Ca' Foscari
University
of Venice

Master's Degree programme

in Data Analytics for Business and Society

Final Thesis

Exploring CNNs and Attention Mechanisms for Brand Identification in Fashion Runway Shows

Supervisor
Ch. Prof. Luca Cosmo

Graduand
Elena Martarello
Matriculation Number 872265

Academic Year
2022 / 2023

To all the women who would have liked to study.

To all women, period!

Una donna che riesce, riesce per tutte le altre.

Acknowledgements

Ringrazio solo le persone che mi vogliono bene veramente.

Abstract

In today's fashion landscape, characterized by an abundance of competing brands, establishing a unique and captivating visual identity has emerged as an essential pillar of effective branding strategies.
The core challenge that drives our research is the extraction of brand-specific information from a diverse array of runway fashion presentations and the subsequent classification of these images into six distinct fashion brands. To this end, we developed a sophisticated deep learning model, specifically a Convolutional Neural Network (CNN)-based classification model enriched with attention mechanisms. Accurate brand classification could signifies the presence of a highly recognizable brand, one that boasts a robust and distinctive visual identity. Conversely, when our model yields lower accuracy in brand classification, it hints at the possibility of a weaker or less distinctive visual identity for the brand in question. The versatility and the applicability of this model in the fashion industry is evident in its multifaceted utility across various domains. Fashion brands can leverage this tool to gain insights into their brand identity, thereby enhancing their ability to resonate with their target audiences effectively. It could be transformed into a tool aimed at amplifying fashion houses' ability to resonate deeply with their target audiences, creating stronger connections and achieving greater engagement. To accomplish this, our training process heavily relies on a meticulously annotated dataset of fashion images, where each image is accompanied by detailed brand information, forming the bedrock of our model's training and learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the current competitive fashion enviorment, where numerous brands compete for recognition, having a unique and captivating visual identity has become a fundamental element of effective branding strategies. The aim of our research is to assess the utility of a Convolutional Neural Network (CNN) with an attention mechanism for the multiclassification of images into six distinct fashion brands. The interesting aspect of our approach is the utilization of an attention mechanism to extract brand-specific information from the images, enabling us to discern and classify them accurately. A brand that is consistently and accurately classified can be considered highly recognizable, reflecting a strong and distinctive visual identity. Conversely, if a brand's classification accuracy is lacking, it may indicate a weaker or less distinctive visual identity.

To achieve our goal, we annotated a dataset of fashion images. This dataset contains comprehensive information about the brands represented in the images, allowing our model to learn and generalize the distinctive features associated with each brand. The applicability of our research extends beyond the realm of machine learning and image classification. Understanding the nuances of brand identity in the fashion industry holds significant implications for brand differentiation and establishing a meaningful connection with the target audience. In a crowded marketplace, where consumers are exposed to an abundance of visual stimuli, the ability to identify and communicate a brand's unique visual characteristics is a key driver of success. In conclusion, our research delves into the realm of fashion image classification with a focus on brand recognition. By leveraging CNN with attention, we aim to enlighten the visual identities of fashion brands, ultimately contributing to a deeper understanding of how these identities influence consumer perception and loyalty.

In this chapter, we will take a closer look at the topic of visual identity, with an example aimed at helping the reader better understand the roots of the problem that this thesis aims to address. Furthermore, even in these early pages, we will provide an analytical insight by approaching the subject from a data analysis perspective. We will discuss the benefits as well as the limitations that this approach inherits from its statistical-mathematical framework.

1.1 Importance of Brand Identification

In today's bustling marketplace, where countless brands vie for attention, a distinctive and captivating visual identity has become a cornerstone of successful branding strategies. [15] Visual identity transcends mere aesthetics; it encapsulates the essence of a brand, communicating its unique attributes and resonating with consumers on a profound level. At the heart of this concept lies the interplay between brand identity and visual identity – two interconnected facets that collaboratively shape how a company is perceived. The visual identity of a brand refers to the visual and aesthetic aspects that identify a brand. It encompasses visual characteristics such as the logo, colors, patterns, graphic style, and design that communicate the brand's identity and values to consumers. As intuition, Visual Identity is a part of Brand identity. In simple terms, brand identity built all the distinctive elements that make a company recognizable to the public. On the other hand, visual identity is the collection of visual languages and elements that a company utilizes to communicate. With these basic notions, it becomes clear how visual identity plays a fundamental role in the success of a brand by establishing an emotional connection with the target audience.

