faster R-CNN experiments, we note that the framework evaluates the mAP in the range of 0 to 100. To facilitate meaningful comparisons, we have normalized these values to a 0 to 1 scale.

Table 5.4: Performance according to the normalized mAP metric of each and all classes involving training for standard "full" and FSOD in FRCNN

All classes mAP	*Qalidus*	*Qaracter*	*EpiQ*	*Portugal*	*Angola*	Train	Test	Shots
0.58688	0.51319	0.31990	0.33197	0.15668	0.25191	1454	750	Full
0.42403	**0.46545**	**0.31365**	0.00006	0.14547	**0.22447**	78	750	30(Hp)
0.38089	0.42063	0.27143	0.00006	0.14497	0.21060	78	750	30
0.38089	0.42346	0.26046	**0.00050**	**0.15074**	0.20619	38	750	10
0.38670	0.42900	0.27082	0.00036	0.14971	0.20487	21	750	5
0.37761	0.43045	0.26457	0.000	0.14743	0.19989	10	750	2
0.24941	0.35310	0.22029	0.000	0.14574	0.17989	5	750	1

These results reflect the meticulous evaluation of each framework's performance, enabling meaningful insights into their capabilities. Below is a visual example of the best model compared with the standard experience.

Figure 5.8: Predictions on standard FRCNN

Figure 5.9: Predictions on Handpicked 30-shot FRCNN

We can examine a very similar behavior in both images, with both of them detecting the objects correctly, meaning that the knowledge was indeed transferred from one model to another. Both also struggle to detect the *EpiQ* instances. The standard model also seems to identify a background object as *Qalidus* with low confidence.

Taken together, these results accentuate the significant influence of the training data provided to the model. Notably, the considerable increase in mAP in the handpicked scenario, approximately 3.5 points higher than the randomly selected data version with the same shot count, implies that the quality of the data plays a pivotal role. The poor performance of the 1-shot scenario can likely be attributed to the quality of the images used. If an image lacks essential conditions such as proper lighting, object positioning, size, and so on, the model faces substantial challenges in learning from a single image effectively.

This underscores the potential for human error in contributing to the suboptimal performance of a given model. It suggests that even the most advanced models may struggle to perform well when confronted with subpar data conditions. However, the hypothesis regarding random data's impact can only be

definitively validated with a more extensive set of experiments for each model. To thoroughly evaluate the models' capabilities, including the relevance of hand-picked data and seed variation, we propose conducting each experiment a minimum of 30 times. Such a comprehensive approach would allow for in-depth Exploratory Data Analysis (EDA) to draw robust conclusions.

Looking at our best model in the FRCNN experiences in 5.4, the 30-shot hand-picked, if we compare it to the actual standard experience we can affirm that there isn't a very significant drop in mAP in terms of what it is to be expected from training a model with a lot less examples. However, if we look at the 5.2 we can notice a very drastic drop in the metric. However, there is one odd behavior in the detection task. When scrutinizing the mAP scores for each class, it becomes evident that the class *EpiQ* displays conspicuously low results, to the extent that it can be considered virtually undetected. Several factors could contribute to this anomaly.

Firstly, it's plausible that certain color-related features associated with the *EpiQ* class may have introduced interference, making it more challenging for the model to correctly identify objects within this category. The choice of images used for training might have inadvertently included problematic instances that hindered the model's ability to generalize effectively. Another potential explanation could be linked to the base model with the possibility that it itself lacks specific features necessary for distinguishing the *EpiQ* class. Reflecting on these results raises the question of whether the limitations in FRCNN constrain the learning process. This concern prompted our decision to conduct further experiments with the YOLOv8 framework. Additionally, we trained a standard Faster R-CNN model to expand the scope of our comparative analysis.

5.3.3 Discussion

The presented experimental results, in conjunction with the images, showcased earlier, yield valuable insights into the performance of the YOLOv8 and FSOD frameworks within the context of few-shot object detection. This discussion aims to delve deeper into the obtained results and their implications.

Table 5.5: Performance of the final experiences according to the mAP metric of each and all classes involved in training for standard "full" and FSOD N-shot training.

Model	All classes	*Qalidus*	*Qaracter*	*EpiQ*	*Portugal*	*Angola*	Training type
YOLOV8	0.901	0.908	0.897	0.926	0.858	0.917	Full
Detectron2	0.587	0.513	0.320	0.332	0.157	0.252	Full
YOLOV8	0.484	0.339	0.413	0.243	0.702	0.722	30-shot (Hp)
Detectron2	0.424	0.465	0.314	0.001	0.145	0.225	30-shot (Hp)

We begin by examining the results of the FRCNN framework, which constituted our primary focus for the FSOD task. Comparing the standard experience results in Table 5.5 a significant drop in mAP is evident. This decline provides compelling evidence that YOLOv8 outperforms FRCNN in the context of overall object detection. This observation corroborates the findings of previous research, where YOLOv4 demonstrated superior performance over all two-stage detectors.

