Universiteit Leiden

Master Computer Science

Brand Recognition in the Textile Waste Stream

Name:	Aaron Dunlea
Student ID:	s2687763
Date:	05/07/2024
Specialisation:	Data Science
1st supervisor:	Daan Pelt
2nd supervisor:	Yves Sohege

Master's Thesis in Computer Science
Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Universiteit Leiden

Contents

Acknowledgements

I would like to take this opportunity to give my thanks to a number of parties that helped me undertake this project. First and foremost, my two supervisors Daan and Yves, who were amazing mentors and guiding voices throughout the entire process. Next, I would like to give my sincere thanks to Erdotex Group for allowing me to use their facilities and being gracious enough to release the gathered dataset publically. Finally, I would like to thank my family, partner and friends for their unwavering support and encouragement. Their belief in me was a constant source of strength and motivation.

Abstract

In this project, we explore the application of various methods of Logo and Brand Detection to the textile waste industry. In light of upcoming Extended Producer Responsibility (ERP) policies due to come into effect in 2025, making brands and producers responsible for the waste they create. Producers must provide systems to process their products for recycling after the end-of-life of their products. We apply several methods to explore the feasibility of adhering to such a policy in a real-world industry setting. This included gathering an image dataset for evaluation and training purposes on-site in a large textile sorting facility. Three experiments were conducted to investigate the best-performing technology when applied to this setting. The first experiment investigates the use of readily available models on 4 evaluation datasets. The second experiment explores the effect of fine tuning a YoloV5 object detection model with an industry dataset and large logo dataset. The third experiment compares the performance of available LLM's on the evaluation datasets. Depending on the context of the implementation, the resulting preferred method falls into two categories: (1) If only a small amount of brands are needed to be extracted then the best choice for this is to follow standard machine learning practices with a custom gathered dataset to train a object detection model for the specific brands in question. (2) In a general setting, where all visible brands need to be considered, then it is not feasible to gather a dataset for each possible brand that may show up in the waste stream. In this case, the use of vision capable Large Language Models (LLM's) achieve acceptable results due to their large corpus of training data.

Introduction

The textile industry is one of the oldest industries in existence, with records of textile sales dating back to before times of ancient Egypt and Mesopotamia. In the modern textile industry, the supply chain of a textile product is mature and spans many sub-industries across the world. While the textile industry as a whole has been around for thousands of years, the concept of textile sorting is relatively new, with the first occurrences popping up in the UK Industrial Revolution in the early 1900s, originally starting with human workers manually sorting garments based on garment type, colour, material makeup, and quality. Many of these categories still stand today, and the human worker is a crucial element in the process across many instances.

While at its core, much of the work of the textile sorting industry still relies on human hands to carry out the sorting of materials, there have been several technologies and innovations introduced over the last few decades to improve efficiency and sorting accuracy. These largely fall into three categories: Composition scanning through spectroscopy, visual sorting techniques using technologies such as computer vision and object recognition, and robotics, which can come in many degrees of verbosity; from fully autonomous systems that require little to no human interaction down to systems that aid human worker in transporting and maneuvering items of clothing across a facility.

Modern-day recycling techniques lean heavily into sustainable and environmentally friendly practices. These practices require clever ingenuity, heavy investment, and the reluctant adoption of new technologies over older legacy systems. The textile recycling industry is no different, around 53 million tonnes of textiles are produced every year, and about 73% of this ends up in landfills or is incinerated [22]. Of the remaining 27% of garments that have various destinations such as feedstock for other industries or cascading recycling systems, less than 1% of textiles remain in a closed loop recycling circle [22], [15]. With the modern trend of "fast fashion," this number is likely to increase if it goes unchecked. Many companies and municipalities are forced to invent new ways to "Greenify" the textile industry in a bid to tackle this problem. Typical waste recycling systems across the globe rely on several different independent processes working together to produce a product and then manage the waste that comes from it for the entire lifespan of the item. While it is easy to look at landfills as the end destination of the problem, it is every individual step of this production and waste stream we must look at to solve it.

The lifecycle of a garment can take many different forms depending on several factors such as, for example, the material composition, value, or location in which it is used. However, this process generally falls into the following workflow. At the start of the loop, we have producers and brands. These entities are responsible for creating the products and items that we use every day. Next up are the consumers.

The waste stream varies greatly as it depends on many different services working in parallel to handle and rework the waste so that it can once again gain value. One example of this process can be seen in Figure 4. Typically, when looking at the chain from a recycling standpoint, the first step in the waste stream is collection. In the textile industry, collection can come in some of the following forms; Door-to-door collection, where individuals work through municipalities to take donations or pay to collect from consumers and businesses, textile items that would otherwise end up in residual waste. Next, would be one of the more well-known methods for textile collection which is textile recycling bins, usually located and controlled either by the municipality that has implemented them or company-controlled collection services. This has different caveats and rules depending on the constituency, region, municipality, or even country. For example, in the Netherlands, Municipalities own the rights and locations that are used for textile collection bins, and companies must bid for certain locations. Bidding can be financial but can also be dependent on the company policies, vision, standings, and commitments to green and closed-loop recycling. All of these factors can come into play when figuring out where a municipality's waste ends up.

Next in the chain are typically sorters. Sorters take the collected garments which are almost always mixed to some degree and work through them to fish out specific categories and types of textiles. The goal of the sorting company is to take something of low value, such as a tonne of mixed textiles, and through the sorting process increase its value. At the end of the sorting process, the output can take many forms. They can be highly pure virgin materials (for example 100% Cotton, 100% Polyester, 100% Cashmir ..etc.), items sorted based on type (T-shirts, Bed linens, Socks, etc.), and many others. It is normal for a sorting company to adhere to special requests from customers to attempt to sort based on a customer's needs. For example, should a chemical recycling company be looking for items of 90% polyester and up, then they would usually approach a sorting company to create this kind of feedstock from the

collected waste they gather. Some of the output streams from a sorter may include, retail stores, charities, brands, and many cases ending up in landfills.

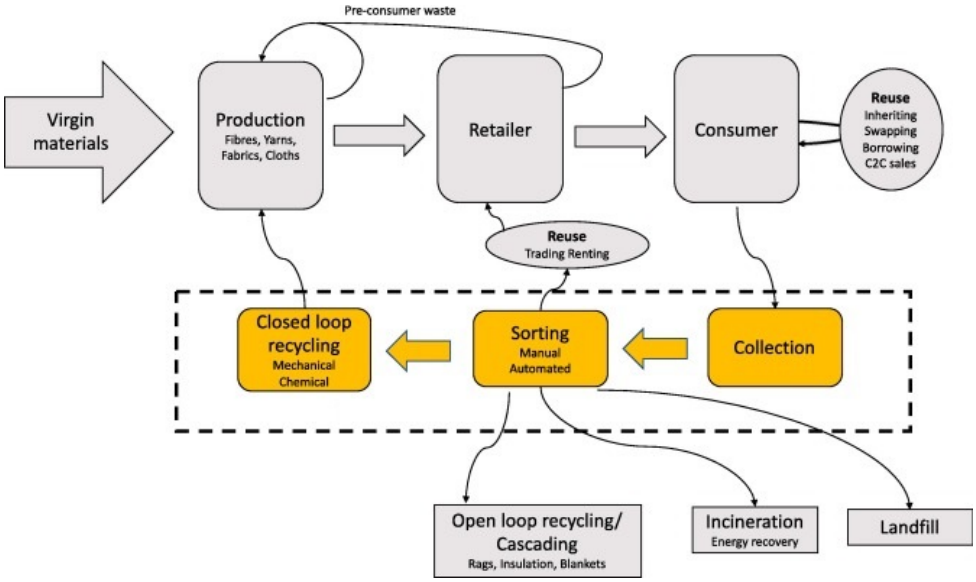

Figure 1: Lifecycle of a garment. Here we see a typical recycling and supply chain in the textile industry. Clockwise from top left: *Virgin matierials* refers to the raw sources such as cotton, poplyester etc., *Production* refers to the brands and roducers that take feedstock from virgin matierials and close loop recycling, *Retailers* receive stock directly from production and second hand clothing supplied by sorting companies, *Consumers* purchase textiles from retailers and other reuse sources such as buying from other consumers. Once a garment leaves the consumer it enters the waste and recycling loop. *Collection* refers to the various ways textile waste is gathered from different consumer sources (Collection bins, door-to-door collection, donations etc.). *Sorting*, being the main context of this project, looks at revaluing the waste through organising bulk garment types and extracting relevant and more valuable items from the bulk waste stream, Sorters have various output destinations including landfills, incinerators and other uses for the sorted garments such as insulation. blankets etc. Finally, we have *closed loop recycling*. Breaking down textiles into raw forms using mechanical and chemical recycling that can be used in textile production and other industries. Adapted from a European study on supply chains in the textile-to-textile field [37]

In light of this, several policies have been introduced to mitigate this issue with the goal being a circular textile industry. One such policy has been around for some time, Extended Producer Responsibility [5] was originally conceived in 1975 and adopted by the EU formally in 2011, the policy looks at the producers of goods as responsible for the waste their companies produce and must provide appropriate collection system, recycling, and reuse of their products. Specifically in the textile industry, EPR was first introduced in France in 2011 [4], this outlined the conditions of EPR for the textile industry. The bill was then introduced in the Netherlands in July of 2023 outlining the steps applied to the Dutch textile industry [26]. It states that "..from 2025, 50% of all textile waste must be prepared for reuse and recycling."

With the current state of the art in the textile industry falling in the fields of Robotics [46], [34], Near-Infrared Spectroscopy (NIR), [40] and Computer Vision (CV). While many large players are in a position to leverage new technologies through heavy investment, the overall problem is one that both large and small players in the industry must address to make any valid attempt at stemming the flow of textiles into end-of-life areas such as landfills and incinerators.

1.1 Main Problems Facing the textile industry

Throughout the entire recycling process, there is only one point of the chain in which every item is individually touched, this is important because most of the chain works in large batches and bundles making it impossible to even see the garment you are trying to assess.

Figure 2: Extended Producer Responsibility Textile Loop. The core idea is to reintroduce products back into the loop either through brand intervention in returning end of life garments to brands and producers to recycle in house. Or re-introducing the items back into the retail sector to be resold and used by customers again [41]

.

The problem lies mostly with human workers, and with thousands of brands creating millions of textiles, a fast and automated way to address this problem is needed sooner rather than later., which is the sorting step of the recycling chain Figure 4.

One potential solution to this problem is the introduction of automated image recognition and computer vision technologies, using cameras placed above the workers' station can identify brands from garments as the worker sorts through the the stream. This information can then be communicated to the worker to aid in the sorting process. However, the lack of real-world datasets from the sorting tables makes existing logo detection systems based on Artificial intelligence not feasible.

If ERP is to be successful, brand recognition must be carried out at every sorting table where the garments are handled on an individual basis.

1.2 Research Aim or Key Objectives of the study

In order to properly assess the feasibility of applying object detection frameworks to the sorting industry, the first logical step is to gather a dataset of images of textile garments. This will be used for both evaluating existing systems and training/fine-tuning new models.

1.2.1 Application of pre-trained models on unseen datasets

For the first research question, the goal is to assess how the selected pre-trained models perform at the task of logo classification on many unseen logo datasets. Crucially, the context of this research question surrounds the implementation of available pre-trained logo detection models on a variety of logo classes, as in the context of textile sorting there is no predefined class set to capture all branded items. This refers to the fact that in textile sorting, the textile stream is inherently random and although some of the more commonly seen brands can appear frequently in the stream such as Nike, Addidas, or Puma, solely selecting the mode commonly found brands in the stream does not accurately reflect the real data that will be found during the day to day operations of a textile sorting setting. The focus of the experiment is to determine an appropriate and generalized approach available to address the random context of the textile sorting industry concerning brand and logo classification.

1.2.2 Application of fine-tuned models on unseen datasets

For the second research question, we look at how the inclusion of a custom dataset gathered in the context of the setting can improve the performance of the model when applied to images taken in the same setting. The goal of this experiment is to determine whether a fine-tuning process taking various types of models, both specific to logo detection and general object detection models which are then fine-tuned with logo classes, can improve the observed performance of the first experiment. The expected outcome of this experiment is that the fine-tuning step will indeed increase the performance accuracy over the "off-the-shelf" models given that the classes introduced will better reflect the real-time data.

1.2.3 Application of various LLM-based vision models on unseen datasets

For the third experiment, a comparison of available vision transformer models is made to assess the prediction accuracy on unseen datasets. Each of the selected models for comparison is paid systems that use API calls and token-based systems to charge the user on a per-prompt basis. The assumption going into this experiment is that since the LLMs are trained on massive amounts of data both image and text, they may be able to generalize to the problem of brand classification well.

1.3 Contribution of the thesis

For this project, I am working with Erdotex Group, which is one of the largest textile sorters in the Netherlands employing close to 400 employees and has a throughput of about 200 tonnes of textiles a month. Representing a large player in the space is the perfect use case for testing brand recognition technologies in a sorting facility.

The first part of the study is to gather a real-world industry-based dataset in Erdotex. This dataset is openly available as a result of this thesis and contains 1685 images of branded and unbranded items

on a sorting table. This also includes an accompanying CSV file with annotation information on each image. The created dataset reflects a real-world, industry setting where perfect scenarios of clear images and distinct brands are not always clearly visible in an image, as is how most textile sorting settings occur.

The second contribution of this thesis is an examination of available methods used for logo and brand classification on a subset of the gathered dataset mentioned above and other openly available brand logo datasets. The results of this show that existing methods do not perform well on the curated industry dataset.

The third contribution of this thesis is a fine-tuned classification model based on YoloV5 architecture trained with a large clothing brand dataset and the curated industry dataset. This model largely outperformed existing logo classification models on unseen data.

The fourth contribution of this thesis is an evaluation of recent publically available Multimodal Large Language models (LLM) with vision capabilities on the brand recognition task. We found that the application of these models outperformed all previously evaluated approaches when applied to the industry dataset without needing fine-tuning. This is due to the inherent fact that LLMs are trained on massive amounts of diverse data, which allows them to generalize problems well.

In summary, brand recognition is most feasible using vision-capable large language models because the diversity of the training dataset allows it to generalize well into the textile sorting process without needing to gather extra annotated data and fine-tune specific models on the setting. In the context of EPR, this highlights the most logical choice for most textile sorting settings due to the ease of access and minimal setup. Being the closest to a "Plug and play" system that is currently available.

Background and Literature review

2.1 Brand vs Logos for recognition

In the marketing world, many factors come into play as to how a brand is perceived and recognized by consumers in an industry. The logo of a given brand is one tool in a large toolbox of how a company displays its presence to not only stand out against competitors but also to plant the seed of thought into a consumer's head at just a glance. For this project, a distinction must be made between a brand and a brand logo when trying to classify where a garment hails from. While almost all companies have a distinct logo that may be recognizable, not all companies extend their branding to incorporate other easily recognizable elements. A few examples of a brand's logo versus other examples of how the brand is portrayed can be seen in Figure 3. Adidas is a perfect example of this. The clothing company not only has several distinct logos but it also uses other elements of design that are instantly recognizable as Adidas even without seeing the logo itself. The very recognizable "Three stripes" is one such element that extends the branding of an item passed the logo itself. These three stripes can be seen across various product lines produced by Adidas, and while they can show up in many different areas of the garment depending on the design, a common sight is to see the stripes lining the seam of a garment down from top to bottom often on both sides. This can be seen in many different types of garments produced by the brand and depending on the product shows up in different places. In a shirt one might see them down the arms and side of the torso, on trousers they are down the sides of the legs, on shoes they can show up on the back or sides, etc. For the context of brand recognition, this makes creating a training dataset tricky, as there are so many variances of the same element just for one brand. For this project, the focus is on the larger scope of brand recognition, with logo recognition being a specialized subcategory of object recognition.