Visual identity acts as the face of a brand, offering consumers a first impression that can significantly impact their perception of the company's values, personality, and quality. A well-crafted visual identity not only captures the essence of a brand's story and ethos but also communicates professionalism and attention to detail. A distinctive logo, for instance, serves as a succinct representation of a brand's mission and values, helping consumers recognize and remember it in a crowded marketplace.

To better capture the idea of what Visual and Brand Identity means, see the *Etro Case Study*, which is a practical explanation of these concepts applied to the brand Etro.

Etro Case Study 1

Before going into practical, note that Etro was founded in 1968 by Gerolamo "Gimmo" Etro, and it is an Italian luxury fashion house known for its bohemian-luxe aesthetic.

Originally focused on textiles, it expanded to ready-to-wear fashion, accessories, and home furnishings. Etro remains a family-run brand, celebrated for its vibrant prints and eclectic designs. It worth to note that haute couture and ready-to-wear are two distinct categories within the fashion industry, differing significantly in terms of production, design, and exclusivity. Haute couture, derived from French, translates to "high sewing" or "high dressmaking", representing the pinnacle of craftsmanship and luxury. It involves creating custom-made garments tailored to individual clients' precise measurements and preferences. Haute couture pieces are meticulously hand-crafted by skilled artisans, utilizing intricate techniques and luxurious materials, resulting in one-of-a-kind creations. Due to its exceptional craftsmanship and exclusivity, haute couture is characterized by limited production and high price points. In contrast, ready-to-wear, also known as prêt-à-porter, refers to clothing lines designed for mass production and general consumption. These collections are produced in standardized sizes and are readily available for purchase off-the-rack in stores or online. While ready-to-wear garments may still feature high-quality materials and craftsmanship, they are designed to appeal to a broader audience, striking a balance between style, quality, and accessibility. In Figure 1.1 there is on the left the templete of to build or read the brand identity and the brand visualization. On the right there is the application ment for this case study.

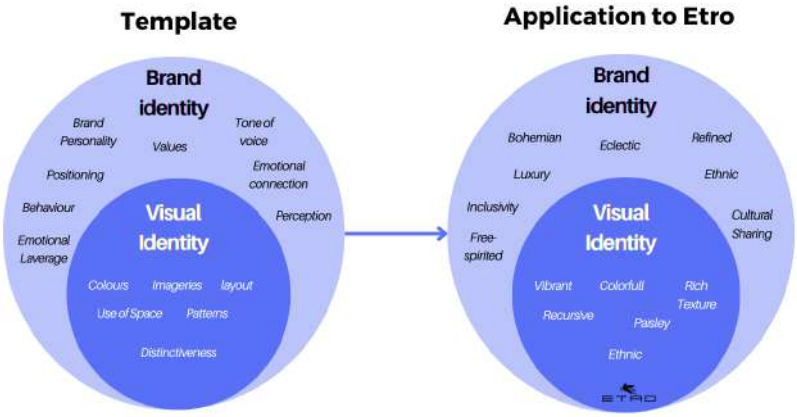

Figure 1.1: Application of Brand and Visual Identity to Etro

Brand Identity of Etro

Etro is known for its unique blend of bohemian luxury and eclectic style. The brand's identity revolves around the values of craftsmanship, innovation, and a celebration of cultural diversity. The brand targets a sophisticated, free-spirited audience that appreciates the artistry and individuality expressed through its collections. Etro's

brand identity is associated with a sense of adventure, wanderlust, and a love for unconventional beauty. In Figure 1.2 there are two example of the Etro brand identity application.

Figure 1.2: An example of free-spirited audience for Brand Identity of Etro - Images of SS23

Visual Identity of Etro - Images from multiple collections

Etro's visual identity is characterized by its iconic paisley patterns, which have become synonymous with the brand. The paisley motif, inspired by traditional Indian and Persian designs, is often incorporated into Etro's clothing, accessories, and even its logo. The brand's visual language exudes a bohemian elegance, with a mix of cultural references and artistic elements. Etro's visual identity is instantly recognizable and conveys a sense of uniqueness and a love for artistry and craftsmanship. looking at Figure 1.3 we can see how Etro play with its motif in several collections.

