Freie Universität Berlin

Bachelor Thesis at The Department of Mathematics and Computer Science

Dahlem Center for Machine Learning and Robotics

Brand Detection on Garments

Aiman Al-Hazmi
Student ID: 5570502
aimaa01@zedat.fu-berlin.de

Advisors:	Prof. Dr. Daniel Göhring, Jan Weimer
Examiner:	Prof. Dr. Daniel Göhring
Second Examiner:	Prof. Dr. Tim Landgraf

Berlin, July 12, 2024

Abstract

Detecting brand logos on garments is an important but time-consuming task for retail and e-commerce businesses. Since the traditional manual methods are slow and prone to errors, which limits the potential value of the products, automated solutions overcome these issues and simplify the process of sorting garments for second-hand businesses. However, their detection accuracy is not yet mature. This thesis uses an advanced automated solution that leverages deep learning techniques. It combines a classification model to identify logos and a zero-shot object detector to locate and extract logo patches from garment images. The experiments carried out have shown potential enhancements in the accuracy and efficiency of logo detection. Having a large, well-labeled logo dataset would dramatically enhance the precision and efficiency.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

July 12, 2024

Aiman Al-Hazmi

Contents

1 Introduction **7**

2 Background **9**
2.1 Machine Learning . . . 9
2.1.1 Supervised Learning . . . 9
2.2 Deep Learning . . . 10
2.2.1 Convolutional Neural Network (CNN) . . . 11
2.2.2 Transfer Learning . . . 12
2.3 Computer Vision . . . 13
2.3.1 Image Classification . . . 13
2.3.2 Object Detection . . . 13
2.3.3 Data Augmentation . . . 14
2.4 Pre-Trained Models . . . 14
2.4.1 Yolo (You Only Look Once) . . . 14
2.4.2 ResNet (Residual Network) . . . 15
2.4.3 Grounding Dino . . . 16

3 Implementation **17**
3.1 Initial Implementation . . . 17
3.1.1 Dataset . . . 17
3.1.2 Detection Model . . . 17
3.2 Final Implementation . . . 20
3.2.1 Detection Model . . . 20
3.2.2 Dataset . . . 20
3.2.3 Data Preprocessing . . . 21
3.2.4 Training . . . 22

4 Experiments and Results **25**
4.1 Experiments . . . 25
4.1.1 Baseline Experiment . . . 25
4.1.2 Intermediate Experiment . . . 26
4.1.3 Final Experiment . . . 28
4.2 Results . . . 30
4.2.1 Evaluation . . . 31
4.2.2 Predictions . . . 33

5 Conclusion and Future Work **35**

A Appendix **39**

1 Introduction

Logo detection is a critical task in computer vision with significant applications in marketing, brand monitoring, and e-commerce. Detecting brand logos on garments is particularly challenging and labor-intensive for retail and e-commerce industries. Traditional manual methods are not only slow but also prone to errors, significantly reducing the efficiency of garment sorting operations, especially in second-hand businesses. As the volume of garments requiring sorting increases, there is a pressing need for automated, reliable solutions.

Several studies have focused on developing robust methods for logo detection and classification, leveraging advancements in deep learning and the availability of large-scale datasets. Among these, the LogoDet-3K [36] and Logo-2K+ [35] datasets are notable contributions. The Logo-2K+ [35] dataset, which is utilized in this thesis, offers over 2,000 brand classes with more than 167,000 images designed specifically for scalable logo classification. Both datasets have significantly advanced the fields of logo detection and classification by providing large-scale, annotated data that support the development of more accurate and efficient models.

This thesis explores an advanced automated approach to detect brand logos on garments using deep learning techniques and aims to enhance detection precision and efficiency.

Within this thesis, a series of experiments are conducted to train a YOLO (You Only Look Once) model [24] for detecting and classifying logos on garments. However, this requires a large-scale dataset of clothing items, which is collected from Sellpy [29], a platform for buying and selling second-hand items. This dataset contains 204.652 images of clothing items and 3.207 unique brand classes. However, the dataset required manual annotation of bounding boxes around each logo, which is time-consuming for large datasets. To avoid such a manual process, an automated method for the dataset annotation is investigated to enable the Yolo model to be trained. The Grounding Dino model [18] is used for the annotation as it has the ability to identify object locations based on a text prompt that contains the words representing the desired objects without requiring additional training. However, this model often mistakenly detects other objects as logos, making its resulting annotations unreliable for direct use as input for YOLO.

To address the challenge, this thesis investigates a refined method that combines a zero-shot object detector with a classification model. This approach utilizes a composed model for enhanced logo detection, leveraging the strengths of both components to achieve more accurate and efficient results. While the detector locates and extracts logo patches from garment images, the classification model identifies these logos. The Grounding Dino model is used as a zero-shot object detector. Since none of the available classification models can be used without being specifically trained on logo datasets, even if some are pretrained on datasets in certain domains, comprehensive experiments are conducted to train and select a suitable model. The pretrained models ResNet [10], VGG16 [30], and EfficientNet07 [31] are considered in these experiments. ResNet50, a version of the ResNet architecture, has demonstrated the best performance with accuracy and an F1 score of 76.20%, along with its computational efficiency, shorter training time, faster inference speed, and lower memory usage.

This composed model, integrating Grounding DINO for detection and ResNet50 for classification, demonstrates significant improvements in precision and efficiency. The experimental results indicate that while the model performs well for certain well-represented brands, it struggles with less-represented and scenarios where logos are not visible or partially obscured on the garments. A large, balanced, and well-labeled dataset is essential for further enhancing the model's performance.

This thesis provides a detailed understanding of the model's development, from initial experiments with the YOLO model to the final implementation using Grounding DINO and ResNet50.

This thesis is structured as follows. Chapter 2 presents the current state of the art as well as the topics covered within this thesis. The implementation work is described in chapter 3 and the experiments and results are discussed in chapter 4. The thesis is then concluded in chapter 5.

2 Background

Distinguishing several relevant terms and concepts from each other is necessary to provide a fundamental understanding of brand detection and classification. For this purpose, the fundamental concepts of Machine Learning, deep learning, and computer vision required for this thesis are described. Thereafter, the pre-trained models utilized in this thesis are introduced and the metrics used to evaluate machine learning models.

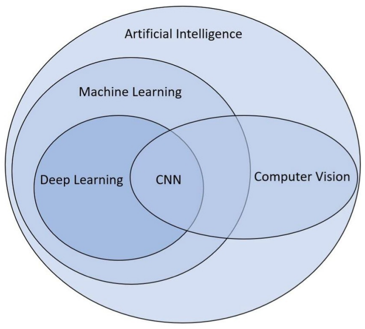


Figure 1: Euler diagram of artificial intelligence [3]
CNN: Convolutional Neural Network

2.1 Machine Learning

Machine learning is a subfield of artificial intelligence, as illustrated in Fig.1. It focuses on developing algorithms and statistical models that enable computer systems to perform tasks without explicit instructions [15]. Instead, these systems rely on patterns and heuristics. By processing large amounts of data, machine learning algorithms recognize patterns, improving their ability to predict outcomes from given input datasets. Machine Learning is instrumental in solving problems in vision, speech recognition, and robotics.

Defintion of Machine Learning Model: A model is a mathematical representation or the output of a machine learning algorithm that can find and recognize patterns and make predictions or decisions based on new input data[5] [19]. It consists of parameters and structures adjusted during training to minimize error and improve accuracy.

Machine learning can be categorized into several types based on the given problem and the available data. Figure 2 illustrates the common types such as supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. This Thesis focuses on supervised learning

2.1.1 Supervised Learning

This type of machine learning differs from other types in that the data fed into the model must be labeled. A labeled dataset contains many data points, each consisting of two parts: a feature vector (or instance) and a label indicating the desired output. This data is divided into training and testing datasets. The model is trained on the training dataset to find a mapping from inputs to outputs to generalize well to unseen data, ensuring the model performs accurately on new, unobserved inputs. The testing dataset is fed into the model to evaluate its performance and validate its predictions, as the model has not seen this data before.

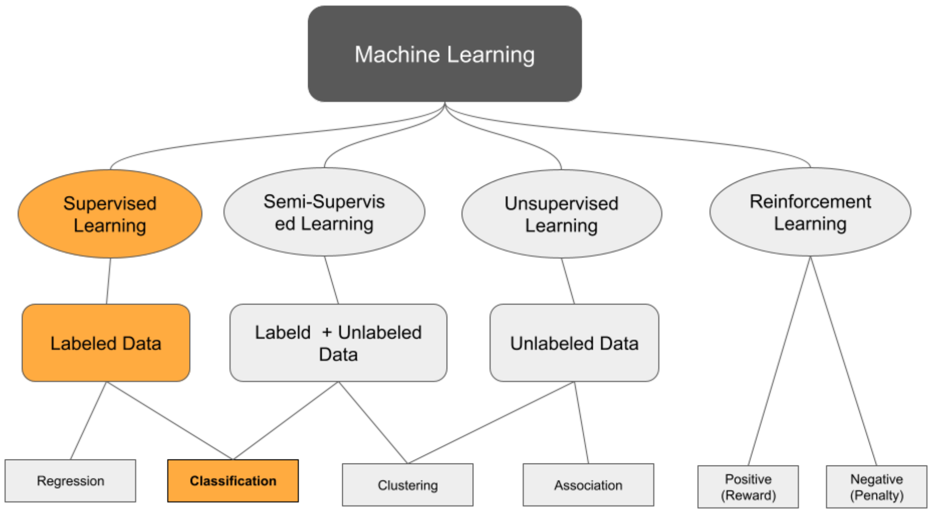


Figure 2: Main categories of machine learning [14][28]

During the model's training and validation, a loss function calculates and measures the error margin between its predictions and actual labels. Supervised machine learning is further classified into two main categories: Classification and Regression [13][14], as shown in Fig.2. The classification is discussed later in section 2.3.1.

2.2 Deep Learning

Deep learning is a subfield of machine learning where algorithms are developed to learn in a way that is inspired by neurons in the human brain, thus imitating the way the human mind works[2]. The authors in [2] believe that deep learning performance has become superior to human performance in computer vision tasks such as classification (Fig. 3).