Further analysis of the various shot scenarios in Table 5.4 reveals several noteworthy insights:

- **Consistency in mAP from 30 to 2 Shots:** The remarkable consistency in mAP values from 30 to 2 shots can be attributed to the independence of each experiment with its respective dataset. In these experiments, the model commences training anew for each shot scenario. The proximity of results may suggest that the underlying base model exhibits a capacity for effective generalization.

- **Drop in mAP at 1 Shot:** The sharp decline in mAP at the 1-shot scenario highlights a critical threshold concerning the availability of training data. The model grapples with severe data scarcity with only a single instance of each class available for training. Consequently, the model's capacity to generalize and acquire discriminative features for each class is significantly impeded, leading to a pronounced reduction in detection accuracy. This underscores the formidable challenges associated with one-shot learning scenarios and contradicts the previously proposed idea that the results stem solely from the base model's capabilities.

- **Increase in Handpicked Images:** The observed rise in mAP for the handpicked (30 shots) scenario underscores the model's proficiency in handling well-annotated and diverse training data. When furnished with a larger

dataset containing 30 instances of each class, thoughtfully selected based on FSOD criteria, the model gains more opportunities to assimilate intricate details and finer-grained features associated with each object class. Consequently, the model attains a heightened level of accuracy in object detection.

When examining the overall results of our FSOD experiments with our models, several conclusions can be drawn regarding the performance of FSOD on our dataset and areas for potential improvement.

Both *YOLOv8* and FRCNN exhibit similar performance, with *YOLOv8* outperforming FRCNN by approximately 6 points in scenarios with a higher number of training examples, where transfer learning is advantageous. However, in cases with a lower number of training examples, Faster R-CNN significantly outperforms *YOLOv8*. In fact, *YOLOv8* struggles to learn certain classes in these low-shot scenarios, with the exception of the *epiQ* class, which underperforms in both models.

A notable observation is the substantial performance gap between the standard training experiences and few-shot experiences in *YOLOv8*, while FRCNN exhibits a smaller performance discrepancy between these scenarios.

One significant challenge we encountered is the presence of products displayed together on store shelves. This situation contradicts several principles for creating an ideal dataset, as classes frequently co-occur in images, resulting in fewer instances where a single class appears in isolation.

To improve our results, we could enhance the dataset by ensuring a higher proportion of instances contain only one class per image. Additionally, conducting more experiments with more random seeds may yield a more robust conclusion.

However, it's worth noting that the field of FSOD is still evolving, and it currently lacks the capabilities to effectively distinguish objects within shelf images due to the substantial amount of information present. This limitation renders the automatic onboarding framework ineffective in such scenarios.

Chapter 6

Conclusions and Future Work

Brands&Ninjas (B&N), a corporate entity, has enlisted the services of *Redlight Software* with the aim of developing a software solution to address specific challenges. The proposed solution is conceived as a crowdsourcing platform strategically designed to assist brands in inventory management, price monitoring, and shelf scanning, with a particular emphasis on adhering to restocking schedules.

This platform leverages a dedicated mobile application involving a user base referred to as "*ninjas*" who are entrusted with the execution of various missions related to these objectives. It's crucial to acknowledge that the data generated during these missions is subject to manual validation, considering the potential for inaccuracies in the responses provided by the "*ninjas*."

The overarching goal is to engineer an automated system capable of autonomously validating all images captured by the "*ninjas*."

In this document, our objective was to create an automatic training and annotation system for OD. To achieve this, we introduced key definitions necessary for a comprehensive understanding of important concepts such as AA and FSOD. Additionally, we presented a selection of tools and models that might be suitable for our problem. We also provided context about the existing framework and the approach to pinpointing the root of the problem.

However, as we highlighted in the SOTA, AA is a rapidly evolving field with techniques continually under development. Yet, it requires a pretrained model to successfully annotate the desired objects. On the other hand, FSOD aims to train a model to detect classes with limited examples. This implies that these two concepts cannot coexist since we cannot annotate an object we have not yet learned to detect.