Figure 1.3: The paisley motif for Visual Identity of Etro - Images from multiple collections

Brand identification in today's competitive business landscape is crucial for success. It encompasses the process by which consumers recognize and associate a brand with its unique qualities, values, and offerings. Building a strong brand visual identity is crucial for the success and sustainability of a business, regardless of its industry. First and foremost, brand identification plays a significant role in creating brand loyalty. When consumers can easily identify and differentiate a brand from its competitors, they are more likely to develop an emotional connection and trust in that brand. This emotional bond fosters customer loyalty, encouraging repeat purchases

and positive word-of-mouth recommendations[1]. Brand identification enables businesses to cultivate a community of loyal customers who become brand advocates, driving long-term growth and success. Furthermore, brand identification facilitates effective brand positioning. By clearly defining and communicating a brand's unique selling proposition and value proposition. This positioning allows businesses to attract and retain their desired customer segments, establishing a competitive advantage in the market. Brand recognition and recall are led by several aspects. If it is strong, it increases the likelihood of being top-of-mind during purchase decisions. Whether through visual elements such as logos, colors, and packaging or through consistent messaging and brand voice, brand identification enables businesses to leave a lasting impression and enhance brand recall.

The aim of this work would be to provide insights into the distinctive aspects of brand visual identity. The idea of using algorithms to detect recurring patterns, colors, patterns, or specific product features[2] of a brand can contribute to better brand identification. However, it is crucial to recognize that brand identity goes beyond the visual aspect. It involves a range of elements such as company philosophy, values, brand experience, and customer interactions. These factors can profoundly influence consumers' perception of a brand and the emotional connection they develop with it.

While a classification model can provide information about visual identity, it is important to emphasize that image analysis represents only a part of the overall brand identity. Other components, such as brand communication, tone of voice, reputation, company history, and social impact can influence the brand's image and perception. The Balenciaga scandal, well explained in the New York Times article "When High Fashion and QAnon Collide" by Elizabeth Paton and Vanessa Friedman, involving promotional photos depicting children in bondage-like poses serves as a compelling example of how mishandling visual identity can be detrimental to a company. This incident underscores the potential consequences of a controversial and poorly executed visual identity, which can undermine a previously strong and successful brand image. In this instance, the ill-conceived promotional images ignited significant backlash and public outrage. The shocking and inappropriate nature of

[1]Word-of-mouth recommendations are when satisfied customers voluntarily share positive experiences with others, influencing their purchasing decisions through personal testimonials and recommendations.

[2]Features refer to specific and meaningful characteristics or patterns extracted from data. In the context of machine learning, they represent relevant information used to make predictions or perform tasks effectively.

the images was deemed offensive and inappropriate, resulting in a barrage of negative media coverage and social media uproar. This controversy not only tarnished Balenciaga's reputation but also eroded the trust and loyalty it had built with its audience over the years. The subsequent chaos, including instances of customers burning their Balenciaga merchandise in protest and the ensuing negative media coverage, showcased the power of visual identity in shaping public perception. The incident underscored that even a well-established brand with a solid history can face a severe blow if its visual representation contradicts its values and audience expectations.

Ultimately, the Balenciaga case serves as a stark reminder that a brand's visual identity is a reflection of its core values and messaging. When not aligned with these aspects or when poorly executed, it can lead to damaging repercussions that erode the foundation of a previously successful brand identity. Another aspect to consider while classifying the brand in the deep learning model is the issue of brand identification. It's essential to consider the concept of rebranding, which can pose unique challenges. Rebranding involves altering a brand's visual identity, messaging, or overall image to adapt to changing market dynamics or to address issues that may have arisen over time. While rebranding can be a strategic move to revitalize a brand, it also carries risks. For instance, in the world of fashion, changes in creative direction and the appointment of new artistic directors can significantly impact a brand's visual identity. The transition from one creative vision to another may lead to shifts in the brand's aesthetics, potentially causing confusion among consumers accustomed to a particular brand image. In the upcoming section, we will address this potential concern.