2.2 Technologies in the Textile industry

2.2.1 Industrial Automation

While human hands do the vast majority of textile sorting worldwide, this does not make the industry immune to automation for efficiency and cost reductions. Generally speaking, the textile sorting process is extremely labor intensive. Due to this companies have invested heavily in conveyor belt systems and general industrial automation systems such as pneumatic air propulsion, Valvan is one such company that builds and deploys such automation machinery into textile facilities[46].

Figure 3: Examples of Adidas logo styles over the years. [25]. Different versions of the logo can also be associated with various product lines. In textiles all versions of the logo can show up in the waste stream, making the problem of brand classification an ever growing issue with every newor revised logo. Each new brand causes brand recognition to be an ever evolving scope for each brand.

2.2.2 Near-Infrared Spectroscopy

Looking first at spectroscopy, NIR infrared radiation was discovered back in the year 1800 when William Herschel described what he referred to as "radiant heat" on a conventional blackened bulb thermometer [29] [10] [23]. Following this, it would be more than 150 years until analytical tools and instrumentation manufacturers could make any sense of this excess radiation in the 1960s. Its first major use case was on the determination of moisture and fat in meat, wheat flour, and intact peanuts. Previously the method to measure these points required measuring the mass loss of a sample after it is heated. The use of NIRS in this context sped up the process significantly while also reducing the sampling errors which were common with mass loss measurement techniques.

After its initial introduction into the industry, improvements in instrumentation, mathematical and analytical techniques such as partial least squares (PLS) made the

Nowadays the use of NIR has spanned many industries and one of the more recent introductions is the textile sorting world. NIR spectrometers, combined with various statistical and artificial intelligence techniques allow workers to get a fast, second opinion on the composition of a garment. As an industry textile sorting is a fast-paced job, where large players deal in throughputs upwards of hundreds of tonnes a month. In this kind of environment every second counts, it is just impractical and infeasible for a worker to look at the clothing tag of every garment that crosses their workbench. On top of the timing restrictions, it is widely known in the textile industry, that about 30% of all textile information tags are labeled incorrectly either due to error or in an attempt to inflate the value of the material. With this in mind, NIR spectroscopy can not only speed up the process of identifying the composition or material type but also increase the accuracy and reliability of the sorting process.

2.3 Computer vision and Object Detection

Computer vision is a branch of artificial intelligence that focuses on the use of images and video to infer patterns and recognize specific features. Computer vision spans a multitude of industries including Automotive, healthcare, environmental monitoring, retail, manufacturing, agriculture, security, Banking, Transport, Construction, sports, entertainment, defense, telecommunications, and many more. New industries are constantly innovating as technology becomes more widespread the adoption of computer vision is only expanding as time goes on.

Computer vision generally takes images as inputs, either in full colour or some subset of colours, and looks to either predict, recognize, classify, or infer some specific information from the input to produce an output of some kind. This output can vary depending on the implementation; In autonomous driving the output could be to activate the brake if the module recognizes an obstacle in its path, in a smartphone, computer vision could be used to grant access to the device using facial recognition, whereas in other contexts CV can be used to gather information based on the environment.

Computer vision techniques have been recently brought into the textile world with works focusing on garment identification such as the famous FashionMNIST [52], and Fabric Defect Detection including stains, tears, etc. Reviews [30], [16], have covered fabric defect detection extensively. Other forms of computer vision applied to the textile industry

2.3.1 Problems facing textile brand recognition

Traditionally, object and brand recognition models require vast amounts of high-quality annotated data. In a perfect world, this effectively captures all classes that can fall into the stream, however, given the inherent randomness of the textile recycling stream, this is a difficult task for two separate reasons. The first is the fact that even if an item of clothing contains a brand that does indeed fall into a class that has been trained into a given model, branded textile items are often uniquely designed. While still displaying examples of logos and text describing a brand, the logos can be distorted or integrated into other designs which can interfere with the accuracy when the model is trying to predict new unseen versions of the logo. In Figure 5, we can see 6 different examples of textiles with different types of Adidas logos. Garments can have simple and complex designs.

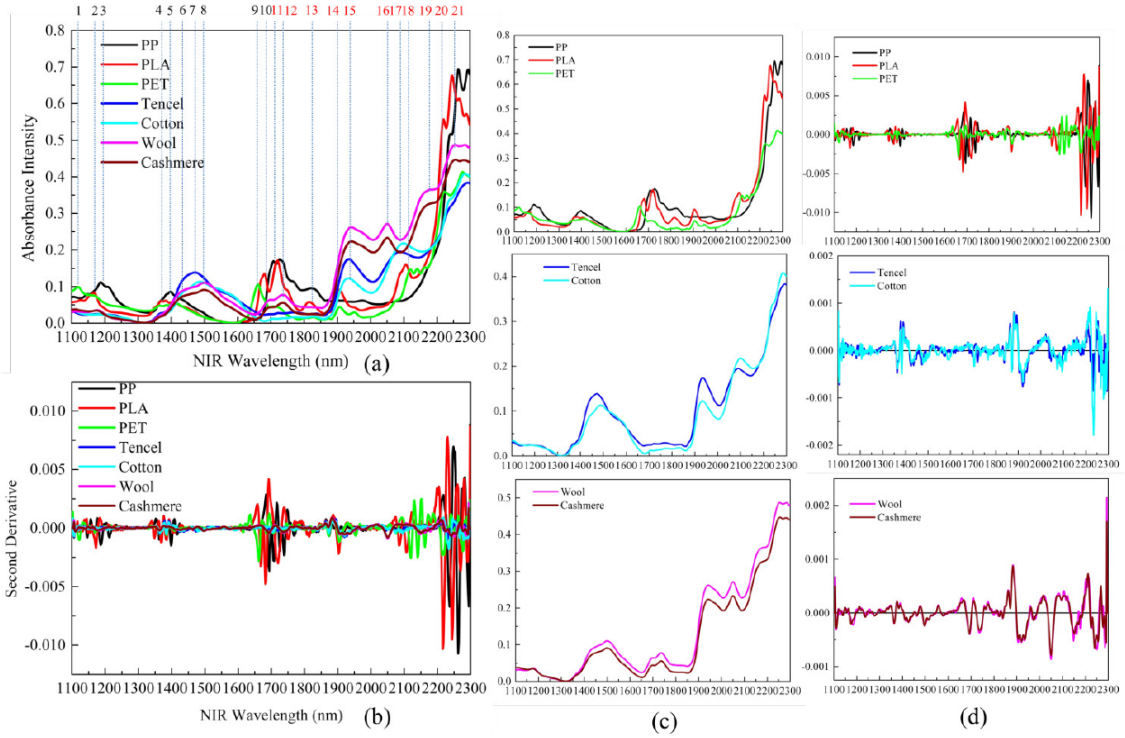

Figure 4: An example of a Near-Infrared Spectroscopy study done on various textile samples using a NIR spectrometer in the range of 1100nm - 2200nm. The Y-axis denotes the light absorbtion rate and the X-axis denotes the wavelength at which the measurement is taken. The peaks and troughs represent areas of high and low absorbance of light at the given wavelength. This represents a unique fingerprint for various molecular compositions, that can then be recognized trained into a model, and recognized efficiently. (a) Shows a comparison of 7 different textile types and their respective spectral reading. (b) Shows the second derivitive of the spectral reading in (a), the second derivtive gives a different view of the spectral scan for deeper analysis. (c) and (d) show comparisons between subgroupings of the textiles in first and second derivative. Textile groups being compared in each graph are plastics [PLA, PP and PET] on top, Middle are organic plant based [Tencel, Cotton] and bottom are animal based [Cashmere, Wool] [53]

Figure 5: Different versions of the Adidas Logo. Designs are often very unique on textile garments. Both in terms of colours and styles but also in overall design. Logos can be very clear or hard to read. Images were taken onsite in industry setting. Blue tint in left most images is due to camera colour correction with specific colours.

Secondly, while it might be possible for datasets to be curated and created on some of the larger, well-known brands. The problem becomes infeasible when one must consider every possible textile brand that may enter the recycling stream. Assuming no pre-made dataset is available for a particular brand, then the creation of such datasets is a time-consuming, labor-intensive process that relies heavily on luck. If a textile sorter is large enough to capture multiple examples of a brand, this must also be extracted from the sorting stream efficiently. If a sorting company is not large enough to dedicate time and resources to finding and annotating custom datasets, the pure scale of the problem is simply too large to address with standard machine-learning approaches.

2.3.2 Logo Detection using Deep Learning and Machine Learning

A survey [38], describes a mature field of machine learning methods applied to logo detection. Many of the systems and algorithms within the field fall into either the general topic of object detection or specified Logo Detection. Within the subfield of Logo Detection, some state-of-the-art algorithms have through the years been used as the backbone for many use case-specific variants. Region-based convolutional Neural Networks are one such backbone, although itself a variation of the convolutional neural network it applies a system called selective search [44] to extract Regions of interest (ROIs), in other words bounding boxes. This was then improved multiple times with adaptations for specific contexts. Fast R-CNN [9] Improved on the performance of R-CNN by only passing the image through the neural network once instead of passing each ROI through the neural network like the latter, these are then pooled at the end of the run instead of during, vastly improves performance. Faster R-CNN [36] removes the need for selective search at the end of a run by baking the region generation into the neural network itself this method is called a Region Proposal Network (RPN). Mask R-CNN introduces instance segmentation and a new method to replace the pooling called ROIAlign, allowing the model to represent subregions of pixels.

You Only Look Once (YOLO) is an object recognition framework originally introduced in 2015 [31] that addresses object recognition tasks by structuring it as a single regression problem predicting bounding box coordinates directly from image pixels. This was a novel approach that differed from classic object detection methods which used repurposed classifiers applied to object recognition tasks. Since its original conception, there have been many major revisions and novel approaches to improve performance and

efficiency. YoloV2 [32] achieved performance increases by introducing batch normalization and anchor boxes to predict bounding boxes, this model was trained on the ImageNet [6] and COCO datasets [19] and also making use of a novel Darknet-19 network. YoloV3 [33] introduced Darknet-53 network as the backbone achieving higher efficiency, it also included an improved loss function for convergence improvement, Feaure Pyramid Networks for improvements on small object detection, multi-scale for recognizing objects of different sizes in images. YoloV4 [2] used complete IoU (Intersection over Union) loss function, a new backbone network called CSPDarknet53, new data augmentations techniques called Mosaic and Self-Adverserial Training. YoloV5 [45] was the first revision released by ultralytics and generally the most widely used due to its ease of implementation and different model versions available. YoloV6 [18], YoloV7 [48], and YoloV8 [13] introduced incremental improvements in recent with notable additions such as extended efficient layer aggregation networks (E-ELAN), training bag of freebies and general usability improvements and optimization in Yolov8. These are just two of the most well-known frameworks in the object recognition space. Many more can be found in Deep Learning for Logo Detection: a review [11].

2.4 Transformer Based Approaches

Originally introduced in 2017, the transformer [47] has become one of the most influential architectures in recent years. Acting as the core backbone of many large language models, it has enabled the ability to allow text prediction to be the base of many problems in the world today. Since its introduction, many revisions have come forward adapting transformers to various tasks.Image recognition tasks [8], Object Detection [3] [28], 3D Object detection [24], transformers for small object detection [35]. Many more object detection applications using transformers can be found in Object Detection with Transformers: A review [39]. More recently, the release of vision-capable LLMs such as OpenAI's GPT4o [27], Google Gemini vision [42] and Anthropics Claude 3 opus [1] have simplified the world of computer vision and brought it to the masses through prompt based image queries. This is just a small sample of the use of transformers for vision tasks,

2.5 Logo/Brand Datasets

With computer vision problems at present, the majority of computer vision tasks rely on vast amounts of domain-specific data to address a problem in a specific field. There are several large open and public datasets submitted to the field to help progress the industry as a whole. Alongside this, many companies and institutions also produce or create their context specific datasets to train their models. Table 1 describes a large collection of some of the publicly available datasets commonly used for various computer vision tasks.

2.6 Evaluation Datasets

With the goal of the experiment ultimately being the ability of each model to predict branded classes from images of textiles in an industrial setting, four datasets were selected with varying amounts of logo classes and settings. Since the setting of textile sorting is inherently random, it is unlikely that in any given model, that its prediction classes will always cover what is in the evaluation dataset. Meaning if a model is implemented, it is unlikely that the training classes will match what is present in the evaluation sets.

2.6.1 Belga Logos

The Belga Logos dataset [14] contains 37 different classes of logo categories taken from real-life situations annotated with human-made bounding box labels and a total of 10,000 images in the training and test sets combined. For this experiment, however, the BelgaLogos dataset was reduced to only include textile-based brands, [Adidas, Puma, Nike, and Umbro], totaling 712 branded images.

Table 1: Statistics of Existing Logo Detection Datasets. Adapted from Deep learning for Logo Detection: A Survey [11]

#Scale	#Datasets	#Logos	#Brands	#Images	#Objects	#Public	#Year
BelgaLogos	37	37	37	10,000	2,695	Yes	2009
FlickrLogos-27	27	27	27	1,080	4,671	Yes	2011
FlickrLogos-32	32	32	32	2,240	5,644	Yes	2011
MICC-Logos	13	13	13	720	—	No	2013
Logo-18	18	10	10	8,460	16,043	No	2015
Logos-32plus	32	32	32	7,830	12,302	No	2017
TopLogo-10	10	10	10	700	—	No	2017
Video SportsLogo	20	20	20	2,000	—	No	2017
VLD 1.0	66	66	66	25,189	—	No	2019
SportLogo	61	31	31	2,836	Yes	Yes	2020
VLD-45	45	45	45	45,000	—	No	2020
Logo-160	160	100	100	73,414	130,608	No	2015
Logos-in-the-Wild	871	871	871	11,054	32,850	Yes	2017
QMUL-OpenLogo	352	352	352	27,083	—	Yes	2018
PL2K	2,000	2,000	2,000	295,814	—	No	2019
FoodLogoDet-1500	1,500	—	559	99,768	145,400	Yes	2021
Open Brands	1,216	559	559	1,437,182	3,113,828	Yes	2020
Logo2k+	2,341	2,341	2,341	167,140	-	Yes	2019
LogoDet-3K	3,000	3,000	2,864	158,652	194,261	Yes	2020
PL8K	7,888	7,888	7,888	3,017,146	—	No	2022

2.6.2 Logo2k dataset

Logo2k+ dataset [50] contains 167,140 images with 10 root categories and 2,341 leaf categories the 10 root categories are Food, Clothes, Institution, Accessories, Transportation, Electronic, Necessities, Cosmetic, Leisure and Medical. for this experiment, only the clothing category was considered. this leaves a corpus of 26,514 images across both the train and test set.

2.6.3 Logo3k Dataset

Logo3k Dataset [49] contains a total of 158,652 images across 9 root categories Clothes, Electronics, Food, Leisure, Medical, Necessities, Others, Sports, and Transportation. 2864 individual brands span these categories however once again for this experiment only the clothing category was considered. This reduces the total image set down to 31,266 while still including 604 distinct brands

2.6.4 Logos in the wild

Logos in the wild [43] introduces a novel dataset of 871 different brands and 11,054 images containing 32,850 unique regions of interest. This dataset will be used as a training set in Experiment 2 focused on tuning an existing Object recognition model with a logo dataset. For this experiment, we will only be using the clothing-related brands from the dataset which make up 559 training and 307 validation images.