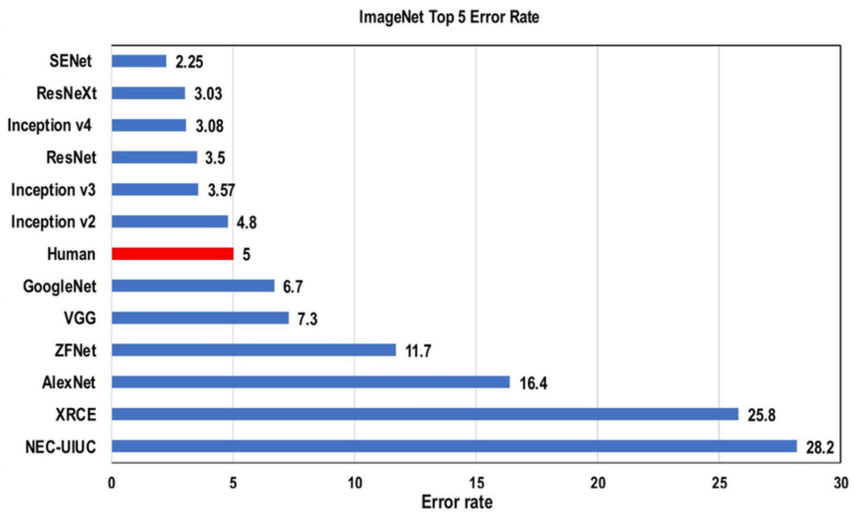


Figure 3: performance of deep learning models compared to human [2]

Deep learning models are based on neural network architectures (Fig. 4), consisting of interconnected nodes (neurons) in a layered structure that connects the inputs to the desired outputs. The neurons between a neural network's input and output layers are called hidden layers. Deep learning allows computational models to learn data at multiple levels of abstraction. It finds complicated patterns in large amounts of data using a backpropagation technique. This helps machines change the settings that are used to create each layer's representation from the previous layer's representation[17].

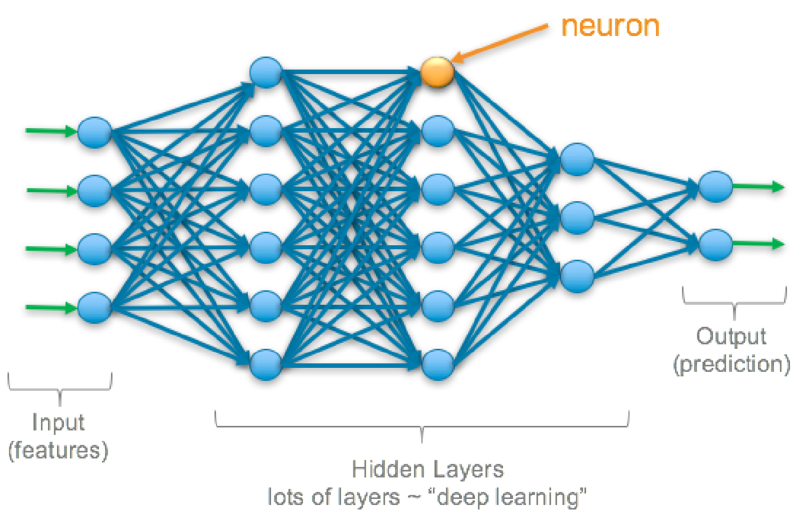


Figure 4: Multi-layer Perceptron (MLP)[26]

While there are various types of neural network architectures within deep learning, some are specialized for particular data types and tasks. One such specialized architecture is the Convolutional Neural Network (CNN), which is highly effective for processing and recognizing patterns in grid-like data such as images.

2.2.1 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is the most famous and commonly employed algorithm in Deep Learning[2]. CNNs have become essential in computer vision tasks such as image classification, object detection and segmentation [2] [17]. An example of CNN architecture for image classification is illustrated in Fig 5.

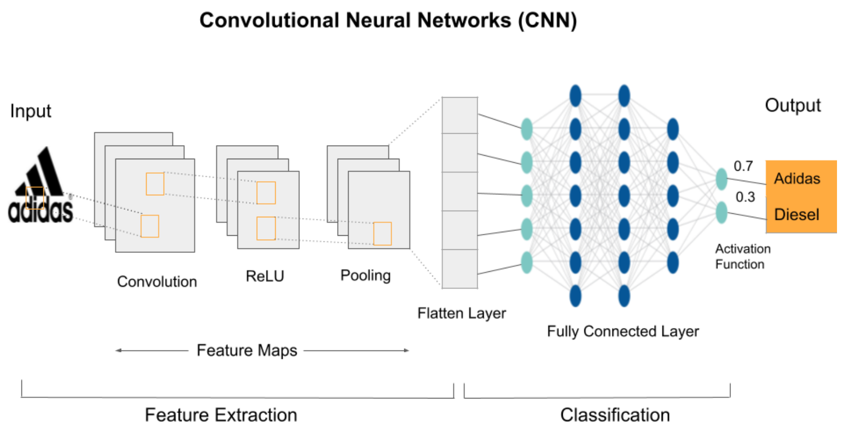


Figure 5: An example of CNN architecture for image classification

The layers of a CNN architecture can be classified into the following:

1. **Convolutional Layer:** This layer is the most significant component, in which a collection of convolutional filters (kernels) is applied to the input image to extract features such as edges, textures, and patterns[2][27]. The output of this layer is known as the "feature map" or "convolved features."

2. **Activation Layer:** In this layer, a non-linear activation function, such as the ReLU function, is applied to the output of the Convolutional Layer. This function helps to introduce non-linearity into the model, allowing it to learn more complex representations of the input data [2].

3. **Pooling Layer:** It performs a sub-sampling operation to reduce the spatial dimensions of the convolved features, which helps to decrease the computational complexity of the model, reduce the number of parameters[2]. The most common aggregation functions that can be applied are:

   (a) Max pooling, which is the maximum value of the feature map.

   (b) Sum pooling corresponds to the sum of all the values of the feature map.

   (c) Average pooling is the average of all the values.

4. **Flatten Layer:** This layer converts the feature map, which is essentially a multi-dimensional array containing pixel values, into a one-dimensional array for the fully connected layer.

5. **Fully Connected Layer (FC):** This is typically the final layer of the convolutional neural network after a series of convolutional and pooling layers. It connects all the neurons in the previous layer to those in the next layer. The fully connected layer is responsible for combining the features learned by the convolutional and pooling layers to make a prediction [2][27].

CNNs have shown that they can perform many different computer vision tasks. However, they also have limitations, such as a high computational cost, overfitting, and a lack of interpretability [2]. A large amount of data is also needed to obtain a well-trained deep-learning model and increase performance [2][28]. Many techniques can be applied to achieve a well-behaved performance model, such as transfer learning, data augmentation, regularization.

2.2.2 Transfer Learning

This technique involves transferring the knowledge, including the feature extractors and the weights (and biases), from a pre-trained model designed for a specific task to a new model. This transfer aims to enhance the new model's performance on a different but similar task[2][6]. Through a process known as fine-tuning, pre-trained models can be adjusted to perform new tasks with relatively little data. In this process, the fully connected layer is adapted and trained. It is usually needed to fine-tune specific pre-trained layers before the fully connected layer by unfreezing them to achieve better results. Figure 6 illustrates the concept of transfer learning using Convolutional Neural Network.

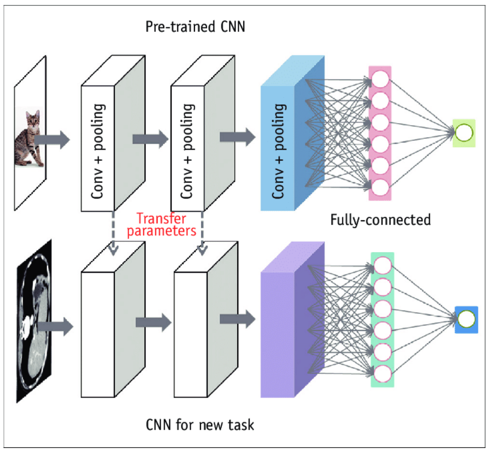


Figure 6: Transfer learning [6]

2.3 Computer Vision

The process of extracting useful information from digital images, videos, and other visual inputs using machine learning and neural networks is known as "Computer Vision" [12]. It falls under the field of artificial intelligence, as shown in Fig. 1. This section covers the two major computer vision tasks, Image Classification and Object Detection, and the concept of data augmentation.

2.3.1 Image Classification

Image classification is an essential task in computer vision. In this task, the entire image is classified into a particular class, representing all images with similar attributes and characteristics. This task involves training a model to identify and categorize images based on their features.

In the following, the common classification problems are presented.

1. **Binary classification:** In this Classification problem, a trained model classifies the input into one of two possible classes. For example, classifying emails as "spam" or "not spam."

2. **Multiclass Classification:** This involves classifying the input into one of two or more classes. For example, a brand logo can be assigned to one of many companies.

3. **Multi-label Classification:** In this classification problem, each input can be assigned to multiple labels. For example, A hoodie can be classified under "casual wear", "outerwear", and "unisex clothing."

2.3.2 Object Detection

Object detection is a computer vision technique that involves localizing and classifying objects within an image. The process starts by drawing a rectangular box around each detected object. This is followed by classifying the object and assigning it a label (class).

Open-Set Object Detection (OSOD): This kind of detection has enhanced object detection by enabling the identification of instances of unknown classes. However, it still labels all recognized objects of an unknown class as "unknown" [38].

Zero-Shot Object Detection: Zero-shot detection (ZSD) is the process of identifying and locating objects in images, even if the objects were not part of the training dataset. This involves utilizing knowledge from known classes to detect and localize objects from unknown classes in test images [23].

2.3.3 Data Augmentation

Data augmentation is commonly used in machine learning, especially for deep learning models. It is the process of modifying and augmenting a dataset to increase its size and diversity. Various transformations can be applied to the existing dataset, such as flipping, cropping, rotating, and color jittering, to overcome small datasets' limitations and improve models' performance and generalization [9].

2.4 Pre-Trained Models

A pre-trained model is trained on a large dataset to solve a specific problem. Various pre-trained models exist, such as Yolo [24], Resnet [10], EfficientNet [31], VGG [30], and Grounding Dino [18]. This section will discuss the Yolo model as a detection model and the ResNet model as a classification model. We will also provide a comparative summary of ResNet, VGG, and EfficientNet as classification models. Finally, we will introduce Grounding Dino and explain how it works.

2.4.1 Yolo (You Only Look Once)

Yolo is a popular real-time object detection model that is implemented as a convolutional neural network (see Fig. 7). The network's initial convolutional layers extract features from the image, while the fully connected layers predict the output probabilities and coordinates [24].

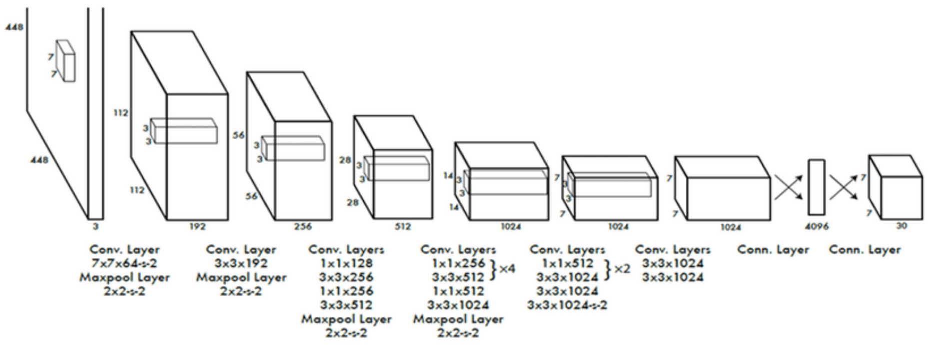


Figure 7: YOLOv1 architecture [24]

Several versions of this model have been developed to leverage its performance through architectural improvements, enhanced training techniques, and data augmentations(Fig. 8) [32].