1.1.1 Visual Identity in Fashion and its Variation through Releases

As already said the visual identity of a fashion brand extends beyond a mere logo; it encompasses everything from color palettes, patterns, and typography to photography style, models, and overall design aesthetics.In the medium-to-high luxury brand, visual identity often starts from a well-defined baseline. This visual identity then evolves across various collections, absorbing the inspiration and conceptualization of each, until it seamlessly integrates with the brand's core identity. In other words visual identity for fashion industry is by no means static. Instead, it undergoes a continuous evolution throughout the successive releases of various collections. At the onset of each collection's creative journey, the brand draws inspiration and conceptualizes a distinct theme or narrative. [39] This inspiration then becomes a

guiding force, shaping the design direction, color schemes, and even the finer details of the collection's visual components.To clarify this concept, excerpts of attire have been selected from collections spanning several years under the Rick Owens brand, see from *Figure 1.4* to *Figure 1.6*.

Figure 1.4: Rick Owens Ready To Wear Fall Winter 2015 Paris

Figure 1.5: Rick Owens Ready To Wear Spring Summer 2016 Paris

Figure 1.6: Rick Owens Ready To Wear Spring Summer 2020 Paris

Despite originating from diverse years, encompassing both Fall/Winter and Spring/-Summer seasons, a prominent underlying theme becomes evident, even amidst substantial diversity among the garments themselves. Foremost among these observations is the distinct color palette, in which gray assumes a prominent role as a neutral or achromatic tone, positioned midway between black and white. These shades, indeed, are abundant. However, the shared traits extend beyond color; the designs of the outfits display mutual gentleness and remarkable eccentricity, signifying a unique

characteristic that binds them together. While the ensembles within the Stella collection demonstrate noteworthy similarity across various groups, a recognizable brand identity is apparent even within individual assortments. Similarly remarkable is the consistent alignment of the chosen locations with the attire's palette. Moreover, the selected locales appear notably subdued when juxtaposed with the outfits, further enhancing the overall aesthetic appeal.

As each collection takes shape, the underlying inspiration merges harmoniously with the brand's existing visual identity. This amalgamation is not merely a superficial overlay, but rather a deep integration that seamlessly bridges the thematic elements of the collection with the overarching identity of the brand. It's akin to weaving threads of creativity and innovation into the fabric of the brand's established character. This process of evolution and integration is very complex and considering that there are typically two collection releases per year and genders, this process represents a substantial commitment.To better understand how the construction of a release takes place, it's essential to examine its key phases. The first phase is that of inspiration and conceptualization. Here, the central idea of the collection is born, drawing from sources such as art, culture, or personal experience. This concept becomes the guiding thread that informs the creation of the unique visual identity of that specific release. Next, we move on to the development and design phase. Designers translate the initial idea into concrete designs, selecting fabrics, colors, and materials in line with the collection's theme. This phase demands advanced technical skills and close collaboration between designers and artisans. Once the design is finalized, the production and manufacturing phase begins. This stage involves material selection, prototype creation, and quality control to ensure consistency with the brand's vision. The timeline for this process can span several weeks or months, depending on the collection's complexity. The marketing and promotion phase is crucial for generating interest around the collection. Through teasers, previews, and behind-the-scenes content, anticipation is built among the audience before the official launch. This phase is vital for creating excitement and engaging the target audience. Finally, we arrive at the collection's presentation through runway shows or presentations. This moment offers an opportunity to showcase the creations in a curated and engaging manner, often involving industry insiders, celebrities, and media [19]. In *Figure 1.7* a recent example of involving celebrities as guests.

Beginning with initial inspiration, it traverses multiple phases to become an integral part of the brand's visual identity. This process demands effort, time, and attention to detail. The journey of visual identity in the realm of fashion brands, particularly those occupying the medium-to-high segment, is a continuous evolution

that begins with a well-defined foundation. As inspiration flows into each successive collection, it transforms and refines the visual identity, ultimately converging with the brand's core essence. This approach not only sustains the brand's uniqueness but also fosters a compelling narrative of evolution, adaptability, and creative expression.

Figure 1.7: Elton John, Lil Nas, Miley Cyrus as guests in Versace's FW 2023 Runway show in Los Angeles

1.2 Role of Deep learning

Deep learning is playing a significant role in analyzing and understanding brand-related topics, including brand identification and visual identity. With the advancements in data availability, computational power, and algorithmic techniques, deep learning has become a powerful tool for extracting insights from vast amounts of data, including visual data. In the context of brand identification, Deep learning algorithms can be trained on large datasets of images to classify and identify brand-related features. For example, image classification models can be used to recognize logos, patterns, colors, or specific product characteristics associated with different brands. These models can provide automated and scalable solutions for analyzing visual elements that contribute to brand identity.