2.7 Selected Models for comparison

For this experiment, the aim is to assess the ability of "off the shelf" models to predict and classify whether an image contains a brand and if so classify the brand accordingly. This experiment works from the viewpoint that no extra data is available for fine-tuning to assess the ability of any company or individual to implement such a classification model with no prior knowledge or ability to build further training data. Therefore "Off the shelf" in this context refers to models that require no extra training or fine-tuning being carried out before the deployment of the models. This is to evaluate the likelihood that when policies such as Extended producer responsibility come into play, it is even possible for the industry to adhere to and aid in the capture of branded items to be returned to the original producers.

2.7.1 DeepLogo

The DeepLogo Model [12] uses the Tensorflow Object Detection model with Single Shot Detection for the backbone as a base model that is then fine-tuned on the publically available Flickr27 logo dataset. Tensorflow Object Recognition is an open-source platform that provides functionality to deploy different object detection base models as well as functionality to fine-tune the base models with further context-specific custom data.

2.7.2 LogoDetect

The LogoDetect Model is implemented through the Roboflow platform which allows for models and datasets to be hosted and inference carried out through either API calls or natively within the platform. The model in particular utilizes Roboflows own object recognition model. Information on the architecture of the model itself is not publically available however a forum discussing its use in the research mentioned that when referenced in research the model is similar to Vision transformer and uses vit-base-patch16-224-in21k, i.e. the imagenet dataset as a base model [51], [7], [8]. A commenter representing Roboflow mentioned that there is also customization added by the Roboflow team which is as far as they go to describe the inner workings of the base model. This base model is then fine-tuned with user-uploaded data, in the case of this particular project the model was fine-tuned with two separate datasets totaling 31,044 annotated images. While not explicitly mentioned, part of this dataset used to fine-tune the model contains the Belga Logo dataset (9,843 images) and a further 24,960 images from unknown sources spanning 352 logo classes and at minimum sampling from Belga Logos and Logo32 datasets.

2.7.3 GPT4o

The final model chosen for the first experiment is the GPT4o model released by OpenAI [27]. The "o" in the GPT4o model name means to Omni, which refers to the types of data that were used to train this particular transformer vision model. Traditionally, vision transformers are created with separate architectures for the LLM, Vision and STT/TTS each feeding in to each other sequentially. The difference with GPT4o is that the model was trained on all three types of data at the same time. GPT4o is currently proprietary to OpenAI and therefore little verified information is available for the public.

2.7.4 Yolov5

Yolov5 by ultralytics [45] is a very popular choice among object detection tasks due to the inclusion of the model in the pytorch library for easy integration and loading. YoloV5 has been optimized for fast inference and training and provides clear documentation and frameworks for applications in a variety of object detection tasks.

2.7.5 Gemini 1.5 Flash

Gemini 1.5 flash represents Google's second-best vision model available today. It is faster and cheaper than its bigger brother Gemini 1.5 Pro. While pro is trained on more parameters, currently undisclosed, making it better at complex reasoning. For our experimental purposes, Gemini Flash might still be a viable option. Assuming that it has seen enough examples of textile brands, it might be able to generalize to the problem well.

2.7.6 Claude 3 Opus

At the time of writing, Claude 3 Opus [1] is currently Anthropics Flagship multi-modal model. Beating out GPT4o in several areas its a valid competitor in the space, and might prove to be effective at brand recognition. Claude 3 boasts a massive context window of 1 million tokens and has achieved exceptional results in vision tasks in categories like Math and reasoning, Document and visual Q&A and chart interpretations.

Methodology

3.1 Dataset Gathering

3.1.1 Industry Dataset

As mentioned in the previous methodology section a custom dataset was gathered and annotated in the industry setting taken directly from the real-world textile stream. 1612 images were collected with varying levels of clarity and readability to assess the model's ability to identify branded materials without interrupting the worker's general day-to-day activities. As the images were collected the worker was instructed to carry out their task as they always would while spending a little bit more time turning and maneuvering the textile items to get varying degrees of readability from the images. The first preprocessing step carried out on the industry images was manually rotating the images to be upright (as per the camera's view) as this proved crucial for some of the model's ability to recognize and identify branded materials in initial testing, given that most Logo detectors are trained on standardized images which are always upright. Data augmentation steps to rotate training images and increase the robustness of models are possible however to make the experiments fair for all models the images were standardised.

For the annotation of the data, an item is labeled with a brand if, to the human eye, the brand is recognizable. While this may be difficult for some of the models as recognizing small logos proves to be a difficult task in object detection, the study aims to assess the effectiveness of computer vision models to recognize brands in a real-world setting. Therefore, the consensus when annotating the dataset was that if it could be recognized by a human, then it should be labeled as branded. it is assumed that such a system is expected to at best match a human worker's ability to distinguish branded items from non-branded items.

For the gathering of the dataset, a USB camera was mounted above the sorting area facing downwards onto the worker's desk, to capture a flat image of the garments in question. During the collection and in a typical sorting setting, the item is turned over/maneuvered several times to assess the quality and type of garment under scrutiny. This process can be done at most for a few seconds before the worker has determined the category of clothing. This step is important as in most publically available datasets the image of the brand/garment is typically quite clear which does not reflect the fast-paced movement of the sorting process. A Python tool was created to trigger the camera to take a photo on button press, along with text fields to annotate and fill out information on the garment. During the initial gathering of the dataset, the following fields were noted and saved with a garment information CSV:

- **ID** (A unique id corresponding to the filename of the image), Garment Type (T-shirt, Hoodie, Trousers etc.)
- **Color** (All visible colours in the garment, starting with the most prominent and less prominent in descending order. Multi-coloured items were separated using a semi-colon so as not to disrupt the formatting of the csv file.),

- **Brand** (The associated brand contained in the garment, Should no brand exist this is set to unbranded.)
- **Features** (All notable features in the image, this includes both garment features such as zips, pockets, buttons etc, as well as whether hands or head are visible in the image.)
- **Quality** (The perceived quality of the garment during data gathering on a scale of 1 to 5. features such as stains, tears etc negatively affect this rating.)
- **Damage** (any noticeable damage points on the garment such as 'worn printed design', 'Stain', 'Tear' etc. If no damage exists this is set to 'None').

After the initial round of gathering and annotation, the dataset went through several further rounds of cleaning and preprocessing. This was primarily to clean up any human errors on initial data entry. For example, with each new item, the information on the garment was entered before the images were taken and a batch of images was captured of the same garment in different positions. If for example during the maneuvering of the garment the brand associated with the garment is no longer visible in the images, then the information is updated to be 'Unbranded'. Some special cases make it difficult to determine whether an item is branded or not. For example, when the back of the item is being shown to the camera it is not possible to determine the exact brand in some cases where the only visible branding might be a logo on the front side of a garment. In most cases, this is straightforward forward and the item is labeled as unbranded to the human eye, however, in cases where the garment contains features or branding that may not necessarily correspond to an exact logo makes this more difficult to determine. One example of such a case is with Adidas-branded items. Adidas has another form of well-known branding being the very recognizable "three stripes". While many will be able to easily recognize the three stripes as being associated to Adidas, this becomes quite difficult for a model to recognize without vast amounts of training examples. A human may be able to easily infer the brand from features such as this, the problem becomes more complex when all three stripes are not visible. A human might be able to tell that an item is Adidas when only one or two of the stripes are present, depending on the placement (Such as down the arms/ sides of a sweater which is quite common) this might be obvious to a human but without sufficient training, data will be almost impossible to infer for most models. This in mind for the experiments if a branded logo was not visible then the item was marked as unbranded.

To prepare the industry dataset for fine-tuning using the different model architectures, an application was developed to enable the images to be annotated with bounding boxes where appropriate. This involved loading the image and displaying on a graphical user interface where the user can manually draw a bounding box around the contained logo. It should be noted that for this project only one example of a logo was annotated. While some items may contain multiple examples of branded features and some models can handle multiple classifications on a single image this is out of the scope of this project. The created program for annotating the images takes four-pixel point coordinates based on the user input. Once general annotation of the bounding boxes are created this can be manipulated into various forms depending on the required format of the model at hand. The gathering and annotation of the industry dataset represent the first contribution of this thesis.

3.2 Model Architectures

3.2.1 Single Shot Multibox Detectors (SSD)

The DeepLogo implementation uses a Single Shot Detector architecture [21] which is a pre-trained object detection model from the Tensorflow Object Detection API. For the DeepLogo implementation the model uses a pretrained checkpoint of the SSD architecture trained on the MS COCO dataset. This sets up the model base as an object detector, which is then further fine-tuned with the Flickr27 dataset. The Signle shot detector architecture can be seen in Figure 7. The network has a number of key components. Taking an image as an input it first passes through the base network which is a shortened version of the VGG-16 network this works as a feature extractor. This is then passed on to a series of more convolutional layers allow the model to catch features of different depths and sizes as it works through the network. Finally, the model contains a number of localization and classification layers which create the bounding boxes and confidence scores for each of the classes. The non-maximization layer removes redundant bounding boxes only to show the highest confidence predictions.

Figure 6: A subset of the industry dataset displaying 4 different branded garments. From top: Coca-Cola, Calvin-Klein, Adidas, Levis. Different lighting effects on the images are due to the auto-focusing camera adjusting to movement. The items are turned

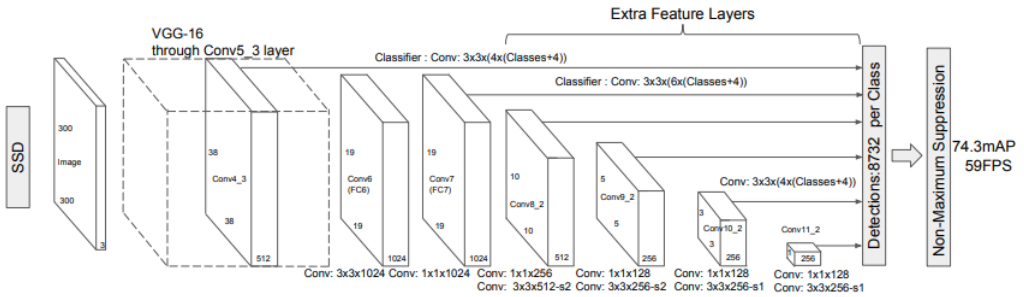

Figure 7: Single Shot Detector Architecture introduced in 2016 [21]. The network has a number of key components. The first block of the network is a VGG-16 Base network used as a feature extractor. Next a number of different convolutional layers used to reduce the dimensionality of the feature space and create more depth which allows the model to extract features of different sizes. Next, several localization layers used to generate bounding boxes and classification layers to map to the bounding boxes. The non-maximization suppression layer removes detections that overlap retaining only the ones with the highest confidence scores.

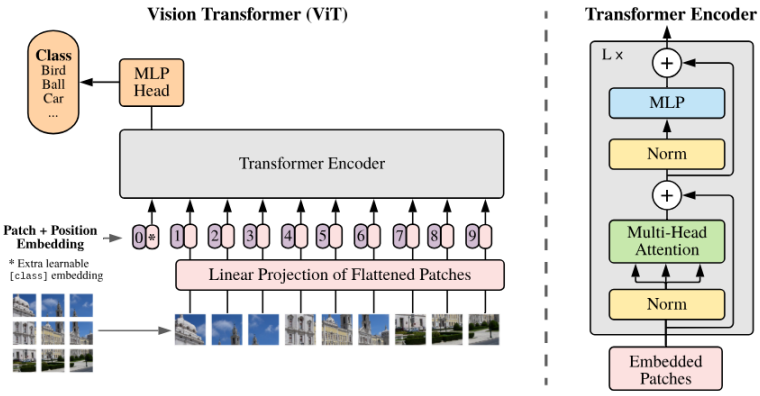

Figure 8: Vision transformer Architecture introduced in [8]. The model uses a combination of Positional embeddings on split image patches and feeds sequences of the images through transformer encoders. Left side of the image shows an example of how an image is broken up into segments and fed into the transformer encoder architecture using patches and positional embeddings. The MLP head converts output into different classes. On the right we can see a more detailed architecture of the transformer encoder. Embedded patches with positional embeddings are passed through normalisation layers and a multi head attention block, allowing the model to process multiple parts of the image simultaneously in a feed forward network. The model has multiple layer normalisation blocks which help stabalize the training. The Multi-Layer perceptron processes the output and converts to different output classes.

3.2.2 Vision transformers (VIT)

Vision transformers, originally introduced in 2021 [8], follow a simple architecture design so that optimizations and architecture created for Natural language processor (NLP) based transformers will be able to be applied to this network with little to no changes. Since the original transformer architecture expects 1D input sequences, the 2D images must be reshaped before being passed to the transformer encoder. It does this by converting the image to a sequence of flattened 2D patches. These are then mapped to patch embeddings which can then be passed to the transformer along with positional embeddings to work with the multi-head self-attention element of the transformer, The transformer is made up of the multi-head attention layer, a feed-forward network, and layer normalization. The classification uses class token representation. The output of the transformer is a fully connected, multi-layer perception. A diagram of the Vision Transformer architecture can be seen in Figure 8

3.2.3 You Only Look Once (YOLO)

The YoloV5 Architecture (Much the same as the YoloV4 Architecture) takes images, patches, and image pyramids as inputs and uses a backbone network called CSPDarknet53 Cross Stage Partial (CSP) refers to the connections between nodes. This backbone converts the images to a feature map. Next, in the neck of the architecture we have a Path Aggregation Network (PANet) [20] that helps with information flow in the network and feature hierarchies. Spatial Pyramid Pooling (SPP) layers are used to identify features at different scales. For the head, YoloV5 uses multiple detection layers so that it can predict objects of different sizes in images. The Yolov4 Architecture can be seen in Figure 9 below.

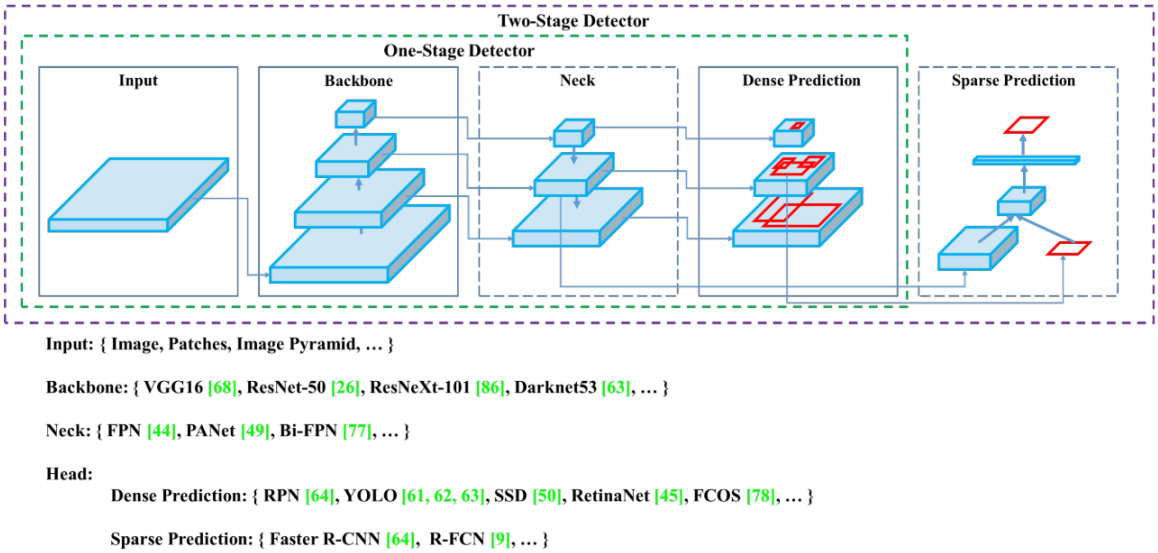

Figure 9: YoloV4/V5 Architecture introduced in [2]. The image shows the general architecture used by YoloV4 and YoloV5. Different revisions of the model components can be seen below the architecture diagram for the input type, Backbone (Used for feature extraction), the Neck (refines the output of the backbone), the Dense prediction module (Makes predictions on the directly on the image) and the sparse prediction layer which refines the predictions made by the dense prediction component. A One stage detector makes predictions in a single pass by combining the neck and Dense prediction laer. Two stage detectors add extra sparse predictions to refine existing predictions from a Dense prediction component to achieve higher accuracy. The YoloV5 model uses a CSPDarknet53 (Cross-Stge Partial) backbone network, Path Aggregation Network (PANet) for the neck, and a dense prediction head in a one-stage detector architecture.