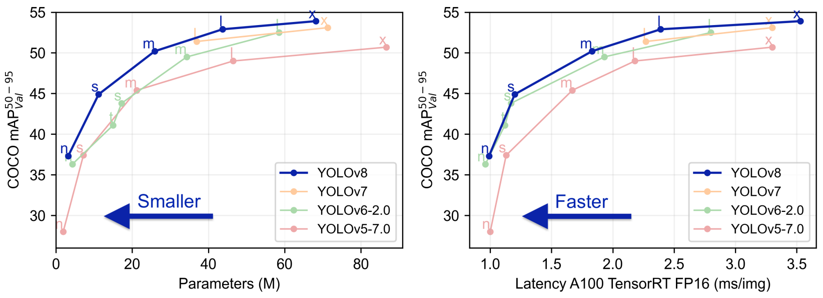


Figure 8: YOLO family performance [34]

2.4.2 ResNet (Residual Network)

ResNet is a model based on CNN architecture and was introduced by Kaiming He [10]. It was designed to address the vanishing gradient problem encountered during the training of deep neural networks. By incorporating residual connections, it enables information to bypass certain layers and flow directly from one layer to another (see Fig. 9). The residual learning approach not only simplifies the training of even deeper networks but also facilitates the learning of more complex features [10][1].

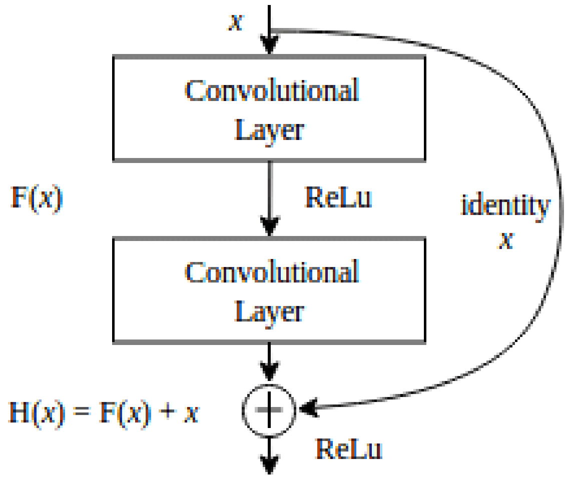


Figure 9: Residual learning: a building block [10]

There are various versions of ResNets, such as ResNet50 and ResNet101.ResNet50 is one of the most famous ResNet architectures. [1].

Table 1 provides a comparative summary of three convolutional neural network models: ResNet, VGG, and EfficientNet, trained and evaluated on the ImageNet dataset. The architecture of these models differs from each other, with the suffix number indicating the number of layers used in the network architecture.

Model	ResNet	VGG	EfficientNet
Year	2015	2014	2019
Versions	ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152	VGG-11, VGG-13, VGG-16, VGG-19	EfficientNet-B0 to EfficientNet-B7
Top-1 Accuracy	76.0% (ResNet-50)[4][31]	71.5% (VGG-16)[4]	84.3% (EfficientNet-B7)[31]
Top-5 Accuracy	93.0% (ResNet-50)[4][31]	89.9% (VGG-16)[4]	97.0% (EfficientNet-B7)[31]
Parameters	25.5M (ResNet-50)[4][31]	138M (VGG-16)[4]	66M (EfficientNet-B7)[31]
FLOPs	4.1B (ResNet-50)[4][31]	19.6B (VGG-16)[4]	37B (EfficientNet-B7)[31]

Table 1: Comparative summary of ResNet, VGG, and EfficientNet on ImageNet dataset

2.4.3 Grounding Dino

The Grounding Dino model is an advanced zero-shot object detector built on the DINO architecture and integrates multi-level text information through grounded pre-training [18]. Its architecture is shown in Fig. 28 in Appendix A.
It can also process images and textual data together. It takes a text prompt separated by a dot as input to detect the described objects within the image, which is also given as input [18]. Figure 10 illustrates an example of how the grounding dino model works. The Input includes an image representing a garment item with a logo and a text prompt containing three words representing the objects (logo, arm, hood) separated by a dot. The output of the model is a list of detections, each containing the coordinates of the prompted object.

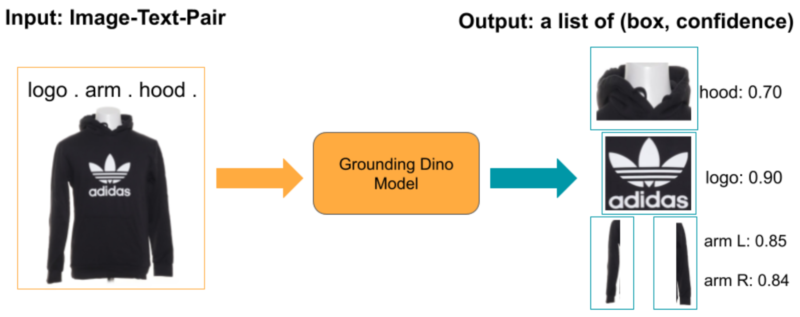


Figure 10: Grounding Dino

3 Implementation

This chapter presents the implementation of the solution presented in this thesis to detect brand logos on garments automatically. The implementation is based on the utilization of deep learning models. The work in this thesis was performed in an iterative and incremental manner to find a suitable model and a comprehensive dataset for training and evaluating the model. Therefore, the implementation went through two phases. The initial phase had limitations in terms of reliability and accuracy, which were the focus of the next, final implementation phase. This chapter provides an overview of the initial implementation phase, followed by a description of the final implementation phase.

3.1 Initial Implementation

3.1.1 Dataset

To train and evaluate a model, a representative, large-scale dataset is required. The dataset used in this implementation phase was collected from Sellpy [29], a platform for buying and selling second-hand items. The gathered dataset contains 204.652 images of clothing items, each labeled with a brand name, as shown in Fig. 11.

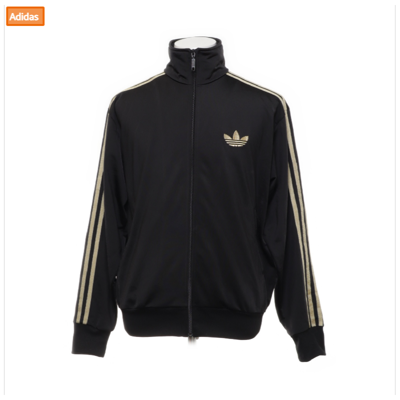


Figure 11: Garment item from Sellpy dataset

In total, this dataset includes 13.207 unique brand classes. As shown in Figure 12, the dataset is imbalanced, and more than 5.350 images are assigned to the ’None’ class, i.e., the brand name is unknown. All brands with less than 1000 images are grouped in the ’Others’ Category for clearer visualization.

3.1.2 Detection Model

This section presents two iterations for detecting logos on garments and discusses their limitations.

First Iteration

Various object detection models exist. The suitable model can be selected depending on the detection task and other factors such as speed, accuracy, and complexity. Since speed and accuracy are important, it was decided to select Yolov8 for the logo detection task. The Yolov8 model is popular, and it promises a good mix of training, inference speed, and high accuracy, as seen in Fig 8, making it suitable for most practical applications, especially for detecting objects in real-time.

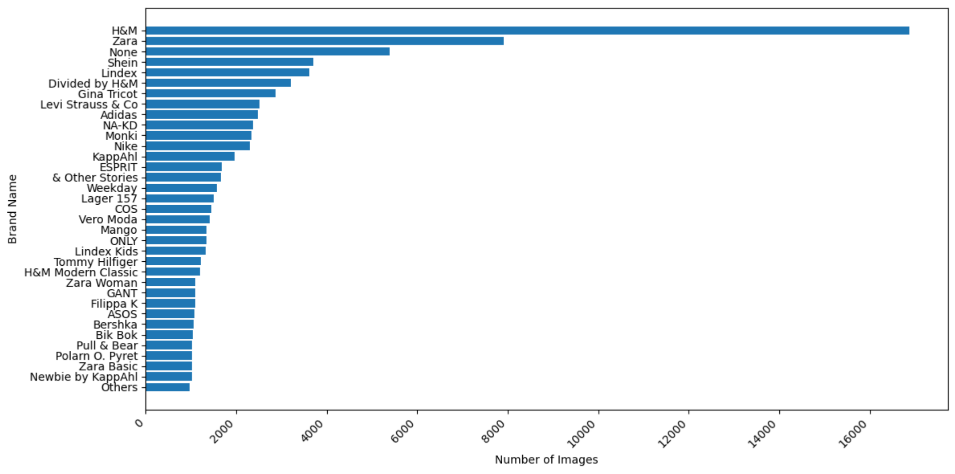


Figure 12: Number of images for each brand name or category

Limitations of the First Iteration
To train a detection model like Yolov8, the dataset must be divided into two or three parts: a training dataset, a validation dataset, and a test dataset. However, the dataset must be annotated with rectangular boxes around each logo in the image. The dataset collected from Sellpy was not annotated, and the annotation process can be done manually using tools such as labelImg. Still, this process is quite time-consuming, especially for a large dataset, which highlights the need for an automated annotation method or a way to automatically locate the logos.

Second Iteration
An automated method for detecting logo locations is investigated to check whether it overcomes the limitations faced in the first iteration, and thus enabling the Yolov8 model to be trained. Therefore, the Grounding Dino model is utilized as an annotating tool since it can identify the locations of objects in the image and assign them labels based on the text descriptions (see Fig. 13).

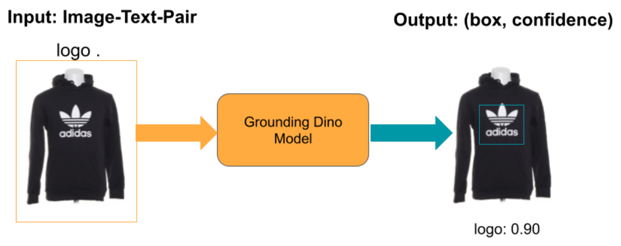


Figure 13: Logo detection by Grounding Dino model

The Grounding Dino model was fed with the following prompt and 204.510 images from the Sellpy dataset.

```
text_prompt = "logo . belt . button . hanger . hood . arm .
               mannequin . neck . pocket . zipper"
```

The model interprets these words to identify and locate these objects within the images. Thus, It generated 40.378 logo detections spread over 37.499 images. Detections involve successfully identifying and locating a logo within an image, each detection includes a bounding box around the detected logo in the image and an accuracy score indicating the likelihood of the bounding box containing a logo. Figure 14 shows the imbalanced image distribution across all brands.