This dissertation subsequently shifted its focus entirely towards successfully training a model for FSOD. We replicated previous work experiences using a full training set in *YOLOv8* and also used a standard FRCNN as benchmarks for comparison with few-shot approaches. Subsequently, we compared our experiences with these benchmarks and among themselves, concluding that both *YOLO* and FRCNN were not suitable for the FSOD task. This can be attributed to factors such as the dataset quality, which was not initially prepared for few-shot learning but instead designed for standard object detection. Other factors include the base model itself and the absence of replicable experiences. If we were to chose, when considering the practicality and suitability of these models for FSOD, the preferred choice would be the YOLOv8 30-shot approach. This model closely aligns with the requirements of an onboarding framework. The obtained results are reasonable and indicate room for improvement, potentially pushing the mAP beyond the 50-point threshold. Additionally, YOLOv8's training process proves significantly faster than its competitor, with an average training time of approximately half an hour, whereas d2 requires an average of 19 hours for training. We don't recommend the use of any of the models above 30-shot in spite of being unpredictable,having a close mAP value but not reflecting the results that well visually and always underperforming on the *EpiQ* class.

In terms of future work, several actions could potentially enhance the results of this experiment. One approach is to consider replacing the existing training dataset with individual, unshelved images for each class object targeted for detection. This may help mitigate potential biases introduced by the dataset's initial design. Additionally, conducting experiments using different base datasets could reduce dependence on the current COCO dataset, providing a broader perspective on model performance. Moreover, increasing the number of samples for each experiment would allow for the formulation of stronger hypotheses, more robust conclusions, and potentially a more streamlined path to finding a solution, if feasible.

References

[1] N. D. Angeli. State of the art on: Meta-learning for few-shot classification. In *State of the Art on: Meta-learning for Few-Shot Classification*, 2020.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, jun 2002.

[3] Y. Chen, X. Zeng, X. Chen, and W. Guo. A survey on automatic image annotation. *Applied Intelligence*, 50, 10 2020.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[6] R. Girshick. Fast r-cnn, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. End-to-end learning of deep visual representations for image retrieval, 2016.

[10] G. Han, S. Huang, J. Ma, Y. He, and S.-F. Chang. Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):780–789, Jun. 2022.

[11] G. Han, J. Ma, S. Huang, L. Chen, and S.-F. Chang. Few-shot object detection with fully cross-transformer, 2022.

[12] J. Jäger, G. Reus, J. Denzler, V. Wolff, and K. Fricke-Neuderth. Lost: A flexible framework for semi-automatic image annotation, 2019.

[13] X. Ke, J. Zou, and Y. Niu. End-to-end automatic image annotation based on deep cnn and multi-label data augmentation. *IEEE Transactions on Multimedia*, 21(8):2093–2106, 2019.

[14] A. Kumar. Different types of cnn architectures explained: Examples https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/, Apr 2022.

[15] M. Köhler, M. Eisenbach, and H.-M. Gross. Few-shot object detection: A comprehensive survey, 2021.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, and Thrun, Sebastian. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

[18] J. Liu, B. Wang, M. Li, Z. Li, W.-Y. Ma, H. Lu, and S. Ma. Dual cross-media relevance model for image annotation. In *Dual cross-media relevance model for image annotation*, pages 605–614, 01 2007.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[20] D. G. Lowe. Sift: The scale invariant feature transform. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[21] N. J. Nilsson. *MLBOOK - Introduction to machine learning: An early draft of a proposed textbook*. Stanford University, 1996.

[22] X. Qin, S. He, Z. Zhang, M. Dehghan, and M. Jagersand. Bylabel: A boundary based semi-automatic image annotation tool. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1804–1813, 2018.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[24] L. Roberts. *Machine Perception of Three-Dimensional Solids*. 01 1963.

[25] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[26] O. G. Selfridge. *Pandemonium: A Paradigm for Learning*, page 115–122. MIT Press, Cambridge, MA, USA, 1958.

[27] C. Shorten and T. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019.

[28] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[29] v7. 13 best image annotation tools of 2023 https://www.v7labs.com.

[30] v7. The essential guide to data augmentation in deep learning https://www.v7labs.com.

[31] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[32] K. Weiss, T. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3, 05 2016.

[33] B. Wu, W. Chen, P. Sun, W. Liu, B. Ghanem, and S. Lyu. Tagging like humans: Diverse and distinct image annotation, 2018.

[34] B. Wu, F. Jia, W. Liu, and B. Ghanem. Diverse image annotation. In *Diverse Image Annotation*, 07 2017.

[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[36] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen. Image data augmentation for deep learning: A survey, 2022.

[37] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *arxiv*, 2019.