3.3 Evaluation Metrics

To assess the selected models' performance, two categories of metrics are gathered on the resulting classifications. Since the focus of the project is primarily on the ability to carry out brand classifications using readily available models, the resulting metrics fall into two sections. First, is there a logo present in a given image, and second, is the classification of said logo accurate against the true label? For the first category, we will be looking at Accuracy, Precision, Recall, and F1 score. Depending on the dataset being assessed this makes interpretation slightly different. For each of these categories the reason behind choosing these metrics along with the resulting figures vary significantly. To understand the metrics applied to this question first we must outline the definitions of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

3.3.1 Is there a logo present in a given image?

For this category of predictions, the primary goal is to assess whether the given models can capture whether a logo exists in the image or not. While this might seem trivial in a normal setting, the context of clothing logos adds a layer of complexity to the problem that is generally not seen in most benchmark datasets. In the context of textile sorting it is more common than not that the items will be occluded in some way or another. Be it by the worker, moving their head or hands in front of the logos, or if the item is folded in such a way that some of the logo is no longer visible. With this in mind the simple task of 'Does this image contain a logo/brand' becomes crucial to assess before actually classifying it. In this case, the description of the metrics is as follows:

- **True Positive (TP)**: A prediction is classified as "True Positive" if the true label is **not equal** to *Unbranded* and the prediction is **not equal** to *Unbranded*
  e.g True Label = 'Nike', Prediction = 'Nike' / 'Adidas' / 'Puma' etc.

- **True Negative (TN)**: A prediction is classified as "True Negative" if the true label is **equal to** *Unbranded*, and the prediction is also **equal to** *Unbranded*.
  e.g. True Label = 'Unbranded', Prediction = 'Unbranded'.
- **False Positive (FP)**: A prediction is classified as "False Positive" if the true label is **equal to** *Unbranded*, and the prediction is **not equal** to *Unbranded*
  e.g. True Label = 'Unbranded', Prediction = 'Nike' / 'Adidas' / 'Puma' etc.
- **False Negative (FN)**: A prediction is classified as "False Negative" if the True label is **not equal** to *Unbranded* and the prediction is equal to *Unbranded*
  e.g. True Label = 'Nike', Prediction = 'Unbranded'.

Using the above definitions we can now outline the 4 selected metrics and describe why they are relevant to the problem at hand.

$$\text{Accuracy} = \frac{\text{True Positives+True Negatives}}{\text{Total Instances}}$$

Accuracy, in the context of Logo Detection looks at the number of predictions that were made that were accurate over the total number of test cases. Meaning that it measures how many predictions were accurate for both the branded and unbranded items. If a model predicts Nike and the true label is also Nike then this counts towards true positives. if the model predicts unbranded and the true label is unbranded, then this counts as True Negative.

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

Precision looks at the total number of true positives, i.e. the number of correctly predicted brands (Prediction = Nike | True Label = Nike) over the number of positive predictions (True Positive + False Positive). A positive prediction, in this case, is any prediction that is not 'Unbranded'. When looking at our evaluation datasets, before even looking at the results, we can already make an educated guess as to the precision of most of the datasets/models. Since the BelgaLogos, Logo2k, and Logo3k are all logo datasets, they inherently will not contain any unbranded items. This means that for these datasets there will be no False Positive cases. i.e. cases where the true label is 'Unbranded' and the model predicts a brand. So if we take the number N to represent the number of true positive cases and 0 for the number of false positive cases we get the following:

$$\text{Precision} = \frac{N}{N+0} = \frac{N}{N} = 1$$

The only case where Precision is expected to be anything other than 1 is when evaluating the models on the Industry dataset. Since it indeed contains "Unbranded" items.

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

Recall: Recall looks at the ratio of True positives over the True positives + False Negatives. In the context of our experiments, this assesses how well the model performs at logo detection over the entire dataset. To make this a little clearer we can see a filled-in recall equation below. It is important to note here that this is different from the Brand Accuracy metric as for this metric we don't care if the prediction of the brand is the correct brand. So if the model predicts 'Nike' and the actual brand is 'Adidas', then this still counts towards a True positive prediction because it can accurately tell that a brand exists in the image, even if it was incorrect.

$$\text{Recall} = \frac{(\text{True} = \text{'Branded'} \mid \text{Prediction} = \text{'Branded'})}{(\text{True} = \text{'Branded'} \mid \text{Prediction} = \text{'Branded'}) + (\text{True} = \text{'Branded'} \mid \text{Prediction} = \text{'Unbranded'})}$$

F1 Score: Describes the performance of the model when looking at both precision and recall at the same time. It balances the two metrics into a single result. F1 is a useful metric when dealing with an unbalanced dataset, which is the case in our setting. Since we have many examples of branded samples and no unbranded. If a model achieves a high F1 score, then it is good at predicting branded items and unbranded items accurately. In this context, a high F1 score will be the most important metric to consider since the evaluation datasets are imbalanced in their very few branded items in the industry dataset and no unbranded items in Logo2k, Logo3k, and BelgaLogos.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Similarities between Accuracy and Recall Based on the unbalanced nature of the 3 benchmark datasets, not having any examples of Unbranded items, This effectively means that Recall and Accuracy will likely have the same value since True Negative predictions will not occur, True Label = 'Unbranded', Prediction = 'Unbranded' and False Positive predictions will also not occur, True Label = 'Unbranded', Prediction = 'Nike'. In the case of datasets where no unbranded items exist, it effectively reduces the accuracy and recall equations to the same metric:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True negatives} + \text{False Negatives} + \text{False Positives}}$$

$$\text{Accuracy} = \frac{\text{True Positives} + 0}{\text{True Positives} + 0 + \text{False Negatives} + 0}$$

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

3.3.2 Is the predicted label correct?

For this category of predictions, the primary goal is to assess whether the given models can predict the brand contained in the image correctly. We are solely focusing on the accuracy of the prediction against the true label. If a model happens to predict unbranded for the majority of its predictions, and when it does predict the brand it always gets it correct, then the brand accuracy would be equal to 100%. So if a model makes 100 predictions of brands and it gets 75 correct then the brand accuracy is 75%. This metric is the true test of the effectiveness of the models when trying to address the problem of EPR. It tells us exactly how likely it is that such a system can address the problem of catching branded items in the waste stream.

Experimentation

The following section will describe three separate experiments that evaluate brand recognition approaches on different evaluation datasets, A thorough description of each experiment is outlined along with all necessary steps needed to be carried out to reproduce the given results.

4.1 Experiment 1

In experiment 1 the goal is to assess the capabilities of readily available models in the field today. Three separate models were chosen for this experiment. DeepLogo, Roboflow LogoDetect and GPT4o. These three models were implemented and inference was carried out on different evaluation datasets with varying amounts of classes and image clarity.

4.1.1 Dataset preparation

Logo2k: To prepare the Logo2k Dataset for experimentation the dataset was first downloaded locally and a Python script was created to parse the file structure of the dataset and create a feeder CSV file containing the following headers: *File,Full Path,Train/Test,Brand*. File refers to the file name of the image in question, Full Path refers to the location of the image in the local system where it is saved. Train/Test refers to whether the image is used in training or testing (Note: Since this dataset is used for evaluation

only this is not relevant for experimentation, however, this was included based on the initial structure of the dataset.) and the Brand associated with the image. The brand in this case is the true label we will use when comparing the performance of inference. The file structure of the dataset when downloaded contains 5 text files: Logo-2kclasses.txt contains an alphabetically sorted list of all classes contained in the Logo2k dataset, test_images.txt contains the file names and their associated brand directory, test_images_root.txt contains the file names in the test set with their brand directory name and the root category they are associated with (e.g. 'Clothes/Mitchells & Butlers/12.jpg'). train_images.txt and train_images_root.txt follow the same format but with the training images instead. The images are then stored in a separate folder from the root directory. In the 'trainandtest' directory two more folders containing the training set images and test set images are found in 'train' and 'test' respectively. Within these folders, there are root categories such as Accessories, Clothes, Cosmetics etc. For this project, only the clothes category was considered. This resulted in a csv file containing 26513 images in total. A subsequent Python script was created to allow for the feeder csv file to be easily loaded in and converted to a pandas dataframe for experimentation.

Logo3k: For the Logo3k dataset, while much of the file structure remained the same as the Logo2k dataset above some notable differences were present. The dataset contains the same "root category/brand / image.jpg" format however there is no distinction between the train and test sets like in Logo2k. Also different with this dataset for each image there is a corresponding XML file that denotes the annotation related to a specific image. So for example, Clothes / Adidas SB / 1.jpg has a corresponding XML file Clothes / Adidas SB / 1.xml file. Here we can see the associated annotations with that image including brand labels and bounding box labels. The XML file is in PASCAL VOC Format for object detection. A new parsing script was created to generate a CSV feeder file for experimentation. Once again only the clothing category was considered for experimentation. The resulting header for this CSV file was *File,Full Path,Brand, Bounding Box Coordinates* denoting the filename, path to the image locally, true brand label, and extracted bounding box coordinates. Similar to the logo2k dataset, a subsequent Python script was created to allow for the feeder CSV file to be easily loaded in and converted to a pandas data frame for experimentation.

BelgaLogos: BelgaLogos dataset came pre-prepared with a single folder containing all images and several ground truth and query files that outline the annotations for the images. Here the qset3_internal_and _local.gt [17] was used to extract the information to be used for experimentation. The columns of the ground truth file is as follows": 'Instance_name' (Denotes the instance of a specific brand so for example "Adidas_0080" is the 80th instance of an Adidas logo in the set), 'Logo_name' (refers to the associated brand e.g. Adidas), 'Image_name' (The file name for the image e.g. "07638517.jpg"), 'Instance_type' (All values in this field are 'Logo', this is irrelevant for experimentation), 'Instance_state' (Denotes whether the brand is present in the image or not, for this context we only consider entries with 'Instance_state' = 1), and finally the four bounding box values 'X1', 'Y1', 'X2','Y2'. Since the structure of the dataset was straightforward forward a single loader file was created to read in the ground truth file and images. This was then converted to a Python data frame with the following headers: File, Full Path, Brand, and Bounding Box Coordinates. For this dataset, only the following categories were extracted (Adidas, Nike, Puma, Gucci, Umbro, and Reebok) due to them being the only clothing brands present. This resulted in an evaluation dataset of 712 images out of the initial 10,000.

Industry Dataset: Following on from the initial data gathering step which resulted in a dataset of 1685 images in the form of a folder containing the images and a csv file containing the image annotations. The initial data gathering contained the following headers; (ID, Garment Type, Color, Brand, Features, Quality, Damage), however for the first experiment only the File, Full path and Brand were considered. A python loader file was created to accomplish this task. This dataset is returned in pandas dataframe format for experimentation.

4.1.2 Model Preparation

Deep Logo: To use DeepLogo for inference, several steps were carried out. First, the DeepLogo project

was pulled from github and installed locally. Next, The tensorflow models repository was cloned from github to obtain the ssd_inception_v2_coco_2018_01_28 pretrained model. This Single Shot Detector Model was pre-trained using the COCO dataset as a general object detector. Following this, a subsequent step would be to fine-tune this base model using a fine-tuning dataset. Generally, this would be the point where you would prepare your custom use case dataset for further fine-tuning. In this project, however, the authors have also provided a link to a pre-trained model, fine-tuned using the Flickr27 dataset. This pre-trained model comes in the form of a frozen inference graph for use with tensorflow object detection API. Alongside this, a flickr27 label map is also provided which is a tensorflow object detection label map corresponding to the flickr27 classes. For this experiment, a loader model loader file was created to instantiate the model using the inference graph and label map mentioned. Here the class provides some functions for setting up the model and provides an infer function allowing for images to be easily passed in and predicted based on the given classes. In this function, the model makes a prediction and if the predicted class is not in the flickr27 class set then the model defaults to unbranded as the prediction.

Roboflow LogoDetect: The setup of the LogoDetect model from Roboflow is relatively simple, first an account must be created on Roboflow.com to obtain a user API key allowing access to the Roboflow portfolio of models. Generally, this is considered a paid system however for research purposes Roboflow can be used for inference and training for free with certain limits in place, paid tiers allow for more access and higher request limits. For this project, we only need to use the free tier. Once an account is set up and an API key acquired, we must then obtain the model ID which points to an instance of a pre-trained model hosted on Roboflow. Generally, the way this works is that Roboflow provides various model architectures and datasets that can be used to train and fine-tune new instances of models. In our particular case, the chosen project is a logo detection model trained on a dataset uploaded by the author of the project. Initially, this was thought to be a unique dataset of annotated logos however after deep diving into details during implementation, it was discovered that the training data contained at least some instances of images from the BelgaLogos dataset. While this was not explicitly mentioned in the project it is only having seen images from the Belga logos dataset firsthand that this is at all apparent. Continuing with the implementation, once the API key and Model ID are obtained the implementation of the model loader file is quite straightforward. We must first install the inference package to interact with Roboflows backend and the supervision package to correctly handle the detection results. Functions for loading the model and running inference were added to allow for consistent use with the experiment. If no prediction is made on an image then the model defaults to "Unbranded".

GPT4o To make use of GPT4o for image classification an account must be created with OpenAI to obtain a user-specific API Key similar to the Roboflow implementation. Once an account is created and API key is obtained, funds must be added to the account to use the GPT4o API calls. A detailed description of cost breakdown will follow in a later section. Once an API key has been obtained, we can begin structuring the model loader file. When using the API for inference, the GPT4o backend works with requests when dealing with prompts and images. First, the API URL must be specified, in this case, the URL is **"https://api.openai.com/v1/chat/completions"** Next, the image in question must be encoded to base64 format to be passed along with the request. Once this is complete a prompt can be then be created which is to be coupled with the image. During the implementation of the first experiment, the initial prompt was created to address multiple aspects of the image. This was created to align with the data-gathering annotations from the industry dataset, After the inference experiments were completed however these extra annotations were not used to focus on the brand classification aspects of the project. The prompt accompanying the images is:

'Please respond in JSON format only. Please fill in the following values based on the image I have provided: Description (e.g. A stylish winter jacket), Category (e.g. Apparel), Brand (e.g. North Face, Adidas etc | Unbranded if none found), Details (e.g. Waterproof and windproof), Fashion-Trend (e.g. Winter 2024), Price-Estimate (single numerical value e.g. 90 euro), Quality (single numerical value between 1 and 10 e.g. 5/10), Rarity (single numerical value between 0 and 100 e.g. 20)'

Now that the prompt and image have been prepared, the request can be created. Below we can see an example of the request. Here we define the model we would like to use, the user, prompt, and image,

and finally the maximum number of tokens the model is allowed to respond with. In this case, 300 tokens is the standard setting and was unchanged.

```
payload = {
    "model": "gpt-4o",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "Please respond in JSON format only. Please fill in the
                    following Values based on the image I have provided: Description
                    (e.g. A stylish winter jacket), Category (e.g. Apparel), Brand
                    (e.g. North Face, Adidas etc | Unbranded if none found), Details
                    (e.g. Waterproof and windproof), Fashion-Trend (e.g. Winter 2024),
                    Price-Estimate (single numerical value e.g. 90 euro), Quality
                    (single numerical value 1 and 10 e.g. 5/10), Rarity (single
                    numerical value between 0 and 100 e.g. 20)"
                },
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"{base64_image}"
                    }
                }
            ]
        }
    ],
    "max_tokens": 300
}
```

Next, the request can be sent to the API URL and a response is received in JSON format. During the experiment, there were several errors returned while running prompts and images in succession. The majority of these errors were due to too many requests to the API too quickly. To address this, a retry request system was implemented that incrementally increases the timeout between requests up to a maximum of 60 seconds between requests. If a request succeeds, then the timeout is reset and the model can continue with the next prompt and image. This fix was sufficient to complete the experimentation runs. Once a response is received, the JSON is then parsed to extract the predicted brand. If no brand is found then the model defaults to Unbranded.