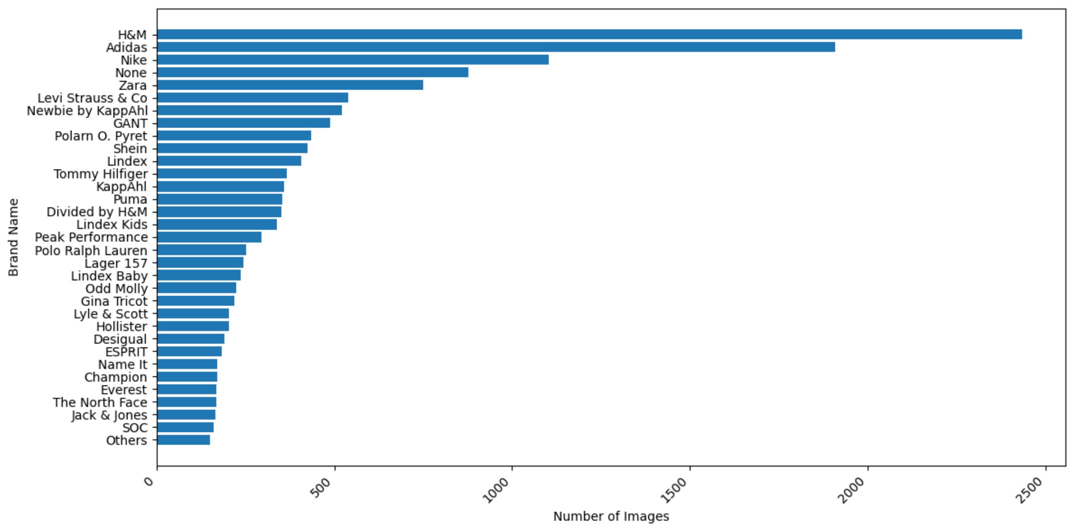


Figure 14: Number of images for each brand name or category

Limitations of the Second Iteration

The detections of the Grounding Dino model were unreliable. Thus, Its output couldn't always be trusted. In some cases, other objects in the image were detected as a logo, but this was false, as illustrated in Fig 15.

(a) Detection error 1

(b) Detection error 2

Figure 15: Detection errors of Grounding Dino model

As shown in Fig 15b, the confidence score of that detected object to be a logo is 59%, which is a high value. Therefore, it is impossible to define a threshold that effectively distinguishes between true logos and false detections without compromising the model's overall performance. It is furthermore challenging to identify the brand of a garment when there is no visible logo on the outside, as seen in Fig 15b. This implies that while the model is good at finding the locations of the logos, it may also incorrectly annotate non-logo objects as logos, highlighting the need for further refinement or additional criteria to improve detection precision.

3.2 Final Implementation

The previous limitations led to a final implementation phase, which is discussed in this section.

3.2.1 Detection Model

Since the detection process usually involves two tasks, object localization and classification, these tasks are implemented in the final implementation phase using two different models. The Grounding Dino model is used as a detector to find the locations of the logos on garments, i.e. bounding boxes around the logos. For the classification task, a classification model is used to classify the patches extracted from the images. However, a suitable robust classification model must be selected and trained on a new custom dataset consisting of various brand logos. ResNet is a commonly used model for classification tasks, which was initially chosen. During the implementation, other models were tested and compared. The results are discussed in chapter 4. Figure 16 illustrates a detection pipeline of the composed model for detecting brands on garments.

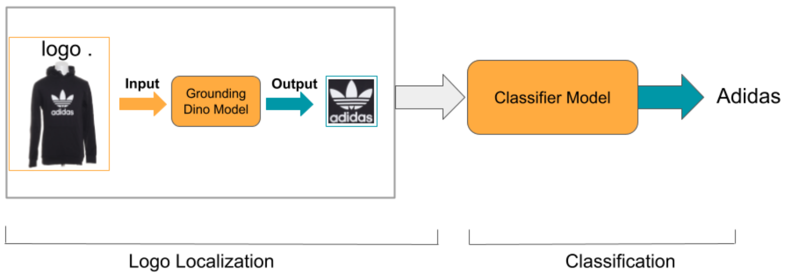


Figure 16: Detection model composed of Grounding Dino and classification model

3.2.2 Dataset

In this phase, the Sellpy dataset is used to test the composed model, and other publicly available logo datasets are used for training and validating the classification model. The main one is the "Logo-2K+" dataset[35], as described in the paper [35], includes 167,140 images with 10 root categories and over 2,000 subcategories (Fig. 17).

The clothing category is selected for the classification task. It contains 286 logos and 20.413 images. Each image is labeled with its respective category (brand name) and has a resolution of 256×256 pixels. Although this dataset is large, it does not include a few brands that can be found in the Sellpy dataset, with a large number of images, such as Nike and Tommy Hilfiger. To ensure correct model testing and increase the number of test images and variety in the test dataset, it is necessary to include as many other brands as possible in the training and validating datasets. Therefore, 10 missing brand classes collected from the dataset "Logos in the Wild"[33] are combined with the clothing category from the "Logo-2K+" dataset. Additionally, a 'None' class is created using 877 image patches from the Sellpy dataset to train the model to classify such patches as 'None'. This modified dataset for training and validating the classification model is imbalanced and contains 297 classes and 21.214 images.

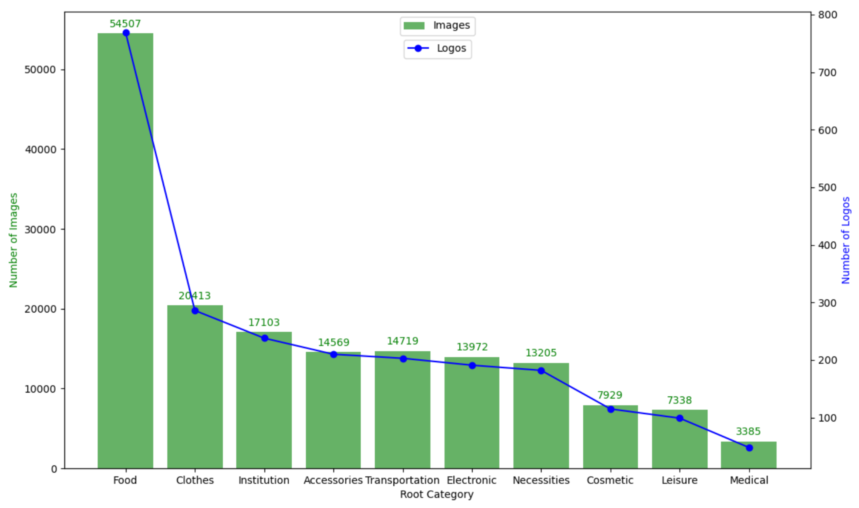


Figure 17: Logo-2K+ dataset

The test dataset for evaluating the composed model is created by applying the intersection operation to the brand classes between the Sellpy and the newly combined datasets. This process identifies the common brand classes that exist in both datasets, ensuring a comprehensive evaluation of the detection model. However, Zara and H&M classes, which contained 3186 images, are excluded because too many of those images do not contain any logos or brand names on the clothing. Thus, the dataset is imbalanced and includes 5109 images with 67 classes.

3.2.3 Data Preprocessing

It is essential to preprocess the dataset before feeding and training the classification model. This helps in improving the model performance, reducing computational complexity, extracting relevant information and patterns from the dataset [8]. Below are the initial preprocessing steps applied to the datasets:

1. **Organizing the Dataset:** The dataset used for training and validating the classification model was organized into a folder structure suitable for the classification and training process, as seen in Fig. 18. Folders are named after brand names. Each Folder represents a class and contains logos of the associated brand.

2. **Label encoding:** This process involves assigning each class to a unique numerical value. For example, the brand name 'Adidas' is assigned to the number 256.

3. **Splitting the dataset:** The combined dataset is randomly divided into training and validation datasets, with 80% given for training and 20% for validation.

4. **Data Transformation:** The process of transforming the data from one format to another. Thus, each image from the dataset is resized to a uniform resolution of 224x224 pixels, which is required for CNN Models like ResNet. Subsequently, the input image is converted from PIL format (or NumPy Arrays) to PyTorch tensors, which is the required input type for PyTorch models. The converted tensors are then normalized to have a uniform scale.

```
Dataset
├─ class1
│   ├─ img1.jpg
│   ├─ img2.jpg
│   └─ ...
└─ class2
    ├─ img1.jpg
    ├─ img2.jpg
    └─ ...
```

Figure 18: Dataset folder structure

3.2.4 Training

As aforementioned in section 3.2.1 in the final implementation of the detection model, a suitable classification model for the classification task must be selected and trained. Fig. 19 shows a PyTorch workflow for model selection followed in this thesis. Beginning with data preparation, which is done earlier and described in section 3.2.3, followed by selecting a model through evaluation and iterative improvement steps up to the choice of a suitable model.

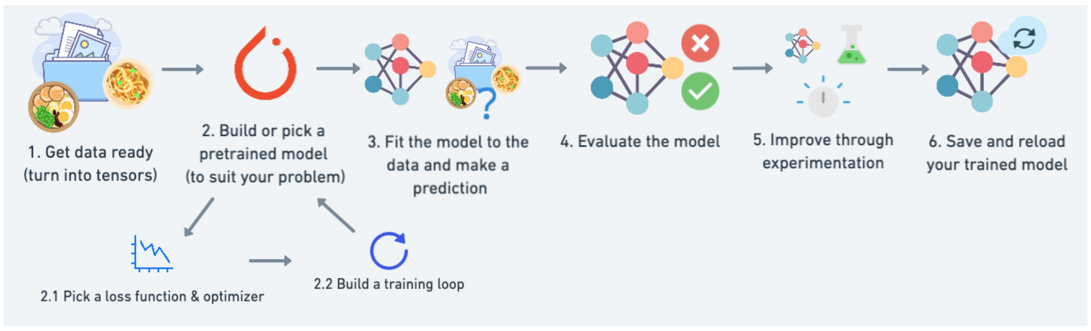

1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2 Build a training loop

Figure 19: A PyTorch workflow [22]

In this thesis, different models are loaded from the PyTorch framework [20], which have been trained on the ImageNet dataset. The transfer learning technique is utilized in this thesis to adapt these models by modifying the fully connected layer. In the transfer process, the input features are maintained without modifications, whereas the output features are adapted to match the number of classes in the previously discussed logo dataset for classification. Thereafter, these models are retrained on that dataset. After loading and adapting the models, a loss function and an optimizer are defined. This step is fundamental to complete the training setup of a deep learning model to improve the model's performance.

Loss Function

As mentioned earlier, the loss function is important for training deep learning models, as it measures the difference between the predicted and actual outputs for each sample fed into the model. The objective of the loss function is to minimize this difference. However, the choice of the suitable loss function depends on the type of classification problem and the nature of the dataset, particularly whether the dataset is balanced or imbalanced. Since this thesis deals with multiclass classification problem and imbalanced dataset, the suitable loss function, which is commonly used, is the 'Crossentropy Loss' [7] [37].