To cite another kind of application, which is a recommendation system [1], we can take a look at the paper named "Study of AI-Driven Fashion Recommender Systems", authored by Shaghayegh Shirkhani, Hamam Mokayed, Rajkumar Saini and Hum Yan Chai [36], which takes into accounts many application. The authors explain how the recommendation can be applied to the whole outfit as part of it. One of the most interesting applications concerns the Similar Item Recommendation

[1] A recommendation system is a technology that suggests personalized recommendations or suggestions to users based on their preferences, behavior, or similarity to other users.

(Item Retrieval). Researchers have explored various image retrieval methods, including Text-based, Content-based, Multimodal Fusion, Semantic-based, and Relevance Feedback-based Image Retrieval. Another innovative approach of DL to fashion is Capsule Wardrobe Recommendations. This concept of a capsule wardrobe is often referred to as a minimalist closet or wardrobe detox. A capsule wardrobe aims to provide a minimal set of clothing items and accessories from a given inventory, allowing for maximum mix-and-match outfit combinations. These techniques enable businesses to gain insights into customer sentiments, brand perception, and emerging trends contributing to a deeper understanding of brand identity.

In this study, Deep Learning will be used to identify salient points in a photo, which will then be used for brand recognition. The model developed with the primary objective of accurately assigning each outfit image to its corresponding brand can have several valuable applications in the fashion industry and beyond. Here are some possible applications:

1. Consumer Perception: If the model identifies outfits accurately across different events, it suggests that the brand's visual identity is resonating well with designers, stylists, and other industry professionals, which can often influence consumer perceptions.

2. Brand Recognition and Recall: A strong visual identity leads to better brand recognition and recall. By analyzing how frequently and consistently the model recognizes a brand's outfits, one can gauge the effectiveness of the brand's visual cues [37].

3. Trend Analysis: Monitoring how well a brand adapts its visual identity to current fashion trends can also indicate its strength. The model can identify if a brand manages to maintain its identity while staying relevant in the ever-changing fashion landscape [7].

4. Consistency Check: The model can analyze the consistency of a brand's visual identity across different fashion events. If the same or similar visual elements (colors, patterns, logos) are consistently recognized in outfits attributed to a particular brand, it indicates a strong and consistent visual identity.

5. Fashion Forecasting: The model's output can contribute to fashion forecasting by identifying emerging brands that are gaining traction in the industry. This insight can guide retailers and designers in staying ahead of trends.

6. Intellectual Property Protection: Brands can use the model to monitor and protect their intellectual property. It can help detect unauthorized use of brand designs or products on runways and events[21].

7. Customized Recommendations: E-commerce platforms can use the model to provide personalized product recommendations to users based on their preferred brands and styles.

Overall, deep learning plays a crucial role in leveraging data-driven insights to understand and enhance brand identification and visual identity.

Chapter 2

Deep Learning and CNNs for Images Analysis

Deep Learning, according to IBM in its topic "What is deep learning?" [1], is a subset of deep learning focused on training artificial neural networks to perform complex tasks. These networks consist of layers that progressively learn features from data. Through backpropagation, they adjust their internal parameters to minimize errors and improve accuracy. Deep learning excels in tasks like image and speech recognition, natural language processing, and game playing. It has revolutionized AI by enabling computers to autonomously learn and make decisions from large datasets. Its depth, scalability, and hierarchical learning mimic the human brain's structure, allowing machines to achieve human-level performance in various domains. In recent years, there has been significant research in the area of image classification, driven by the availability of large databases such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset[29]. Deep Convolutional Neural Networks (CNNs) are a class of artificial neural networks designed to process and analyze data that has a grid-like topology, like in *Figure 2.1*.