4.1.3 Experimental Setup

A parent Experiment_1.py file was created to act as the main running file managing the models and datasets described above. Here the various dataset loaders and model loaders are initialised. API keys and URLs are obtained from a single ModelConfigs,json file and passed down during model initialization. Once the models and datasets have been loaded in the experiment starts by randomly sampling the dataset to obtain 1000 images. Since BelgaLogos only contains 712 images when considering the clothing category, this step is ignored and all 712 images are passed. The script then runs for 3 repetitions to obtain an average result when generating metrics later. Crucially a decision was made to repeat the 3 repetitions using the same 1000 images in each repetition. While models such as DeepLogo and Roboflow may be deterministic, GPT4o can return many different responses with the same prompt so it is important to get an average result on the same images. Each image is taken one by one from the datasets and passed to models generating a prediction. These predictions are then added to a new column in the dataset CSV file saved with the appropriate repetition number. A total of 12 result files are created (4 Datasets x 3 repetitions) for Experiment 1. This experiment and the subsequent results denote the second

contribution of this thesis, being an examination of available logo classification methods implemented without fine-tuning steps carried out. Results for this experiment can be seen in 9.

4.2 Experiment 2

For experiment 2, the goal is to assess the performance of the Yolov5 object detection model fine-tuned on 2 separate datasets. The resulting models fall into three categories. Fine-Tuned using a large publically available dataset (Logos in the wild), Fine-tuned using a subset of the gathered industry dataset and finally, a version that is first fine-tuned on Logos in the wild and then further fine-tuned using the industry dataset. It should be noted that originally this experiment was to include a fine-tuned version of the DeepLogo model however after much time spent trying to get this implemented with no success, this unfortunately needed to be dropped due to time constraints. (More details on this can be found below if interested).

4.2.1 Dataset preparation

To be able to use a dataset to fine-tune the Yolov5 model it must be prepared into a specific format. The data must be in the appropriately named YOLO format. First, a .yaml file must be created which denotes the train and validation image directory locations, the number of classes included with the dataset and a mapping between integer IDs and corresponding class names. An example of the .yaml file can be found below. An important point that was not explicitly clear when researching this method was that the YoloV5 model uses the image directory locations to find the corresponding label files. So although it is not expressed in the .yaml file, the labels must be in a directory next to the images directory in the appropriate train and test folders. So an example would be if your images are stored in train/images, the labels for these images must be contained in train/labels. The same for val/images, labels must be in val/labels.

```
path: path/to/parent/directory
train: train/images
val: val/images
nc: 15
names:
  0: Adidas
  1: Coca Cola
  2: Levis
  3: Kappa
  4: Calvin Klein
  5: Puma
  6: Lacoste
  7: Nike
  8: Tommy Hilfiger
  9: Champion
  10: Reebok
  11: Star Wars
  12: Ralph Lauren
  13: Guess
  14: Superman
```

The labels in question must align with the .yaml file where an example of a label and image pair would be train/images/

Logos in the wild: To use the logos in the wild dataset for fine-tuning the YoloV5 model several preprocessing steps needed to be taken. Initially, the format of the Logos in the wild dataset was using Image URLs to download the images however another user created a new version of the dataset since the original download links no longer work. They have provided a zip file containing a complete version of the dataset which can be downloaded from Github. The new version is the one used in this experimentation however both the original and updated sources can be found in the Methodology section of this report.

Once downloaded and extracted, the dataset we are working with comes in the following file structure, the root directory 2 child directories 'brandROIs' (containing folders for each brand which contain images of the regions of interest i.e. a cropped version of the full image showing just the brand, these are not relevant for the experiments), a directory "voc_format" which contains the images and labels in PASCAL VOC format, (similar to the Logo3k dataset mentioned previously, This format has an image file and an XML file of the same name as the label in the same folder), and finally a brands.txt file which contains all of the brand names contained in the dataset.

To prepare this dataset in YOLO format, a script was created to walk through the VOC_format folder, copying images to either the YOLO train or validation destination folders based on a 70/30 split, concurrently the XML Files are also parsed following the same split but converted into label.txt files in Yolo format. An example of the Yolo format is [class_id, x_center, y_center, width, height], a parsed example might look like this *3 0.205 0.114 0.134 0.083*, each file can have multiple lines which can point to multiple objects in the same image. Once the script is finished walking through the image and label files a yaml file is generated using the corresponding class IDs. While not in the initial preprocessing of the data, the resulting datasets were parsed once more to extract only the clothing classes from the dataset. This step was carried out on Google Colab so was not part of the original reformatting file. More details on this will follow during the training section.

Industry dataset

To prepare the industry dataset for fine-tuning, bounding box annotations needed to be added to the existing annotations to prepare the data for use with the Yolov5 model. To achieve this a secondary Python script was created to load in the industry dataset and one by one draw bounding boxes around the logos on each image. This was done using a user interface that loads the image and corresponding information. The user can then create a bounding box around the logo in question and have the coordinates gathered. This tool was also used to clean up some previously missed human errors. Figure 10. Once complete the updated CSV containing the information on the industry dataset needs to be prepared for Yolo format. This involved a similar process as above however adjusted to suit the format that the industry dataset was in. When carrying out this step the plan was to implement a script that would create multiple Object recognition formats for use with several models, and with this assumption, a large conversion script was created that reformatted the industry dataset into multiple Object detection file-formats including COCO format, Yolo format, Manifest format (For use with AWS), Albumenations format, Albumenation annotation formats, as well as creating the label maps and tf_record files for the DeepLogo implementation. However, due to issues with getting the DeepLogo implementation to work, much of this was rendered obsolete so a stripped-down reformat_industry python script was created to only generate the Yolo format. The original script was however used to create a stratified split of the dataset into a balanced train and test set containing examples of brands in both datasets. These Test and Train CSV files were used to create the train and validation sets from the industry dataset for the YoloV5 Model training.

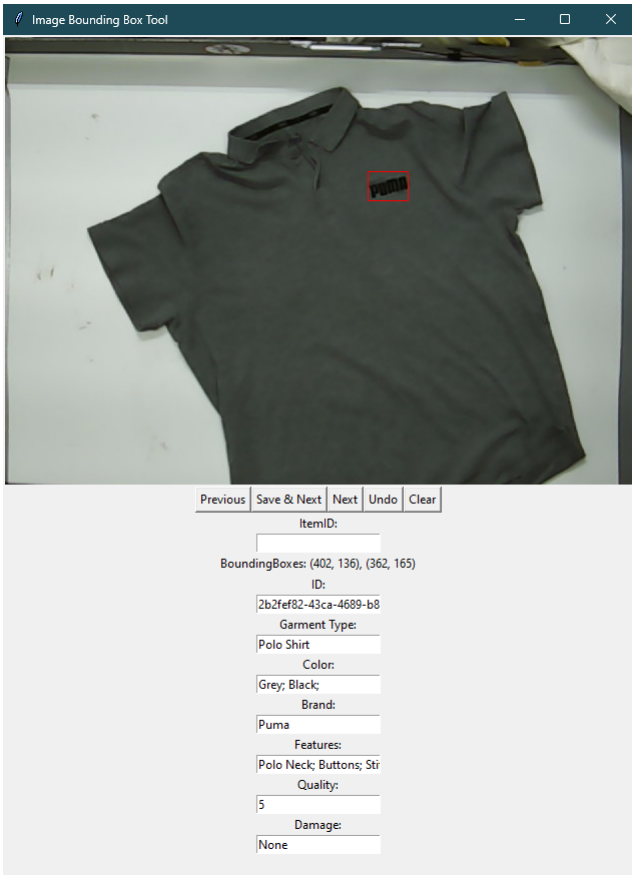

Figure 10: Bounding box tool created to annotate and update industry dataset. The tool loads in the image information CSV and preloads the information for the user. The user can draw a bounding box using their mouse around the logo present in the image (Seen in red). The information previously annotated for the image is loaded in and can be edited if incorrect. The user can also undo their most recent drawing and try again if a mistake is made. The user can also walk forward or backward through the dataset and update information where where relevant.

4.2.2 DeepLogo Fine Tuning problems

Originally, the plan for the second experiment was to take the DeepLogo model from experiment one and assess the effectiveness of fine-tuning it on the industry dataset. The performance of the industry fine-tuned model was then to be compared against the results from Experiment One to highlight the effectiveness of carrying out such a step in an industry setting. Unfortunately, it proved to be extremely difficult to get running due to several reasons. Firstly, the implementation relied heavily on tensorflows object detection model backend which seemed to have several path and version issues. While trying to get the environment into a state to train the model, several packages would throw errors with incompatibility issues. For example, part of the implementation of the DeepLogo/tensorflow Object Detection setup involved manually generating various .proto files using a package called Protobuf or Protoc. Protoc is a dependent package for tensorflow object detection, however depending on the version of tensorflow installed, the Protoc package (installed in the background with tensorflow) can generate different .proto files. This particular issue ended up causing a loop of installs where the errors after each training attempt would say to either downgrade tensorflow or upgrade protobuf. If protobuf was upgraded and the files generated. The to address the next error Tensorflow would be downgraded and this would cause incompatibility with the protobuf files. When this happened, an attempt at implementing in a completely new environment out of the rest of the project code the problems persisted. This was only one of the many problems faced when trying to fine-tune the DeepLogo model. While this particular model is not state of the art, it highlights the issue that at present there are not many options available when trying to apply existing open systems

to a problem such as logo detection. Implementation of older systems can cause major issues relating to codebases no longer being maintained.

4.2.3 YoloV5 Model Training

Once the datasets were prepared for training, an initial attempt to fine-tune the Yolov5 model locally was carried out however this quickly proved to be infeasible as a single epoch took upwards of 4 hours on the full logos in the wild dataset. Following this, a decision was made to migrate the data online to carry out the training using Google Colab and Google Drive to store the data. Although uploading the datasets took multiple hours this proved to be the correct decision as being able to make use of the various GPU accelerators, namely the A100, allowed for multiple training runs for varying numbers of epochs took a fraction of the time it took to run a single epoch on my local CPU machine. Implementing the base Yolov5 model was very straight forward and learning from my previous run I decided to reduce the training and validation set down to only include images containing clothing brands. This required generating a new .yaml file to match the reduced set. The total number of samples in the logos in the wild dataset was reduced from 7756 and 4241 in the train and validation sets respectively, to 559 training images and 307 validation images, and a total of 44 clothing classes.

YoloV5 with Logos in the wild Numerous training runs were carried out and after each one the model's performance was assessed visually with the industry validation set. Since Yolov5 outputs bounding boxes on the images, it was clear to see improvements with subsequent training epochs. The model was incrementally trained using the weights from the last run as a starting point. The model was trained for a total of 200 epochs with a cosine annealing learning rate applied for the second 100 epochs. The training was done in batches, with slight improvements to the annotations each time. The number of epochs in each run was as follows: 20, 40, 40, 60, 40. Once the model was trained to a sufficient level the weights were saved and this was then used as the starting point for fine tuning with the industry dataset.

YoloV5 with Logos in the wild and Industry Dataset Following the initial fine-tuning, two subsequent training rounds were carried out using the industry dataset both for 40 epochs respectively. Both training runs contained the –cos-lr flag for learning rate cosine annealing. After each run the model was assessed on the industry validation set visually to determine whether more rounds were needed. After 80 epochs the model was able to accurately annotate the branded images to a high degree of accuracy.

YoloV5 with Industry Dataset Only For the final training run, the YoloV5 model was fine-tuned using the industry dataset on its own to be able to show the comparison between the three versions of the models. Here a training run for 60 epochs with cosine annealing was carried out. This alone was enough to achieve good results on the validation set based on visual bounding box annotations.

4.2.4 Experimental Setup

For the evaluation of the trained models the various weights files were downloaded from colab and a local installation of the Yolov5 model was used for inference on the same evaluation datasets described in the first experiment BelagLogos, Logo2k, Logo3k, and the Industry data. To load in the pre-trained weights correctly an adjustment needed to be made to the Posix.Path variable, setting it to pathlib.WindowsPath. This was a workaround for a bug where weights trained using a UNIX based system will not work for inference in a Windows system and vice versa. Once the models were loaded in, the results files from the previous experiment were copied and new columns were added for each of the three models. Similar to the previous experiment, the implementation parses through the results files and runs inference on each image individually and the prediction is appended to the file.

For this experiment, a new dataset containing the validation images from the industry dataset was included, as well as the original Industry dataset. Results from this experiment can be seen in the second section of Table 9 and Table 10. The results of this experiment denote the third contribution of this thesis project. The trained Yolov5 Weights will be made openly available for further research.

4.3 Experiment 3

In Experiment 3, the focus was shifted to the use of multimodal LLMs and their effectiveness at brand predictions in images. To explore the promising results from GPT4o in experiment 1 further, this experiment will take two more state-of-the-art LLMs that are vision capable, or in other words, can process images and text. Namely, Claude 3 Opus and Gemini 1.5-Flash. In this section, we will discuss the setup of the models for inference using their respective APIs as well as a cost breakdown on a per-image basis.

4.3.1 Model Implementation

Claude 3 Opus To access the Anthropic API and make use of the newest state-of-the-art model, Claude 3 Opus, an account must first be created with Anthropic to obtain an API key specific to the user. Once an API key has been obtained and funds have been added to the account, then we can make a start on the model loader file. The general methods for sending and receiving requests are the same as with the GPT4o API interactions. in the first experiment. Here the Anthropic Python package must first be installed to create a client variable with the user's API key. Much of the payload for this message remains the same, however, with a slightly different message JSON structure needed for the Claude interactions. Once the message structure is set up, the request is then wrapped in a retry function that increases time outs with each failed request. The prompt was changed to focus solely on the brand(s) contained in the provided image. This change was made to reduce the number of unclear predictions.

```
message = self.client.messages.create(
                    model="claude-3-opus-20240229",
                    max_tokens=1000,
                    messages=[
                        {
                            "role": "user",
                            "content": [
                                {
                                    "type": "image",
                                    "source": {
                                        "type": "base64",
                                        "media_type": media_type,
                                        "data": image_data,
                                    },
                                },
                                {
                                    "type": "text",
                                    "text": "Please fill in the schema I have
                                    provided. What brand do you see in this image?
                                    ONLY RESPOND WITH THE BRAND  YOU SEE NO
                                    EXTRA DESCRIPTION. The only three acceptable
                                    responses are the predicted brand such
                                    as "Nike" etc. NO OTHER TEXT | If you do
                                    not know please say "Unknown" | If no
                                    brand is present please say "Unbranded".
                                    {'Brand': str}"
                                }
                            ],
                        }
                    ],
                    timeout=10
                )
```

Gemini 1.5 Flash To interact with Google Gemini 1.5 Flash model over API, First, an account must be created with Google Cloud and an API key generated. Once the account is created and an API key is

obtained then we can make the model loader file in our local environment. Gemini interactions are done so using the google.generativeai package. Once this package is installed, the gemini-1.5-flash model is selected and setting generation_config="response_mime_type": "application/json". forces the model to respond in JSON format. Once that is set up you must only pass the prompt and image path as strings to the function to receive a response. The same retry logic as used previously was implemented to ensure if any errors were received that the experiment would continue. As with the other models, if an empty response is received this defaults to Unbranded as the returned label.