Optimizer
After defining the loss function, an optimizer is used to update the model's parameters to minimize the loss. This optimizer can be adjusted with different settings or hyperparameters such as learning rate, momentum, decay rate. [21]. The most commonly used optimizers are Stochastic Gradient Descent (SGD) [25] and Adaptive Moment Estimation (Adam) [16], which are used and tested in this thesis.

Evaluation Metrics
The evaluation metric is a measurement used to measure the performance of a model. Therefore, selecting a suitable evaluation metric for obtaining the best classification model is essential, and it depends on the type of classification and the specific objectives of the model [11]. For example, accuracy is suitable for balanced datasets, while precision, recall, and F1-score are better for imbalanced datasets [11]. The metrics used in this thesis for the evaluation described in the following:

1. **Accuracy:** is the ratio of correct predictions over the total number of predictions. Basically, it measures how many logos are correctly classified into their correct classes out of all logos tested. For example, if a model is tested on 100 logos and correctly classifies 90 of them into their correct classes, the accuracy would be 90%.

2. **Precision:** is the ratio of the number of correctly identified logos of a particular class (true positives) to the total number of logos the model predicted to be of that class. For example, if the model predicts 100 logos as 'Nike' and 80 of those are actually 'Nike' logos, the precision for the 'Nike' class would be 80%.

3. **Recall:** is the ratio of the number of correctly identified logos of a particular class (true positives) to the total number of actual logos of that class (sum of true positives and false negatives). For example, if there are 100 actual 'Adidas' logos and the model correctly identifies 70 of them, the recall for the 'Adidas' class would be 70%.

4. **F1 Score:** This metric combines precision and recall by taking their harmonic mean. It is particularly useful for imbalanced datasets where a balance between precision and recall is desired. For example, if a class has a precision of 80% and a recall of 70%, the F1 Score would be:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.80 \cdot 0.70}{0.80 + 0.70} = 2 \cdot \frac{0.56}{1.50} = \frac{1.12}{1.50} \approx 0.746 \text{ or } 74.6\%$$

5. **Macro Average:** Average metrics (like precision, recall, F1 score) equally across all classes, treating each class equally important regardless of size. For example, if the model has three classes with precision scores of 99%, 33%, and 50%, the macro average precision would be:

$$\text{Macro Average Precision} = \frac{\text{Precision}_1 + \text{Precision}_2 + \text{Precision}_3}{3}$$

$$= \frac{0.99 + 0.33 + 0.50}{3} = \frac{1.82}{3} = 60.66\%$$

6. **Weighted Average:** Average metrics (like precision, recall, F1 score) weighted by the number of instances in each class, accounting for class imbalance. For example, if the 'Adidas' class has 1400 instances, the 'Nike' class has 200 instances and the 'Zara' class has 100 instances, with respective precision scores of 99%, 50%, and 33%, the weighted average precision would be:

$$\text{Weighted Average Precision} = \frac{n_1 \cdot \text{Precision}_1 + n_2 \cdot \text{Precision}_2 + n_3 \cdot \text{Precision}_3}{n_1 + n_2 + n_3}$$

$$= \frac{1400 \cdot 0.99 + 200 \cdot 0.50 + 100 \cdot 0.33}{1400 + 200 + 100} = \frac{1519}{1700} \approx 0.894 \text{ or } 89.4\%$$

4 Experiments and Results

This chapter discusses only three of many experiments conducted to illustrate how the final suitable model for the classification task is found and selected. The results of the final implementation phase are described in section 4.2.

The first experiment involved training ResNet50, ResNet101, EfficientNet-B7, and VGG16 models without any modifications. It is conducted to measure their performance under standard conditions, providing a baseline for later comparisons when modifications are introduced. Further experiments are conducted to refine the training process and enhance model performance by implementing various adjustments. These include modifications to the model architecture, fine-tuning additional layers, changing the learning rate, and utilizing techniques such as data augmentation and weight decay. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU with 24 GB of memory, ensuring high performance and efficient resource utilization. A seed is set to enhance the reproducibility of the results and control the random values generated during the training process.

4.1 Experiments

4.1.1 Baseline Experiment

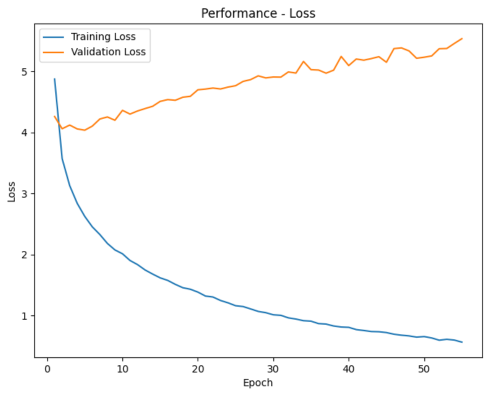
The models are loaded and the last layer, the fully connected layer, is modified with the same input features. However, the output features are set to the number of classes of the custom dataset, which is 297. The models are then trained with a 1e-3 learning rate, 100 epochs, and a batch size of 32. 'Adam' is used as an optimizer, and 'Crossentropy Loss' is used as a loss function. Figure 20 represents the loss curves of the involved models during the training and validation phases.

While ResNet50, ResNet101, and EfficientNetB7 demonstrate strong convergence on the training data, i.e. they learned well on the training data and reached their best performance, they suffer from overfitting, as indicated by the increasing validation loss after initial epochs. Conversely, the training loss of the VGG16 model decreases. Still, it is not as smooth as the other models. The loss curve of the VGG16 model exhibits erratic behavior with frequent fluctuations, indicating instability in learning. As seen in Table 2, EfficientNetB7 shows the highest training accuracy and weighted F1 scores, while ResNet101 and ResNet50 perform best on validation accuracy. VGG16 significantly lags in both training and validation metrics.

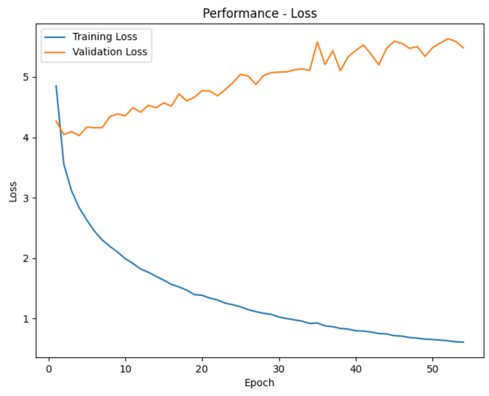
Although the experiment has showen an initial improvement, it was unsuccessful due to overfitting, instability, and lower validation metrics, indicating extreme model complexity or suboptimal training conditions. Thus, techniques such as fine-tuning additional layers, early stopping, data augmentation, or further hyperparameter tuning are necessary to enhance the models' generalization and performance on the custom dataset with 297 classes.

Metrics	**ResNet50**	**ResNet101**	**VGG16**	**EfficientNetB7**
Training Accuracy	82.65%	83.30%	17.78%	88.94%
Validation Accuracy	32.43%	33.09%	13.79%	31.89%
Training Weighted F1 Score	82.70%	83.32%	21.1%	88.95%
Validation Weighted F1 Score	32.50%	32.95%	16.19%	32.02%

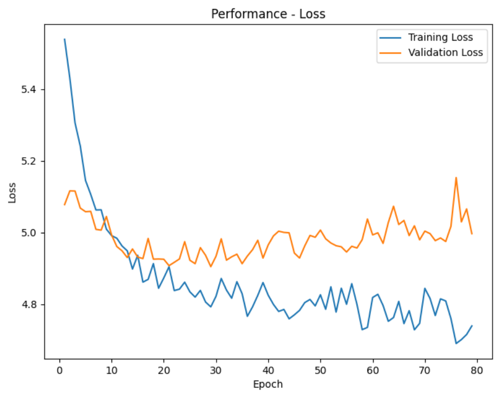
Table 2: Comparison of models on various evaluation metrics



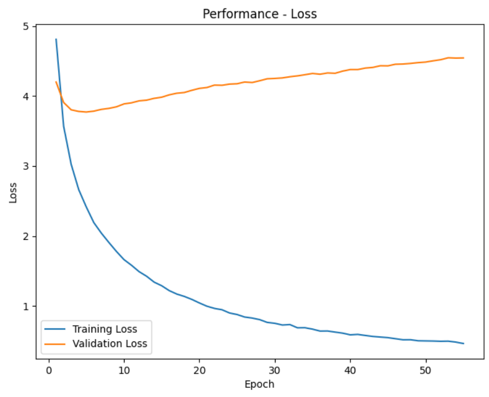
(a) ResNet50



(b) ResNet101



(c) VGG16



(d) EfficientNetB7

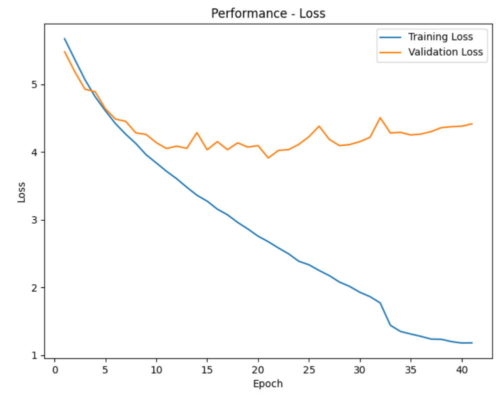
Figure 20: The loss of different models during training and validation phases

4.1.2 Intermediate Experiment

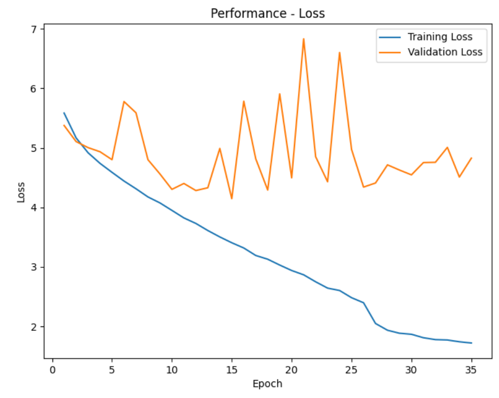
This experiment is done to overcome the limitations faced in the baseline experiment and achieve better performance. Thus, the last two layers of the models are fine-tuned, i.e. re-trained on the custom dataset. This allows the models to adapt specifically to the custom dataset with 297 classes, improving their ability to capture the unique characteristics of the new data without losing the benefits of transfer learning. In the baseline experiment, the validation loss increased and stopped to improve. To overcome this issue, it is necessary to monitor the validation loss and use techniques to prevent overfitting during the training process. Therefore, a learning rate scheduler, early stopping, and data augmentation techniques such as resizing crops, flipping horizontally, rotating, and applying Gaussian blur are used in this experiment. The 'ReduceLROnPlateau' scheduler is used, which reduces the learning rate when the validation loss stops improving. The early stopping technique is triggered if the model performance does not improve over a specified number of epochs (Patience) to avoid unnecessary overfitting. Finally, the models are trained with the same parameters from the baseline experiment and the new techniques discussed above, as shown in Table 3.