4.3.2 Experimental Setup

For experiment 3, following the same format as the previous 2 experiments, we first initialize the models using the two created loader files passing the Gemini API Key and the Anthropic API key. Next, each of the 12 results files are loaded and image paths are extracted to be passed to the two models. Two columns are added to the results files, GeminiProVision Brand Prediction,ClaudeVision Brand Prediction, and filled with the response from each of the models. In order to prevent lost data the results files are saved after each inference. The results from this experiment can be found in the third section of Table 9 and Table 10. The results of this experiment outline the fourth contribution of this thesis project, an evaluation of top-performing LLMs in the field at the task of Logo Detection and Brand classification.

Results

5.1 Experiment 1A - Branded vs Unbranded

This section outlines the results gathered during the first experiment when assessing the model accuracy, precision, recall, and F1.

Table 2: Performance metrics of different models on evaluation datasets for Experiment 1 - (Off the shelf models, no fine-tuning)

Models	Dataset1 Logo2k				Dataset2 Logo3k				Dataset3 Belga Logos				Dataset4 Industry			
Experiment 1A	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
DeepLogo	0.158	1.000	0.158	0.273	0.110	1.000	0.110	0.198	0.0028	1.000	0.0028	0.0056	0.479	0.500	0.0096	0.0188
Roboflow	0.281	1.000	0.281	0.439	0.319	1.000	0.319	0.484	0.7725	1.000	0.7725	0.872	0.545	0.875	0.148	0.253
GPT4o	0.866	1.000	0.866	0.928	0.917	1.000	0.917	0.957	0.9513	1.000	0.9513	0.975	0.5553	0.967	0.417	0.577

5.1.1 DeepLogo

Evaluation Datasets The results from the first experiment using the DeepLogo model can be seen in the first row of Table 9. We will look at each evaluation dataset to discuss the performance of DeepLogo. Since DeepLogo is technically the most outdated method used for prediction, it is not surprising to see a poor performance across all of the evaluation datasets. Since DeepLogo was trained using the Flickr27 dataset, it may not have seen enough variety in its training data to be good at logo detection outside of its classes. What is interesting to see is that DeepLogo made far more predictions in the Logo2k and Logo3k datasets than in Belga Logos. This is most likely because the Logo2k and Logo3k datasets are mostly made up of flat, synthetic images such as digitally created logos and the BelgaLogos dataset is made up of images from real-life photographs. Flickr27 dataset is quite small with only 27 classes and 30 images for each class. While the Flickr27 data is indeed from real-life images, the training set may just have been too small to capture logo detection in real-world scenarios.

Industry Dataset DeepLogos performance on the Industry dataset might initially look promising however if we look at the figures on how many predictions were made in branded vs unbranded categories we can see why these might be deceiving. Precision at 0.5 suggests that when it a prediction it gets it right half of the time. But when we look at how many predictions were made by the model it only made 2 predictions. One correct and one incorrect. The recall and F1 score can give us a better idea of the performance with

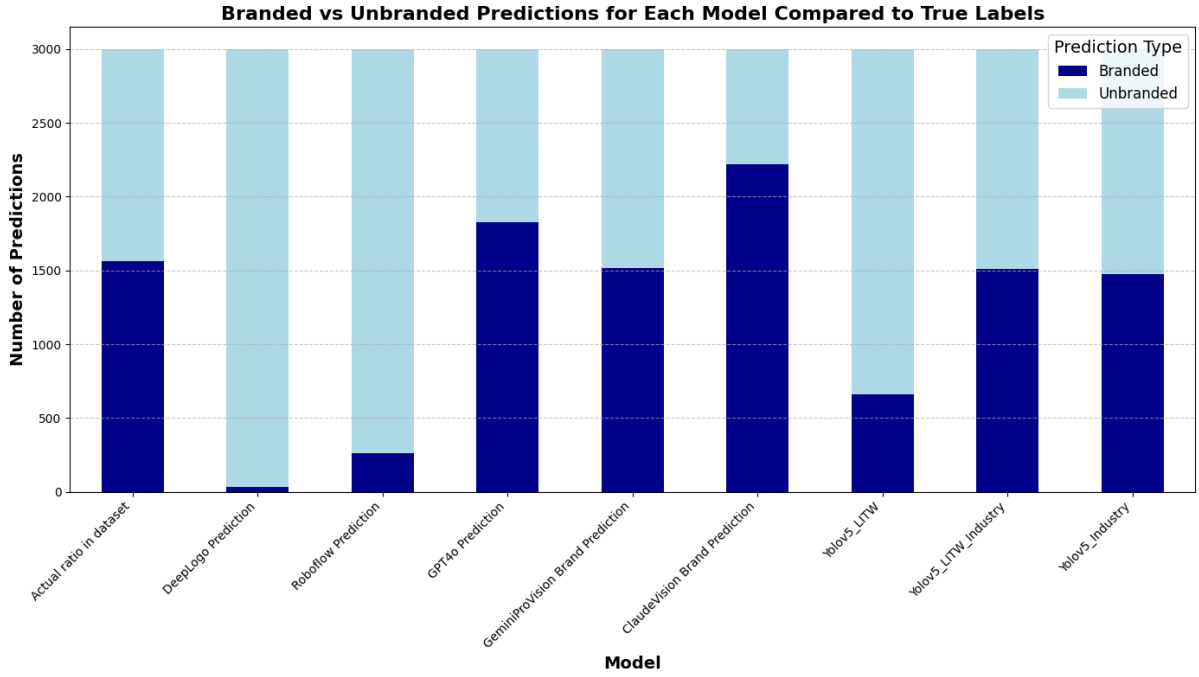

Figure 11: Branded vs Unbranded Predictions for Each Model Compared to True Labels. On the leftmost bar we can see the actual ratio of branded vs unbranded in the industry evaluation set.

Recall and F1 less than 0.02 it suggests that the model does not perform well over the entire dataset at logo detection. We can see the actual ratio of branded vs unbranded in Figure 11.

5.1.2 LogoDetect with Roboflow

Evaluation Datasets While the Roboflow model did perform much better than the DeepLogo implementation, the overall performance was still fairly poor across the Logo2k and Logo3k Datasets. This is most likely because the classes trained into the Roboflow model do not cover the classes in the Logo2k and Logo3k datasets with an F1 Score of 0.43 and 0.48 they do show promising results. The low recall tells us that the model finds it difficult to accurately predict brands in these datasets. As mentioned in the Experimentation section, it was discovered during the implementation that the BelgaLogos dataset was used to train the Roboflow model, So this explains the much higher results here than in any other dataset, The nonperfect score is more than likely because in preparing the BelgaLogos Dataset for experimentation since the clothing category was so small (only 712 images) both the test set and training set were combined.

Industry Dataset The high precision ()and mid-level accuracy (0.5450) on the industry dataset tells us that the model likely predicted Unbranded for many of the predictions. If we look again at Figure 11 we can see this is indeed the case. When it does predict a brand it is only likely to get it correct roughly half the time. This approach brings down the F1 score on the dataset. With only 0.25 for the F1 it is clear that this model struggles to perform on real world datasets.

5.1.3 GPT4o

Evaluation Datasets: As expected, the GPT4o model performed exceptionally well across all three of the evaluation datasets. Its lowest result is on the Logo2k dataset with still a very presentable F1 score of 0.92, this model has consistently high recall and F1 scores across all of the datasets. It should be pointed out that it is highly possible that some if not all of these evaluation datasets could have been used as training data for GPT4o. Without any way of saying for sure we can only take the results at face value

and say that GPT4o is the clear winner in experiment 1 when it comes to logo detection on the evaluation datasets.

Industry Dataset: Since results on the evaluation datasets were so high that it is almost suspicious, the real test is the industry dataset. With an accuracy of 0.55 we can say that the model correctly identifies branded over unbranded items a little over half the time. Indicating that the model has a moderate ability to differentiate between them. While this result is not completely stand out even against the Roboflow model, it is a good mid level accuracy score. The high precision score tells us that when the model does predict a brand it is almost always correct. The recall results for the GPT4o model show a slight drop below half and if we observe the Figure 11 chart and compare the GPT4o Prediction to the actual ratio we can see that GPT4o actually predicted a few unbranded items to contain brands.

5.2 Experiment 1B - Brand Accuracy

The following section examines the results of Experiment 1 using the DeepLogo, Roboflow and GPT4o models to classify brand names accurately.

Table 3: Brand accuracy of different models on 3 evaluation datasets (Logo2k, Logo3k and Belga Logos and 1 custom datasets (Induystry dataset). The brand accuracy refers to the percentage of correctly classified brands against incorrectly classified brands. - Experiment 1

Models	Logo2k	Logo3k	Belga Logos	Industry
Experiment 1 - (Off the shelf models, no fine tuning)				
DeepLogo	0.0127	0.0091	0.5000	0.0000
LogoDetect with Roboflow	0.0178	0.0219	0.4745	0.4156
GPT4o	0.4931	0.8557	0.8475	0.9597

5.2.1 DeepLogo

DeepLogos brand accuracy performance can be found in the first row of Table 10. Overall the model performed extremely poorly in the Logo2k and Logo3k datasets obtaining a result of 0.01 and 0.009 respectively. One might notice a stand out result of 0.5 for the BelgaLogos dataset however when we look at the results in a bit more detail the model only made 2 predictions that were not Unbranded throughout the 700 samples. The score of 0.5 is since one of those 2 predictions was correct. The correct classification was Puma, showing that it can make a correct prediction. This is still not a great result however considering that there are 157 examples of Puma logos in the Belga set. The DeepLogo model showed its poorest performance on the Industry dataset with a total score of 0. Unable to classify a single brand correctly this shows that this model struggles significantly in brand classification both in synthetic and real-world datasets.

5.2.2 LogoDetect with Roboflow

The LogoDetect Model showed similar poor performance on the Logo2k and Logo3k datasets struggling to achieve more than 0.02 the Logo3k set and less than 0.018 on the Logo2k dataset. Potentially this could be due to the mismatch in classes in the training set. The number of classes used to train the model was significantly larger than the Flickr27 dataset used to train the DeepLogo Model however many of the classes were not clothing-based brands. We can see a very large improvement on the BelgaLogos dataset with this model however as mentioned previously this is because the training dataset for the roboflow model did indeed contain images from the BelgaLogos set amongst other unspecified images. Interestingly though the brand accuracy for Roboflow on the Industry dataset was better than expected achieving 0.41

Figure 12: Logo bounding box detections made by the Roboflow model. Here we see three correct predictions and one incorrect prediction. As seen from the top two images, the similarity in colour

on the real-world data. When we dive deeper into the actual results file we see that while the brand accuracy metric is indeed high, it is due to multiple correct classifications in just 2 different brands, Coca-Cola and Adidas. Perhaps if a larger class set is used to train the Roboflow model it may achieve similar results on other classes. The major inaccuracies of this model on the industry set come from the model attempting to classify a logo that is outside of its training classes. However even with classes that are present in both the training set and industry evaluation set it struggled to identify correctly. Overall the results from this model were better than initially expected and if more data is used to train the model on clothing categories, perhaps it could achieve more consistent results in a real-world setting.

5.2.3 GPT4o

GPT4o showed consistently respectable results, with its lowest brand accuracy being the classifications on the Logo2k dataset. Even its lowest score (0.49) tells us that the model is correctly classifying brands in at least just under half of its predictions. performance on the Logo3k and Belga logos are both quite similar achieving Õ.85 on both sets. Again it is quite difficult to say whether these openly available datasets were used to train the model originally but the results cannot be ignored when compared to DeepLogo and Robodlow on all three evaluation datasets, it outperforms significantly. Probably the most surprising result of all three of the experiments shows that GPT4o was able to achieve 95% accuracy on the industry dataset when making predictions. This result is significantly higher than expected because the industry dataset was gathered with a focus of making the images unclear. Even with obstructions and malformations of textiles, the model was still able to achieve a close-to-perfect score on when classifying the brands in an image. As we know from the Logo Detection results from the previous section this is not absolutely a perfect result as it does miss some items that have logos present and classifies unbranded items as having logos. However, this result is still quite significant and suggests a promising solution to the problem of EPR and how to address it.

5.3 Experiment 2A - Branded vs Unbranded

This section outlines the results gathered during the second experiment when assessing the accuracy of the models in predicting the correct brand name. Here we implemented a widely used Yolov5 model trained

on 3 different combinations of datasets. the Logos in the wild dataset, the gathered industry dataset, and finally a combination of the two.

Table 4: Experiment 2 - Performance metrics of YoloV5 model Fine Tuned on 3 combinations of datasets: Logos in the wild (LITW) | Industry dataset | Logos in the wild and Industry dataset in succession. Evaluated on 3 publically available logo datasets (Logo2k, Logo3k and BelgaLogos) and Industry validation set. These results look at how well the models perform at recognizing that there is or isn't a brand present in an image, i.e. its ability to detect brands or logos. Here we look at Accuracy, Precision, Recall and F1 Score. The perfect precision across all the datasets except for Industry is due to there being no 'Unbranded' samples present in that dataset. On the first three datasets, Accuracy and recall having the same value is again due to there being no 'Unbranded' samples which effectively reduces Accuracy to the same formula as Recall.

Models	Dataset1 Logo2k				Dataset2 Logo3k				Dataset3 Belga Logos			
Experiment 2A	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
YoloV5 FT LITW	0.231	1.000	0.231	0.375	0.287	1.000	0.287	0.446	0.573	1.000	0.573	0.729
Yolov5 FT Industry	0.139	1.000	0.139	0.244	0.231	1.000	0.231	0.375	0.389	1.000	0.389	0.560
Yolov5 FT Industry + LITW	0.297	1.000	0.297	0.458	0.395	1.000	0.395	0.566	0.545	1.000	0.545	0.705

Table 5: Experiment 2 - Performance metrics of YoloV5 model Fine Tuned on 3 combinations of datasets: Logos in the wild (LITW) | Industry dataset | Logos in the wild and Industry dataset in succession. Evaluated on 2 versions of the industry dataset. The experiment evaluation set and the validation set (a subset of the dataset that was unseen in training. These results look at how well the models perform at recognizing that there is or isn't a brand present in an image, i.e. its ability to detect brands or logos. Here we look at Accuracy, Precision, Recall and F1 Score. The perfect precision across all the datasets except for Industry is due to there being no 'Unbranded' samples present in that dataset. On the first three datasets, Accuracy and recall having the same value is again due to there being no 'Unbranded' samples which effectively reduces Accuracy to the same formula as Recall.

Models	Dataset 4 Industry				Dataset 5 Industry Validation			
Experiment 2B	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
YoloV5 FT LITW	0.569	0.704	0.297	0.418	0.3382	1.000	0.3382	0.506
Yolov5 FT Industry	0.879	0.906	0.856	0.881	0.978	1.000	0.978	0.989
Yolov5 FT Industry + LITW	0.889	0.907	0.877	0.892	0.9485	1.000	0.9485	0.974

5.3.1 Yolov5 Fine-tuned with Logos in the wild

Evaluation Datasets: On the evaluation datasets, the Yolov5 model fine-tuned with the Logos in the wild dataset generally performed quite poorly. The stand-out performance was on the BelgaLogos dataset where it performed reasonably well. Achieving a recall of 0.389 it still struggled to identify branded items accurately. However the F1 score on the Belga dataset, scoring 0.71, this is the highest non-LLM F1 score on any of the evaluation datasets (not counting the Roboflow/Belga results due to the inclusion of Belga in the training set.) This is a moderately decent result. This high score is not reflected throughout the remaining Logo2k and Logo3k datasets, however. Perhaps due to the mismatch in included classes. Should a larger training set be used this model has good potential to score highly across other unseen datasets.

Industry Datasets:

For this experiment, two versions of the Industry dataset were used to test the fine-tuned model performance. The original Industry evaluation set which is used in all other experiments, and a validation subset of 136 images were used to evaluate the models trained using the industry data. For the main industry dataset, the model was able to achieve a moderate accuracy and precision score of 0.56 for accuracy and 0.70 for precision. Indicating that the model can identify when a logo is present in the images quite consistently. The recall score on this model is much lower leading to a low F1 score. On the industry test set the metrics were slightly different. A slightly higher recall score tells us that this model also struggles with real-world data. The higher overall F1 score is due to the perfect precision, which is since the industry test set only contains branded items. As an overall takeaway, while these results are not the worst-performing models on the industry sets they do still perform quite poorly. This may be since the reduced clothing training set just did not have enough samples to properly generalize to be able to detect clothing brand logos efficiently.

5.3.2 Yolov5 Fine-tuned with Industry Dataset

Evaluation Datasets: The performance of the industry fine-tuned Yolo model was generally poor throughout the three evaluation datasets. This is somewhat expected in the Logo2k and Logo3k datasets as there are far more classes contained in these two than were in the industry dataset. Another reason why performance may have been so poor is that the Logo2k and Logo3k datasets are primarily made up of synthetic images, i.e. graphically designed logos. Since this particular model was only trained on real world examples in a sorting facility the transfer to synthetic images may be difficult for the model. The results on the Belga dataset was the best out of the three evaluation datasets, likely due to the fact that this data contains images from real life scenarios such as sports and photography. It did not however outperform the Logos in the wild model, potentially due to a lack of environmental diversity.

Industry Datasets:

The performance of the model on the industry datasets, full and validation, was the second highest performing model in the entire project. and on the industry test set it was able to achieve an F1 score of .98, while this result is indeed quite high this is more than likely due to the fact that since the industry dataset has only 50 different garments, the model is potentially overfit on the data. While the data in the validation set was not used in training, it did contain the same garments. So potentially the model has learned features such as colours associated with a specific garment and logo bounding box, allowing it to achieve such high accuracy scores. The full evaluation industry dataset was included in this experiment for completeness. The data for this set was randomly sampled but will potentially contain examples from the training and validation set. as well as some images that were not contained in either. Clearly, the use of context specific data has a massive effect on the performance of the model and should be deeply considered in scenarios where gathering datasets is feasible in a setting such as this. Specificallly where generalizability is not essential and its more important to focus on specific brands.

5.3.3 Yolov5 Fine tuned with LITW and Industry

Evaluation datasets The use of both Logos in the wild and industry data combined for fine tuning proved to be effective at improving performance across the Logo2k and Logo3k datasets over their individual fine tuning processes. Recall increasing by at least 5% on Logo 2k over the two individual runs and by nearly 10% on the Logo3k dataset. It shows that the inclusion of synthetic and flat images of logos from logos in the wild in combination with the industry dataset was able to increase performance on both these datasets. The higher recall scores led to an increase in F1 score also. The performance on the BelgaLogos dataset however decreased slightly over the model trained only on the logos in the wild dataset. this was surprising given the fact that the BelgaLogos dataset is primarily real world images. Potentially the blurry annotated data from the industry dataset harmed the models ability to identify logos in settings other than the textile sorting setting.

Industry Datasets

The performance achieved on the industry datasets closely mimicked the results from the previous model trained with the industry dataset on its own. With only a 1% difference in F1 score on both versions of the industry dataset. The model trained on logos in the wild and industry data actually performed 1% better on the full industry dataset and 1% worse on the test set. The addition of the Logos in the wild dataset may have reduced the overfitting effect mentioned in the previous example. A clear takeaway from Experiment 2 is that the inclusion of content-specific data such as an industry dataset has a massive effect on performance overall.

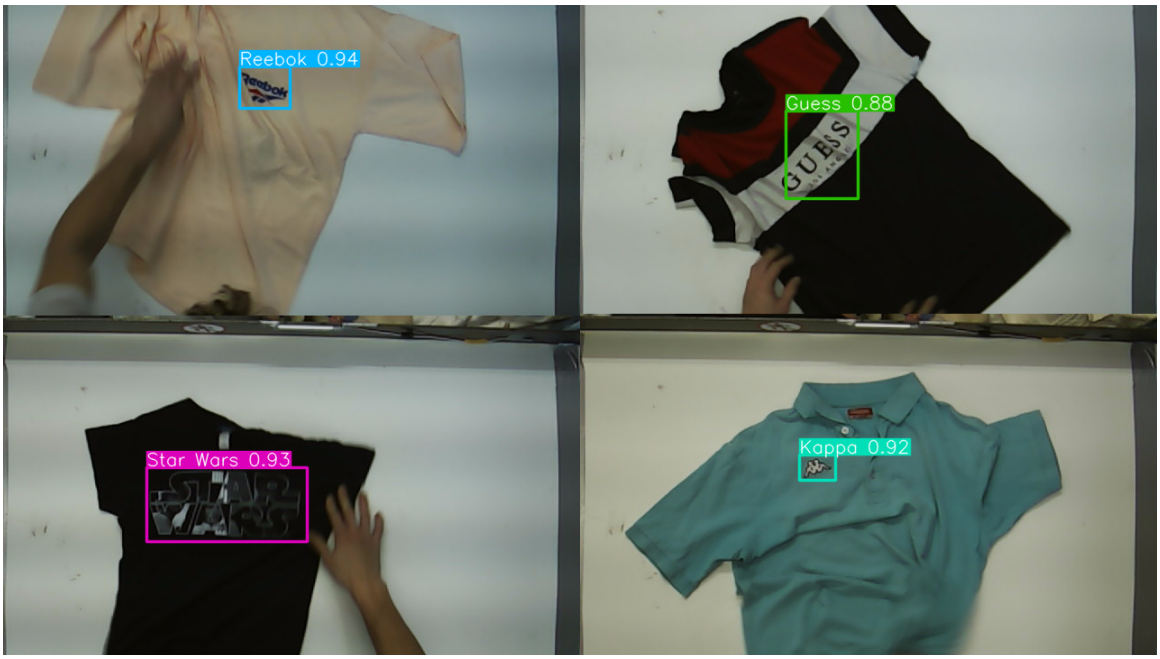

Figure 13: Bounding box annotations on 4 industry textile samples using the YoloV5 Model Fine-tuned on Logos in the wild and industry Dataset

5.4 Experiment 2B - Brand Accuracy

Table 6: Experiment 2 - Brand accuracy of YoloV5 model Fine Tuned on 3 combinations of datasets: Logos in the wild (LITW) | Industry dataset | Logos in the wild and Industry dataset in succession. Evaluated on 3 publically available logo datasets (Logo2k, Logo3k and BelgaLogos) and Industry validation set. The brand accuracy refers to the percentage of correctly classified brands against incorrectly classified brand

Models	Logo2k	Logo3k	Belga Logos	Industry test
Experiment 2 - Fine Tuned Models				
YoloV5 FT LITW	0.0043	0.0105	0.5907	0.3043
Yolov5 FT Industry	0.0144	0.0043	0.1408	0.9850
Yolov5 FT Industry + LITW	0.0067	0.0000	0.2216	0.9845

5.4.1 YoloV5 FT LITW

Generally, the LITW fine tuned model performed quite poorly on the Logo2k and Logo3k datasets. with almost no ability to correctly predict brands in these datasets it is more than likely due to images being

synthetic and the transfer from real world training images does not work well when applied to the synthetic datasets. The performance on the BelgaLogos dataset however was significantly better achieving almost 60% accuracy. This was the highest scoring non-LLM based model on the BelgaLogos dataset and shows a clear link between using real world data to predict on real world data. While this is most likely an obvious conclusion, the vast difference between the Logo2k/Logo3k evaluations and the Belga evaluation confirms this without a doubt. In the same breath we can see that when applied to the industry validation set the performance is not carried over. Whether this is due to a mismatch in class sets or whether the obstructions and malformations of the textiles are what is causing the poor performance is unclear.

5.4.2 YoloV5 FT Industry

The Yolov5 Model fine tuned on the industry dataset performed very poorly across all three of the evaluation datasets. With its highest performance being on the BelgaLogos it still only achieved a mere 14% on brand classification. This is likely due to the lack of diversity in the training set. The model is more than likely overfit to the sorting environment and cannot recognise images in other settings. On the industry test set the model performed extremely well, again giving us the indication that the model is likely overfit on the training data since there are not enough garments to properly test the models ability at generalizing to new items.

5.4.3 YoloV5 FT LITW + Industry

For the model trained on both the Logos in the wild and industry dataset together we see somewhat strange and inconsistent results. For the Logo2k dataset we see a decrease in performance over the industry only model but an increase over the LITW only model, suggesting that the addition of the industry data increased the performance ever so slightly. All three of these models were unable to achieve anything more than 1.4% accuracy on the logo2k dataset further proving that the use of real world data to predict synthetic data does not work well. On the Logo3k dataset the model scored a 0% accuracy unable to predict a single item accurately. On the BelgaLogos dataset the inclusion of the industry dataset seems to have vastly harmed the performance overall as the model dropped from 59% to just 22% after the industry data was added to the fine tuning process. On the industry test set the the combination of the two datasets together for training seemed to have little effect on performance. with the accuracy dropping by a mere 0.0005%, this may have slightly reduced the overfitting of the model but it is difficult to say without more testing.

5.5 Experiment 3A - Branded vs Unbranded

Table 7: Experiment 3 - Performance metrics of different LLMs on evaluation datasets as well as custom industry dataset. These results look at how well the models perform at recognizing that there is or isn't a brand present in an image, i.e. its ability to detect brands or logos. Here we look at Accuracy, Precision, Recall and F1 Score. The perfect precision across all the datasets except for Industry is due to there being no 'Unbranded' samples present in that dataset. GPT4o was included from Experiment 1 for easy comparison with other LLM's

Models	Dataset1 Logo2k				Dataset2 Logo3k				Dataset3 Belga Logos				Dataset4 Industry			
Experiment 3A	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1	Acc	Prec	Re	F1	Acc	Prec	Rec	F1
Gemini Vision	0.652	1.000	0.652	0.789	0.9097	1.000	0.9097	0.953	0.9494	1.000	0.9494	0.974	0.7623	0.955	0.618	0.750
Claude 3 Opus	0.9303	1.000	0.9303	0.964	0.973	1.000	0.973	0.986	0.9906	1.000	0.9906	0.995	0.474	0.909	0.370	0.526
GPT4o	0.866	1.000	0.866	0.928	0.917	1.000	0.917	0.957	0.9513	1.000	0.9513	0.975	0.5553	0.967	0.417	0.577

5.5.1 Gemini Vision

Evaluation Datasets On the Logo2k dataset the Gemini Vision model achieves a good overall recall score of 0.65, indicating that the model can recognize when images contain a brand pretty consistently. This

outperforms all non-LLM's by at least 36%. It is also the lowest score out of the three large language models. When comparing to GPT4o and Claude 3 Opus which both achieved above 85% recall scores it could indicate that Gemini may not have been trained on the Logo2k Dataset. If this were true then it points towards Gemini being able to generalize to unseen data well. The high recall score produces a respectable F1 score of 0.78. If the theory that GPT4o and Claude 3 Opus is correct, in that they were trained on the three Evaluation Datasets used in the experiments. Then this would mean Gemini's score might be the only true results in applying the models to 'Unseen' data.

In the remaining evaluation datasets, Gemini achieves Recall and F1 scores all above 90%. On Logo3k it has a recall of 90% and and F1 score of 95%. the model performs even better on the the Belga data, obtaining a recall and F1 score of 0.94% and 0.97% respectively. Once again this indicates, when looking at the difference between the results obtained on Logo2k dataset versus the score achieved on the Logo3k and Belga data, that Gemini was more than likely trained on the latter datasets. Even though this might be the case, Gemini was the lowest performing of the LLMs on Logo3k, if only by 1 % by GPT4o and 6% by Claude.

Industry Dataset On the industry dataset Gemini performs well at identifying when there is a brand or logo in the image. It beats out all other LLMs in this category and it is only beaten by the YoloV5 Models that were trained on the industry data. The relatively good accuracy score of 76.23% tells us that Gemini can consistently tell between branded and unbranded items. Following on from this, a precision score of 95% means that specifically in the True positive cases, where an item is indeed branded, it is nearly always right. The lowest scoring metric for Gemini was recall. Obtaining a score of 61%, this is still the highest-scoring LLM in this category.

5.5.2 Claude 3 Opus

Evaluation Datasets Claude 3 Opus achieved the highest across all three of the evaluation datasets. Resulting in the highest recall scores and consequently the highest F1 scores in Logo2k, Logo3k and Belga Logos. As with the previous LLMs, having scores this high tends to suggest that the model was trained on the evaluation datasets. However, if the theory is correct, Claude still consistently outperformed the other two LLMs, GPT4o and Gemini Vision in its application to the datasets.

Industry Dataset As with all the rest of the models, the real test is the application to the unseen industry dataset, and Claude actually performed much poorer on this than expected when looking at the results in the evaluation datasets. When it comes to telling when a garment is unbranded or branded the model actually performed the worst out of the three LLMs. Clauds accuracy score, dictating how well the model can identify both branded and unbranded items was 0.47. If we take a look at Figure 11 once again, we can see that the the model is quite eager to predict with brands even when an image contains none. This also contributes to the low recall score of 37% tells us that the model misses a large amount of branded items predicting them as unbranded. These factors combined give us an overall F1 score of of 52% showing moderate performance on the industry set but resulting in the lowest-performing LLM at predicting when an image contains a brand or logo. This is not what is expected when looking at the results on the three evaluation datasets, however, this may also mean that Claude is potentially overfit on the Logo2k, Logo3k and Belga datasets since it was not able to predict accurately in a real-world setting.

5.6 Experiment 3B - Brand Accuracy

5.6.1 Gemini Vision

Evaluation Datasets Gemini was able to score consistently above 70% on all 4 datasets in the context of brand classification with stand-out results being on the Belga dataset (0.92). suggesting that if the model predicts that an item of clothing indeed contains a brand that it is almost always correct in the exact classification of the brand.

Industry Dataset Interestingly enough the Gemini model achieved a very high brand accuracy score on the industry dataset (0.88). This says that when making a prediction, Gemini is correct in its classification nearly 90% of the time on the industry dataset. This is a significant result especially when combined

Table 8: Experiment 2 - Brand accuracy of Gemini Vision and Claude 3 Opus (GPT4o from experiment 1 included for easy comparison between LLM results). Evaluated on 3 publically available logo datasets (Logo2k, Logo3k and BelgaLogos) and the custom industry dataset. . The brand accuracy refers to the percentage of correctly classified brands against incorrectly classified brand.

Models	Logo2k	Logo3k	Belga Logos	Industry	Industry test
Experiment 3 - LLM Comparison					
Gemini Vision	0.7408	0.8721	0.9270	0.8862	N/A
Claude 3 Opus	0.5428	0.8493	0.7131	0.7979	N/A
GPT4o	0.4931	0.8557	0.8475	0.9597	N/A

with the moderate to high scores in the four metrics, Accuracy, precision, recall and F1 from the previous section.

5.6.2 Claude 3 Opus

Evaluation Datasets Claude had a mixed performance between the 3 evaluation datasets, Its highest accuracy score being on the Logo3k datasets with a score of 0.84 and its lowest being on the Logo2k dataset achieving a score of 0.54. Generally, the model performed well overall in its worst case getting more than half of its brand predictions correct, against the true label.