Input Features	Output Features	Learning Rate	Epochs	Batch Size	Optimizer	Loss Function	Scheduler	Early Stopping (Patience)	Data Augmentation
2048	297	1e-3	100	32	Adam	Cross Entropy	ReduceLROnPlateau	20	Yes

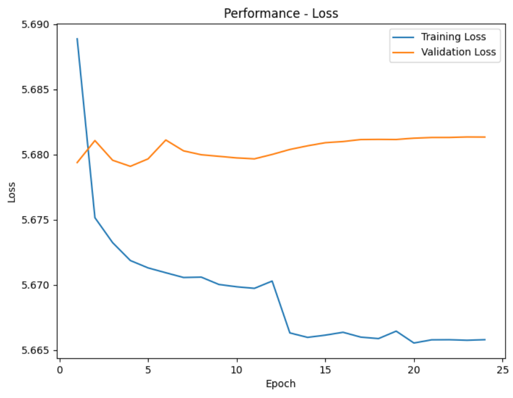
Table 3: Training configurations



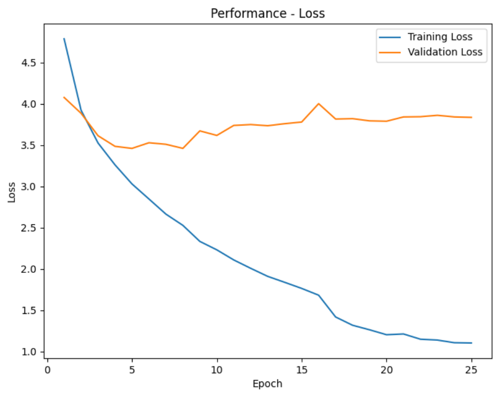
(a) ResNet50



(b) ResNet101



(c) VGG16



(d) EfficientNetB7

Figure 21: The loss of different models during training and validation phases

Although this experiment has made some progress in stabilizing the training process and improving metrics compared to the baseline experiment, the changes are still not drastic, as seen in the loss curves Fig. 21. Due to the early stopping technique, all models stopped training earlier, which helped to prevent overfitting. Table 4 demonstrates that all models, particularly EfficientNetB7, have shown improvements due to the implemented techniques, with decreased training metrics and enhanced validation metrics. However, the VGG16 model performed worse compared to the baseline experiment.

Metrics	**ResNet50**	**ResNet101**	**VGG16**	**EfficientNetB7**
Training Accuracy	72.30%	58.77%	0.068%	73.50%
Validation Accuracy	35.73%	29.04%	0.78%	42.68%
Training Weighted F1 Score	72.78%	58.92%	0.007%	73.82%
Validation Weighted F1 Score	36.51%	29.76%	0.01%	42.60%

Table 4: Comparison of models on various evaluation metrics

In summary, while this experiment has shown some improvements, particularly for EfficientNetB7, it still can not overcome the fundamental challenges of overfitting and poor generalization on the custom dataset. Further refinement and additional techniques are necessary to achieve more significant improvements.

4.1.3 Final Experiment

After extensive experimentation with various models and training configurations to enhance performance and select the best model for the classification task, the decision is made to proceed with the ResNet50 model for the final model selection. Despite EfficientNetB7 showing promising results in the intermediate experiment, both ResNet50 and ResNet101 performed very well across other experiments and achieved similar validation metrics, with the best case showing a difference of less than 2%. Thus, ResNet50 is chosen due to its computational efficiency, shorter training time, faster inference speed, and lower memory usage compared to ResNet101.

In this experiment, the ResNet50 model is trained with three different configurations illustrated in Table 5. The classification layer of the model is modified with the following: three linear layers to enhance pattern learning, between them a BatchNorm layer to stabilize and accelerate training, and finally a dropout layer. Label smoothing is used as a regularization technique with a value of 0.01. The label smoothing and the dropout layer offer several benefits, such as preventing the model from becoming overconfident in its predictions, improving generalization, and stabilizing training. This helps to achieve better performance on new unseen data. Furthermore, two different Ooptimizers, Adam and AdamW, are employed to investigate their impact on the model's performance. A custom scheduler, 'CustomLRScheduler,' is utilized in the second and third configurations, and it is specifically implemented to control the learning rate of the optimizer during the training process. It starts with a fixed learning rate (Base LR) for a defined number of warmup epochs and then linearly decreases it to a specified final learning rate (Final LR) by the end of the training process, as seen in Fig. 22. These values are set based on experimental observations, where the models stopped learning after a specific number of epochs.

Parameter	**Optimizer Configurations**		
	Configuration 1 (Adam)	**Configuration 2 (Adam)**	**Configuration 3 (AdamW)**
Epochs	300		
Batch Size	32		
Learning Rate	0.001		
Weight Decay	0		
Loss Function	CrossEntropyLoss		
Label Smoothing	0.1		
Early Stopping Patience	20		
Scheduler	null	CustomLRScheduler	CustomLRScheduler
Warmup Epochs	null	150	150
Base LR	null	0.001	0.001
Final LR	null	1e-05	1e-05

Table 5: Experimental setup

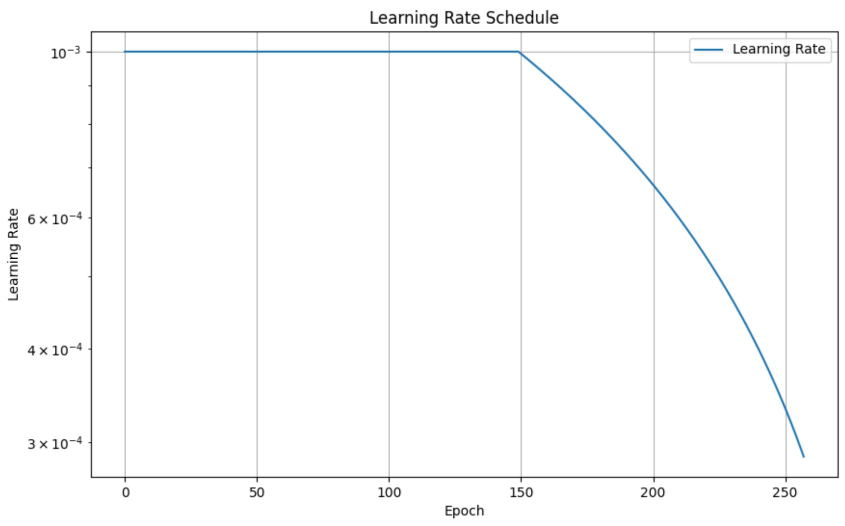
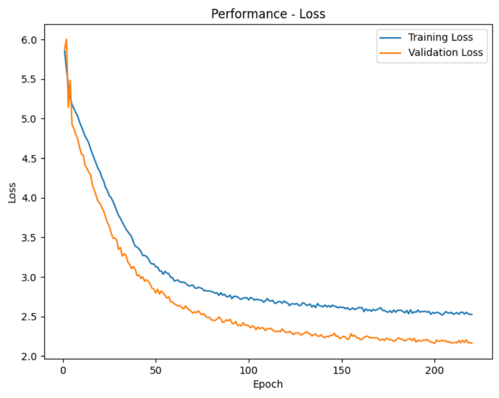
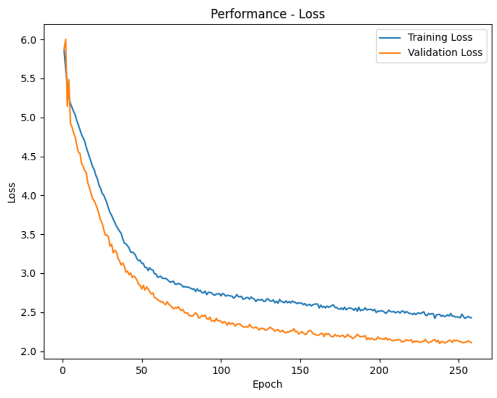


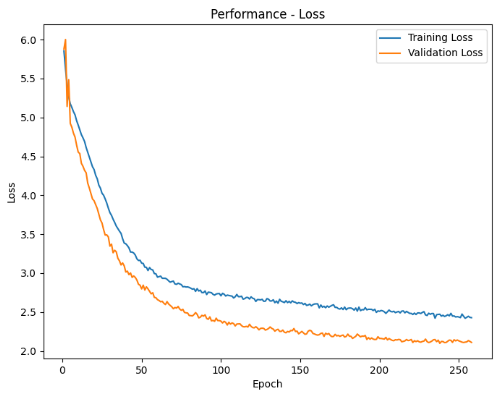
Figure 22: Learning rate CustomLRScheduler



(a) Configuration 1 (Adam)



(b) Configuration 2 (Adam)



(c) Configuration 3 (AdamW)

Figure 23: The loss of different ResNet50 models during training and validation phases

The loss curves shown in Fig. 23 indicate that the models with different optimizers demonstrated good generalization capabilities, as indicated by the decreased validation loss compared to the training loss across all experiments. In particular, the model in the first configuration 23a stopped learning earlier and achieved validation metrics over 74% (see Table 6). In contrast, the models in the other two configurations continued to train further and learn effectively until they stopped after epoch 261, reaching validation metrics over 76% (see Table 6). This indicates that the custom learning rate scheduler effectively managed the learning rate and allowed the models to continue learning and improving over a longer period. These

Metrics	**Configuration 1**	**Configuration 2**	**Configuration 3**
Training Accuracy	62.77%	64.03%	64.02%
Validation Accuracy	74.32%	76.19%	76.19%
Training Weighted F1 Score	62.57%	64.12%	64.11%
Validation Weighted F1 Score	74.23%	76.20%	76.20%

Table 6: Comparison of configuration on various evaluation metrics

results indicate that the models are not overfitting and have reached a good performance, as they maintained a balance between training and validation performance, demonstrating robustness and stability in the training process.

In conclusion, one of the models in configurations 2 and 3 can be chosen for the classification task in the final implementation. Since these models achieved similar validation metrics, the decision is made to select the model in configuration 2.

4.2 Results

All the experiments discussed in section 4.1 are conducted to select the classification model. This Section presents the results of the composed model.

After the classification model is selected, the composed detection model from the final implementation illustrated in Fig. 24 is evaluated on the test dataset, which contains 67 classes and 5109 images.

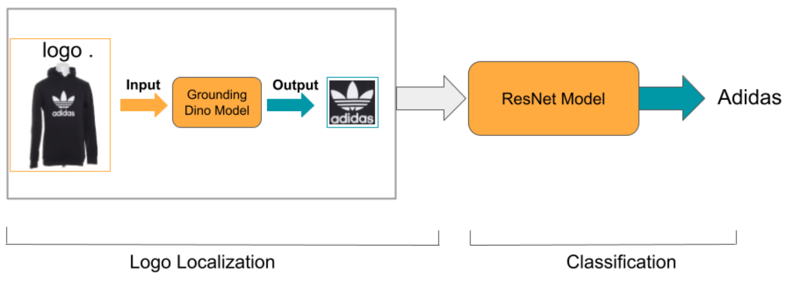


Figure 24: Detection model composed of Grounding Dino and ResNet50

Out of 5109 images, 3728 are included in the metrics calculation. The remaining 1381 images are classified as 'None' and excluded from the metrics calculation because they are assigned a brand name label but are classified as 'None' by the model. This issue could arise from the logo not being visible on the garment, the garment item lacking a logo, or the model being incorrect. This makes it uncertain whether the model or the label is correct.

4.2.1 Evaluation

Figure 25 illustrates the performance metrics of the composed model across several key evaluation metrics without the 'None' class being included.

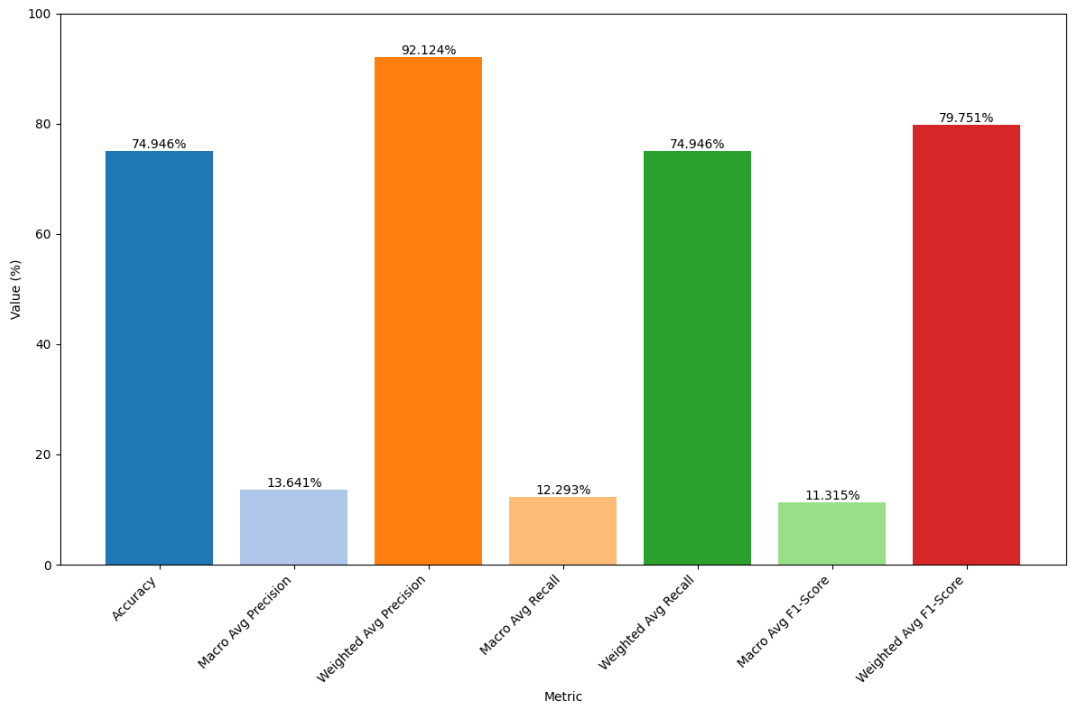


Figure 25: Overall performance in percentage

Although the overall accuracy of the model stands approximately at 75%, indicating that the model correctly detected 75% of 3728 images, it is less indicative of performance across all classes due to the imbalanced test dataset. The significant difference between macro and weighted averages indicates that while the model performs well in classes with many images, it struggles with less-represented classes. The F1-score, which balances precision and recall, has a weighted average of 79.751%, indicating a strong overall performance. Still, the macro average F1-score is low at 11.315%, which further highlights the difference in the model's performance across different classes. Table 7 shows the varying performance of some selected

Class Name	Precision	Recall	F1-score	Number of Images
Adidas	0.99	0.89	0.94	1472
Nike	0.89	0.81	0.85	949
Puma	0.85	0.79	0.82	284
Tommy Hilfiger	1.00	0.06	0.11	230
Calvin Klein	0.99	0.60	0.75	113
Vans	0.90	0.86	0.88	93
Hummel	0.96	0.70	0.81	71
C&A	1.00	0.08	0.14	13
Hanes	0.00	0.00	0.00	6
Nickelodeon	0.00	0.00	0.00	6
Zoo York	0.50	1.00	0.67	1

Table 7: Comparison of classification performance for selected classes

classes without the 'None' class, whereas the performance results for all other classes, excluding the 'None' class, are provided in Table 9 in Appendix A.
The model performs well for classes with a large number of images such as Adidas, Nike, and Puma. Adidas has a significantly high precision (0.99) and recall (0.89), which results in a strong F1-score of 0.94. This high value indicates the model's reliability in identifying Adidas logos. Similarly, Nike and Puma also show good performance with precision scores of 0.89 and 0.85 and recall scores of 0.81 and 0.79, respectively, which give F1 scores of 0.85 and 0.82.

However, the model struggles significantly with less-represented classes such as Tommy Hilfiger, C&A, Hanes, and Nickelodeon. Despite perfect precision (1.00) for Tommy Hilfiger and C&A, their recall scores are very low at 0.06 and 0.08, respectively. This results in low F1-scores of 0.11 for Tommy Hilfiger and 0.14 for C&A. These values indicate the model rarely misclassifies other logos as these brands but fails to detect most true samples of these logos. For Hanes and Nickelodeon, both precision and recall are 0.00, resulting in an F1-score of 0.00. This poor performance implies that the model fails entirely to identify these brands due to the very small number of images available for these classes. This discrepancy highlights the model's difficulty in generalizing to less-represented classes and highlights the need for a more balanced dataset or improved handling of class imbalances.

All images classified as 'None' by the model are saved in a folder and manually reviewed to evaluate the correctness of these classifications. The results of this review for some selected classes are detailed in Table 8, which provides detailed information on the total number of images per brand, the number of images classified as 'None', and how many images are correctly classified as 'None' for some selected classes.

Brand	**Total number of Images**	**Number of images Classified as None**	**Correctly Classified as None**
Adidas	1909	437	76
Nike	1103	154	94
Puma	353	69	16
Tommy Hilfiger	366	136	73
Calvin Klein	149	36	16
Vans	105	12	7
Hummel	93	22	4
C&A	39	26	25
Hanes	9	3	3
Nickelodeon	11	4	4
Zoo York	3	2	2

Table 8: Classification of images as 'None' for selected classes

As mentioned earlier, 1381 images are classified as 'None'. Most of them (1036 images) belong to the classes Adidas, Nike, Tommy Hilfiger, Desigual, Puma, Lacoste, and Diesel.

Adidas and Puma images have a high rate of being classified as 'None', but the correctness of these classifications is low, indicating a need for model improvement in distinguishing Adidas and Puma images from the 'None' category. In contrast, Nike images show a moderate rate of being classified as 'None' with a high correct classification rate. This suggests the model performs relatively well in correctly distinguishing Nike images from the 'None' category. Hanes, Nickelodeon and Zoo York have all their images classified as 'None' correctly (Hanes with 3 out of 3, Nickelodeon with 4 out of 4 and Zoo York with 2 out 2).

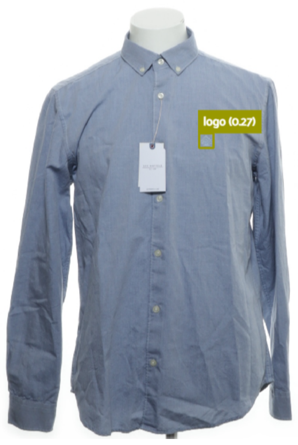
The evaluation of the composed detection model shows general effectiveness with high precision, recall and F1 scores, particularly for well-represented classes like Adidas and Nike. However, the model struggles with less-represented classes, as seen in the significant difference between the macro and weighted averages, represented in Fig. 25.

The exclusion of the 'None' class from the performance metrics and the high number of 'None' classifications for certain brands highlight areas for improvement. The model's difficulty in consistently detecting logos from less-represented classes and its occasional failure to detect logos at all (resulting in 'None' classifications) indicate a need for further refinement and possibly more balanced training data.

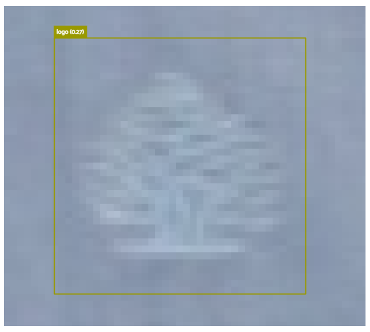
Overall, while the model performs well for certain classes, the variability in performance across different brands and the high rate of 'None' classifications point that further enhancements are needed to improve its reliability and F1 scores for all classes.

4.2.2 Predictions

This section presents some predictions made by using the composed model. As seen in Fig. 26, the Grounding Dino model detected the location of the logo correctly 26a, which is identified by the ResNet50 model as 'None' 26b. This is incorrect because this garment Item contains the logo of the Springfield brand. One possible explanation could be the insufficient number of images featuring such logos.



(a) Garment Item



(b) The patches extracted

Figure 26: Logo on garment item classified as 'None'

Fig. 27 shows a garment item that does not contain any visible logo. Although the Grounding Dino model mistakenly detected another object in the image as a logo 27a, the ResNet model classified it correctly as 'None' 27b.

(a) Garment Item

(b) The patches extracted

Figure 27: Logo on garment item classified as 'None'

5 Conclusion and Future Work

This thesis focused on logo detection on garments by utilizing deep learning techniques. It investigated several advanced automated detection models to accelerate and reduce errors using manual detection methods. Commonly used detection models can not be directly applied without being trained on an annotated logo dataset. Therefore, a composed model is presented in this thesis as an advanced automated solution that combines a detector and a classification model. The first is used to localize logos, and the latter is used for logo identification. Specifically, the Grounding Dino model is used as a detector and the ResNet50 as a classification model. One of the main contributions of this thesis is the selection of a suitable classification model based on experimental observations gained during the training process. As a result of these observations, ResNet50 was chosen as the model of choice. Subsequently, ResNet50 was trained on a publicly available logo dataset to be able to classify logos.

A comprehensive evaluation of the composed model's performance is presented. The evaluation results showed that although the model performs well for some classes, the variability in performance across different brands and the high rate of 'None' classifications indicate that further improvements are necessary to enhance its reliability and F1 scores for all classes. This is left as future work.

The findings emphasize the potential of deep learning techniques in automating brand logo detection on garments, offering valuable insights for future research and practical applications in the retail and e-commerce sectors.

References

[1] Md Zahangir Alom et al. “A state-of-the-art survey on deep learning theory and architectures”. In: *electronics* 8.3 (2019), p. 292.

[2] Laith Alzubaidi et al. “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8 (2021), pp. 1–74.

[3] Benjamin Beltzung et al. “Deep learning for studying drawing behavior: A review”. In: *Frontiers in Psychology* 14 (2023), p. 992541.

[4] Simone Bianco et al. “Benchmark Analysis of Representative Deep Neural Network Architectures”. In: *IEEE Access* 6 (2018), pp. 64270–64277. DOI: 10.1109/ACCESS.2018.2877890.