Industry Dataset

On the Industry dataset, Claude achieved a respectable 0.79. While this is still the lowest out of the three LLMs on the industry dataset, it shows that generally, LLMs perform very well in the context of brand classification, specifically on textiles.

5.7 Cost breakdown

Over the experiment, using the various LLMs had costs associated with the API usage. The costs for the Claude 3 Opus model was 94.52, GPT4o was 79.80 and for Gemini 1.5 Flash there was a $300 sign up bonus and only $25.64 was spent. When we take these figures and divide by the total number of images in the experiments. We get the following, note since image size varied from dataset to dataset these figures are averages. Since there was 11,136 images, we can divide the costs by the total images to get the average cost per image. Claude 3 Opus cost $0.0084 per image, GPT4o cost $0.0071 per image and finally Gemini Flash, although technically free due to the sign up bonus, cost $0.0023 per image. Making gemini Flash the clear winner in terms of cost reduction.

Table 9: Full performance metrics of different models on evaluation datasets. These results look at how well the models perform at recognizing that there is or isn't a brand present in an image, i.e. its ability to detect brands or logos. Here we look at Accuracy, Precision, Recall and F1 Score. The perfect precision across all the datasets except for Industry is due to there being no 'Unbranded' samples present in that dataset. On the first three datasets (Logo2k, Logo3k and BelgaLogos), Accuracy and recall having the same value is again due to there being no 'Unbranded' samples which effectively reduces Accuracy to the same formula as Recall.

Models	**Dataset1** **Logo2k**				**Dataset2** **Logo3k**				**Dataset3** **Belga Logos**				**Dataset4** **Industry**				**Dataset5** **Industry test**			
Experiment 1 - (Off the shelf models, no fine tuning)																				
	Accuracy	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
DeepLogo	0.158	1.000	0.158	0.273	0.110	1.000	0.110	0.198	0.0028	1.000	0.0028	0.0056	0.479	0.500	0.0096	0.0188	N/A	N/A	N/A	N/A
LogoDetect with Roboflow	0.281	1.000	0.281	0.439	0.319	1.000	0.319	0.484	0.7725	1.000	0.7725	0.8716	0.545	0.875	0.148	0.253	N/A	N/A	N/A	N/A
GPT4o	0.866	1.000	0.866	0.928	0.917	1.000	0.917	0.957	0.9513	1.000	0.9513	0.975	0.555	0.967	0.417	0.577	N/A	N/A	N/A	N/A
Experiment 2 - Fine Tuned Models																				
	Accuracy	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
YoloV5 FT LITW	0.231	1.000	0.231	0.375	0.287	1.000	0.287	0.446	0.573	1.000	0.573	0.729	0.569	0.705	0.298	0.418	0.338	1.000	0.338	0.506
Yolov5 FT Industry	0.139	1.000	0.139	0.244	0.231	1.000	0.231	0.375	0.389	1.000	0.389	0.560	0.879	0.906	0.856	0.881	0.978	1.000	0.978	0.989
Yolov5 FT Industry + LITW	0.297	1.000	0.297	0.458	0.395	1.000	0.395	0.566	0.545	1.000	0.545	0.706	0.889	0.907	0.877	0.891	0.948	1.000	0.948	0.974
Experiment 3 - LLM Comparison																				
	Accuracy	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
Gemini Vision	0.652	1.000	0.652	0.789	0.9097	1.000	0.9097	0.953	0.9494	1.000	0.9494	0.974	0.7623	0.955	0.618	0.750	N/A	N/A	N/A	N/A
Claude 3 Opus	0.9303	1.000	0.9303	0.964	0.973	1.000	0.973	0.986	0.9906	1.000	0.9906	0.995	0.474	0.909	0.370	0.526	N/A	N/A	N/A	N/A

Table 10: Full Brand accuracy metrics of different models on various datasets (Logo2k, Logo3k, BelgaLogos and Industry / Industry validation set.

Models	Logo2k	Logo3k	Belga Logos	Industry	Industry test
Experiment 1 - (Off the shelf models, no fine tuning)					
DeepLogo	0.0127	0.0091	0.5000	0.0000	N/A
LogoDetect with Roboflow	0.0178	0.0219	0.4745	0.4156	N/A
GPT4o	0.4931	0.8557	0.8475	0.9597	N/A
Experiment 2 - Fine Tuned Models					
YoloV5 FT LITW	0.0043	0.0105	0.5907	N/A	0.3043
Yolov5 FT Industry	0.0144	0.0043	0.1408	N/A	0.9850
Yolov5 FT Industry + LITW	0.0067	0.0000	0.2216	N/A	0.9845
Experiment 3 - LLM Comparison					
Gemini Vision	0.7408	0.8721	0.9270	0.8862	N/A
Claude 3 Opus	0.5428	0.8493	0.7131	0.7979	N/A

Discussion

Based on the results of the above experiments, we can suggest two different approaches to the brand recognition problem put forward by a policy such as EPR. If the context of the problem is based in only a few brands, meaning that a sorting facility has a discrete list of specific brands that it is looking for in the waste stream. Then based on the above results, the most logical choice is to follow a route of fine-tuning with a custom dataset. This process can take a significant amount of time to carry out. From gathering the dataset, annotating it, making sure it is balanced and contains enough examples of each brand and furthermore, each design that a brand might include in their garments. This problem becomes less and less feasible for every extra brand that is added to the list.

In a scenario where many brands must be caught and processed in the waste stream, then it is infeasible to follow the process of data gathering and training a custom model on every brand that shows up in the stream. The process is far too labor intensive and the costs to set up would be far too high even for larger companies. Even just spending the time to find garments of a specific brand relies heavily on luck. In this scenario, the only logical choice is to make use of models trained on massive amounts of data that can generalize well to new unseen data. Since the LLMs are multi-modal, they can infer information from images that would have to be specifically trained into the custom models. For example, text recognition, while it is possible for this to be trained into a custom model, the process of doing so not only requires human labeled data of bounding boxes around textual logos, but even finding examples of such in the textile stream would be very difficult. The feature of text recognition is a built in feature of the LLMs, and where the LLMs may not be able to recognise a logo in the image they might be able to infer the brand based on the interpretation of the text on the garment. For example, if the term 'Just do it' appeared in an image, and a standard object detection model was only trained on the nike swoosh logo, it would never be able to infer the information and relate it to the brand. Where as with the LLM's, in failing to find a logo might still read the text and make the connection to Nike based off the slogan alone.

Conclusion

In light of the upcoming Extended Producer Responsibility, due to come into effect in 2025, which looks to hold producers of textile products as the cause of much of the current waste problem in the textile sector. Producers must provide systems and processes to retrieve their own products back from the waste stream and process them for reuse or recycling in a closed-loop system. The problem of brand identification on textiles in the waste stream is simply too much for any individual human to be able to address. In this thesis, we explored a number of potential solutions that could help aid workers in their ability to catch branded items in the stream. A detailed assessment was carried out to explore the most feasible solution for applying brand recognition to the textile stream. This included the implementation of readily available solutions, with nothing more than installation required. To API based Vision Capable Large Langage models. To implementing a widely used object detection model trained on a custom gathered dataset in an industry setting.

We found that depending on the scope of the problem being addressed there are two potential paths or solutions. First, if the scope of the problem is small, meaning if there are only a small number of brands that must be gathered from the waste stream. Then it is reasonable to suggest training an object recognition model on a custom dataset built around the brands in question. We implemented a YoloV5 model fine tuned on industry specific dataset, and it achieved the highest scoring metrics on a dataset of unseen items from the waste stream. The problem with this is that the process is extremely labour intensive and only increases with each new brand or logo added to the pool. At a certain point, the labor required to cover each brand becomes too much even for large players with deep resources. This is where solution 2 comes in. Making use of recent Vision capable LLMs which can generalize to new problems well.

With very little setup needed, LLM's reduce the complexity involved in implementing a brand recognition system. Not only this, but with good prompt engineering a user can completely redefine the specifications of the system with a simple change of a prompt. Compared to rigid models trained on annotated data this flexibility saves the user many labour intensive hours of gathering, annotating, training, tuning, deployment and testing. LLMs such as GPT4o, Claude 3 Opus and Gemini Vision can even infer brands that may not be that common or recognizable. This is due to the massive datasets they are trained on, likely containing some examples of many brands that some people may never have even heard of.

Continuing even further, LLMs can also tackle other textile based annotations in parallel. By extending the query or prompt in question the information extracted from each image can be increased significantly. For example recognizing zips in a garment. Without having to gather a dataset specifically on zips and then training it into a given object recognition model is possible but requires significant amounts of work. Where as with LLM's all that must be done to receive passable results is to adjust the prompt and parsing of the response to handle new information. The inherent flexibility of LLM's is what stands out as one of the key take aways of the project.

Future work

I hope that this thesis can act as a starting point for the field of brand recognition in the context of textile garments. With LLM's only improving and already at a fairly respectable point future work in this field may surround the application of newer and improved versions of existing and novel LLMs. If more time was available, an attempt at gathering a textile brands dataset would be the first point of call for use with models such as YOLOv5. Using newer versions of this model would also be a viable next step such as Yolov8 with the same training data to assess the model performance improvements on the same problem. An assessment of LLM's capability at other captioning tasks such as Quality assessments, feature classification on items such as zips/pockets/hoods etc, Fault Detection stains/tears/holes and price/value prediction, Bounding box predictions using LLM's. I believe there is much that can be done to progress this field, and polices such as EPR only tend to increase the rate of development.

Another possible direction for future work in this context is the development of LLMs custom-trained for brand recognition. It would be quite interesting to see a stripped down version of these models focused on only brand recognition. While much of the performance of these models comes from good generalizability obtained through training on massive datasets in varying contexts, it would be interesting

to see if a specified LLM would outperform a general LLM.

References

[1] Anthropic (2024). The claude 3 model family: Opus, sonnet, haiku. *Anthropic*. https://www.anthropic.com/claude-3.

[2] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:1804.02767.

[3] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). "End-to-End Object Detection with Transformers". In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham. Springer International Publishing. 2005.12872.

[4] Commission, E. (2007). Extended Producer Responsibility for Textiles in France.

[5] Commission, E. (2022). EU Strategy for Sustainable and Circular Textiles.

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

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

[8] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929.

[9] Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. arXiv preprint arXiv:1504.0808.

[10] Herschel, W. (1800). XIV. Experiments On The Refrangibility Of The Invisible Rays Of The Sun. *Philosophical Transactions of the Royal Society of London*, (90):284–292.

[11] Hou, S., Li, J., Min, W., Hou, Q., Zhao, Y., Zheng, Y., and Jiang, S. (2023). Deep Learning for Logo Detection: A Survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 20(3). arXiv preprint arXiv:2210.04399.

[12] Iandola, F. N., Shen, A., Gao, P., and Keutzer, K. (2015). Deeplogo: Hitting logo recognition with the deep neural network hammer. arXiv preprint arXiv:1510.02131.

[13] Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics YOLO.

[14] Joly, A. and Buisson, O. (2009). Logo retrieval with a contrario visual query expansion. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 581–584.

[15] Juanga-Labayen, J. P., Labayen, I. V., and Yuan, Q. (2022). A Review on Textile Recycling Practices and Challenges. *New Research Trends for Textiles*, 2(1):174–188.

[16] Kahraman, Y. and Durmuşoğlu, A. (2023). Deep learning-based fabric defect detection: A review. *Textile Research Journal*, 93(5-6):1485–1503.

[17] Letessier, P., Buisson, O., and Joly, A. (2012). Scalable mining of small visual objects. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 599–608. ACM.

[18] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., and Wei, X. (2022). Yolov6: A single-stage object detection framework for industrial applications. arXiv:2209.02976.

[19] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). "Microsoft COCO: Common Objects in Context". In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

[20] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768.

[21] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham. Springer International Publishing.

[22] MacArthur, E. (2017). Ellen MacArthur Foundation A New Textiles Economy: Redesigning Fashion's Future. *London, UK*.

[23] McClure, W. F. (2003). 204 Years of near Infrared Technology: 1800–2003. *Journal of Near Infrared Spectrosc.*, 11(6):487–518.

[24] Misra, I., Girdhar, R., and Joulin, A. (2021). An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2906–2917.

[25] Nisha (2023). The Adidas Logo: A Look Behind the Stripes. Accessed: 2024-07-03.

[26] of Infrastructure, M. and Management, W. (2023). Infographic: Extended Producer Responsibility for textiles.

[27] OpenAI (2023). Gpt-4: Openai's language model. *OpenAI*. https://www.openai.com/research/gpt-4.

[28] Ouyang, H. (2024). DEYO: DETR with YOLO for End-to-End Object Detection. arXiv preprint arXiv:2402.16370.

[29] Pasquini, C. (2018). Near infrared spectroscopy: A mature analytical technique with new perspectives – A review. *Analytica Chimica Acta*, 1026:8–36.

[30] Rasheed, A., Zafar, B., Rasheed, A., Ali, N., Sajid, M., Dar, S. H., Habib, U., Shehryar, T., and Mahmood, M. T. (2020). Fabric Defect Detection Using Computer Vision Techniques: A Comprehensive Review. *Mathematical Problems in Engineering*, 2020(1):8189403.

[31] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[32] Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[33] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[34] Refibered (2023). Advanced Material Detection via AI-Based Hyperspectral Imaging.

[35] Rekavandi, A. M., Rashidi, S., Boussaid, F., Hoefs, S., Akbas, E., and bennamoun, M. (2023). Transformers in Small Object Detection: A Benchmark and Survey of State-of-the-Art. arXiv preprint arXiv:2309.049023.

[36] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[37] Sandberg, E. and Pal, R. (2024). Exploring Supply Chain Capabilities In Textile-To-Textile Recycling – A European Interview Study. *Cleaner Logistics and Supply Chain*, 11:100152.

[38] Sanghvi, J., Rathod, J., Nemade, S., Panchal, H., and Pavate, A. (2023). Logo Detection Using Machine Learning Algorithm : A Survey. In *2023 International Conference on Communication System, Computing and IT Applications (CSCITA)*, pages 136–141.

[39] Shehzadi, T., Hashmi, K. A., Stricker, D., and Afzal, M. Z. (2023). Object Detection with Transformers: A Review. arXiv preprint arXiv:2306.04670v3.

[40] SpectralEngines (2023). Efficient Textile Recycling With NIR Spectroscopy.

[41] Stichting UPV Textiel (2024). Together circular. Accessed: 2024-06-18.

[42] Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., and Gulati, A. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530.

[43] Tüzkö, A., Herrmann, C., Manger, D., and Beyerer, J. (2017). Open set logo detection and retrieval. arXiv preprint arXiv:1710.10891.

[44] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

[45] Ultralytics (2021). YOLOv5: A state-of-the-art real-time object detection system. https://docs.ultralytics.com. 05/06/2024.

[46] Valvan (2023). Sorting Of Textiles Based On Fiber And Colour.

[47] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[48] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475.

[49] Wang, J., Min, W., Hou, S., Ma, S., Zheng, Y., and Jiang, S. (2022). LogoDet-3K: A Large-scale Image Dataset for Logo Detection. *ACM Trans. Multimedia Comput. Commun. Appl.*, 18(1). arXiv preprint arXiv:2008.05359.

[50] Wang, J., Min, W., Hou, S., Ma, S., Zheng, Y., Wang, H., and Jiang, S. (2020). Logo-2K+: A Large-Scale Logo Dataset for Scalable Logo Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6194–6201.

[51] Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., and Vajda, P. (2020). Visual Transformers: Token-based Image Representation and Processing for Computer Vision. arXiv preprint arXiv:2006.03677.

[52] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

[53] Zhou, J., Yu, L., Ding, Q., and Wang, R. (2019). Textile fiber identification using near-infrared spectroscopy and pattern recognition. *Autex Research Journal*, 19(2):201–209.