[5] databricks. *Machine Learning Models*. https://www.databricks.com/glossary/machine-learning-models. Accessed: 2024-06-25.

[6] Synho Do, Kyoung Doo Song, and Joo Won Chung. “Basics of deep learning: a radiologist’s guide to understanding published radiology articles on deep learning”. In: *Korean journal of radiology* 21.1 (2020), p. 33.

[7] PyTorch Documentation. “CrossEntropyLoss”. In: *PyTorch* (2023). Accessed: 2024-07-03. URL: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

[8] Yulia Gavrilova. “Data Preprocessing in Python”. In: *Serokell* (Jan. 2024). Accessed: 2024-06-28. URL: https://serokell.io/blog/data-preprocessing-in-python.

[9] Pouya Hallaj. *Data Augmentation: Benefits and Disadvantages*. Accessed: 2024-06-19. Sept. 2023. URL: https://medium.com/@pouyahallaj/data-augmentation-benefits-and-disadvantages-38d8201aead.

[10] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[11] Mohammad Hossin and Md Nasir Sulaiman. “A review on evaluation metrics for data classification evaluations”. In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[12] IBM. *What is Computer Vision?* https://www.ibm.com/de-de/topics/computer-vision. Accessed: 2024-06-18.

[13] Christian Janiesch, Patrick Zschech, and Kai Heinrich. “Machine learning and deep learning”. In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[14] Vijay Kanade. *What Is Machine Learning? Definition, Types, Applications, and Trends*. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/. Accessed: 2023-10-01. 2022.

[15] Asifullah Khan et al. “A survey of the recent architectures of deep convolutional neural networks”. In: *Artificial intelligence review* 53 (2020), pp. 5455–5516.

[16] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[18] Shilong Liu et al. “Grounding dino: Marrying dino with grounded pre-training for open-set object detection”. In: *arXiv preprint arXiv:2303.05499* (2023).

[19] NVIDIA. *What Is a Machine Learning Model?* https://blogs.nvidia.com/blog/2021/05/14/what-is-a-machine-learning-model. Accessed: 2024-07-03. 2021.

[20] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Accessed: 2024-07-10. 2019. arXiv: 1912.01703 [cs.LG].

[21] Shreya U. Patil. "Loss Functions and Optimizers in ML Models". In: *Geek Culture* (Jan. 2023). Accessed: 2024-07-04. URL: https://medium.com/geekculture/loss-functions-and-optimizers-in-ml-models-b125871ff0dc.

[22] *PyTorch Workflow*. Accessed: 2024-07-03. URL: https://www.learnpytorch.io/01_pytorch_workflow/.

[23] Shafin Rahman, Salman Khan, and Fatih Porikli. "Zero-Shot Object Detection: Learning to Simultaneously Recognize and Localize Novel Concepts". In: *Asian Conference on Computer Vision* (2018).

[24] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[25] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[26] Stacey Ronaghan. *Introduction to Deep Learning*. https://srnghn.medium.com/introduction-to-deep-learning-what-do-i-need-to-know-75794ebc4a62. Accessed: 2024-06-25.

[27] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed: 2024-06-25.

[28] Iqbal H Sarker. "Machine learning: Algorithms, real-world applications and research directions". In: *SN computer science* 2.3 (2021), p. 160.

[29] Sellpy. *Sellpy: Second-hand Shopping Platform*. Accessed: 2024-06-28. 2024. URL: https://www.sellpy.com.

[30] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)* (2015). URL: https://arxiv.org/abs/1409.1556.

[31] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[32] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas". In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716.

[33] Andras Tüzkö et al. "Open set logo detection and retrieval". In: *arXiv preprint arXiv:1710.10891* (2017).

[34] Ultralytics. *YOLOv8—Ultralytics YOLOv8 Documentation*. https://docs.ultralytics.com/models/yolov8/. Accessed on 17 Juni 2024. 2023.

[35] Jing Wang et al. "Logo-2K+: A Large-Scale Logo Dataset for Scalable Logo Classification". In: *AAAI Conference on Artificial Intelligence. Accepted*. 2020.

[36] Jing Wang et al. "Logodet-3k: A large-scale image dataset for logo detection". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 18.1 (2022), pp. 1–19.

[37] Ahmet Okan Yilmaz. *Loss Functions and Optimization Algorithms in Deep Learning*. Accessed: 2024-07-03. 2023. URL: https://aoyilmaz.medium.com/loss-functions-and-optimization-algorithms-in-deep-learning-fc9602a55b79.

[38] Jiyang Zheng et al. "Towards open-set object detection and discovery". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3961–3970.

A Appendix

The code used in the implementation of this thesis is available on GitHub and can be accessed through the following link:
https://github.com/aimanalhazmi/Brand-Detection-on-Garments.git

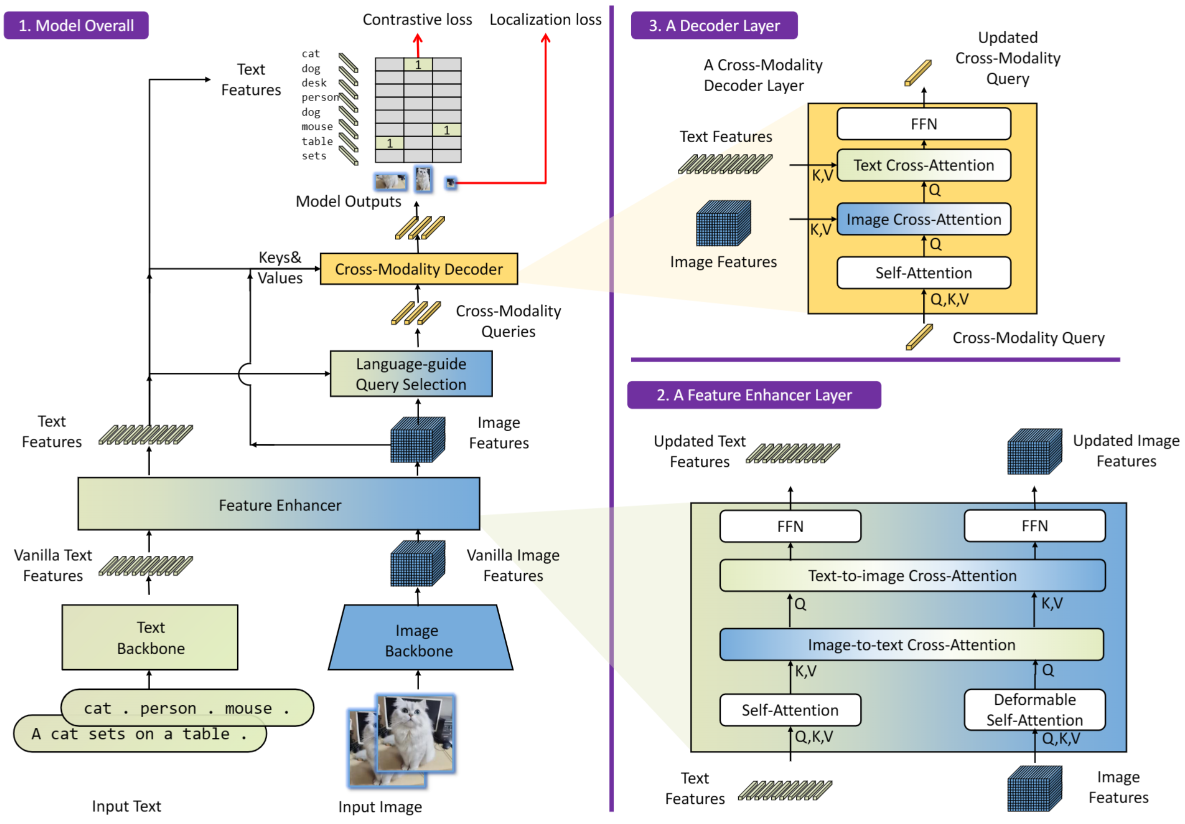


Figure 28: The framework of Grounding DINO[18]

The varying performance of of all classes without the 'None' class are detailed in Table 9 and 10.

Table 9: Evaluation metrics for each brand without the 'None' class

Brand	Precision	Recall	F1-Score	Number of Images
Adidas	0.998	0.891	0.941	1472
Nike	0.890	0.812	0.850	949
Puma	0.852	0.789	0.819	284
Tommy Hilfiger	1.000	0.061	0.115	230
Calvin Klein	0.986	0.602	0.747	113
Vans	0.899	0.860	0.879	93
Ellesse	0.975	0.951	0.963	81
Hummel	0.962	0.704	0.813	71
Reebok	0.973	0.529	0.686	68
Gap	0.917	0.800	0.854	55
Desigual	0.500	0.156	0.237	45
Diesel	0.583	0.226	0.326	31
Next	0.556	0.217	0.313	23
Star Wars	0.923	0.545	0.686	22
Svea	1.000	0.619	0.765	21
CA	1.000	0.077	0.143	13
Columbia	0.625	0.833	0.714	12
Victorias Secret	1.000	0.200	0.333	10
Lacoste	1.000	0.400	0.571	10
Saucony	1.000	0.111	0.200	9
Triumph	0.250	0.250	0.250	8
Boss	0.350	0.875	0.500	8
Batman	0.800	0.571	0.667	7
Hanes	0.000	0.000	0.000	6
Speedo	0.545	1.000	0.706	6
Nickelodeon	0.000	0.000	0.000	6
CCM	0.750	1.000	0.857	6
GStar	0.333	0.200	0.250	5
Arena	0.333	0.800	0.471	5
Roxy	0.400	0.400	0.400	5
Everlast	0.667	1.000	0.800	4
MP	0.667	0.500	0.571	4
Emilio	0.000	0.000	0.000	4
Element	1.000	0.250	0.400	4
Russell	0.000	0.000	0.000	3
Mizuno	1.000	1.000	1.000	3
Sloggi	0.000	0.000	0.000	3
Fusion	0.300	1.000	0.462	3
Salewa	0.667	0.667	0.667	3
Ed Hardy	0.500	1.000	0.667	2
Fox	0.000	0.000	0.000	2
Disney Pixar	0.000	0.000	0.000	2
Burberry	1.000	0.500	0.667	2
Geox	0.000	0.000	0.000	2
Zoggs	0.000	0.000	0.000	2
Trespass	1.000	1.000	1.000	1
Chicago	1.000	1.000	1.000	1
Zoo York	0.500	1.000	0.667	1

Table 10: Evaluation metrics for each brand without the 'None' class (2)

Brand	**Precision**	**Recall**	**F1-Score**	**Number of Images**
Escada	1.000	1.000	1.000	1
Reef	0.500	1.000	0.667	1
Pineapple	0.000	0.000	0.000	1
Nautica	0.000	0.000	0.000	1
Cat	0.063	1.000	0.118	1
Hurley	0.250	1.000	0.400	1
Givova	0.000	0.000	0.000	1
Bounce	0.000	0.000	0.000	1