They have proven to be good in different computer vision tasks, that is why they

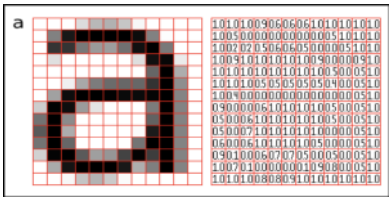

Figure 2.1: Representation of image as a grid of pixels from "Convolutional Neural Networks, Explained" Article by Mayank Mishra, published in "Towards Data Science" on Aug 26, 2020. [22]

are a crucial part of the data scientist CCNs became even more sophisticated over the years, starting with AlexNet [42] proposed by Alex Krizhevsky et al., followed by similar-depth architectures like Zeiler and Fergus's model [45]. Some early work on convolutional neural networks (CNNs) dates back to the late 1980s and early 1990s, they did not gain significant attention and popularity during that time due to various limitations, including the lack of processing power, limited availability of large datasets, and the absence of efficient training algorithms.

The concept of CNNs can be traced back to the 1980s with the work of Yann LeCun, who developed the first successful application of CNNs called LeNet-5[18] in 1998. LeNet-5 was primarily used for handwritten digit recognition tasks. However, during this period, the computational power needed to train and deploy deep neural networks like CNNs was indeed limited, which hindered their widespread adoption. In the early 2000s, there was a renewed interest in neural networks and deep learning, which led to some progress in training deeper architectures, but CNNs were still not the dominant approach. Instead, other machine learning algorithms, such as Support Vector Machines (SVMs), were popular for various computer vision tasks.

It wasn't until around 2012 that CNNs experienced a resurgence and revolutionized the field of computer vision. The main reasons for this revival of interest can be attributed to a few reasons. To mention a few the availability of large datasets, hardware upgrades, and improved training algorithms. As already mentioned, the ImageNet project was one of the first significant steps forward in the ascent of CNNs. It started releasing massive labeled image datasets containing millions of images in 2009. At the same time, the advancements in hardware provide significant computational power and acceleration for training deep neural networks, including CNN. Generally, the Graphics Processing Units (GPUs) became more accessible.

Another important reason is the improvement of training algorithms. Novel techniques like the Rectified Linear Unit (ReLU) activation function and better weight initialization methods helped in training deeper networks more effectively, overcoming the vanishing gradient problem[1].

[1]The gradient problem refers to a challenge encountered in training deep neural networks, which a class of machine learning models. These networks consist of numerous layers, each containing interconnected nodes that process and transform data. During training, the model adjusts its internal parameters based on a loss function to minimize prediction errors. The gradient problem arises from difficulties in propagating gradients – derivatives of the loss function with respect to the model's parameters – through many layers. As the gradients are backpropagated, they can either become vanishingly small (vanishing gradient) or excessively large (exploding gradient). This can hinder the convergence of the training process, making it difficult for the model to learn effectively from data. Researchers have developed various techniques to mitigate the gradient problem, such as careful weight initialization,

The last crucial point to note is that researchers discovered that pre-training CNNs on extensive datasets and subsequently fine-tuning them for specific tasks resulted in faster convergence and enhanced performance.

VGG16[38] is a convolutional neural network architecture that was introduced in 2014 by researchers from the Visual Geometry Group (VGG)[1] at the University of Oxford. The team, led by Karen Simonyan and Andrew Zisserman, aimed to design a deep-learning model that could achieve high accuracy in image classification tasks. VGG16 is so-called because it has 16 weight layers, including 13 convolutional layers and 3 fully connected layers. The VGG16 architecture was mainly employed in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30], where the objective was to categorize images into 1,000 distinct classes like "cat," "dog," "car," "bird," and more. VGG16 participated in this competition and achieved a groundbreaking level of performance. VGG16 achieved significant success in the ILSVRC 2014 [30], demonstrating the effectiveness of deep convolutional neural networks for image classification tasks. Its accuracy in the competition helped solidify the importance of deep learning in computer vision and marked a turning point in the field. Since then, VGG16 and its variations have been widely adopted and adapted for various computer vision tasks, including object detection, image segmentation, and feature extraction in transfer learning [2] scenarios.

using specific activation functions, and employing gradient clipping or normalization methods. These approaches help stabilize the gradient flow, enabling more successful training of deep neural networks.

[1]The VGG16 architecture was part of a series of network designs by the VGG team. The "VGG" in the name stands for "Visual Geometry Group."

[2]Transfer learning is a method within the domain of machine learning where an already trained model serves as an initial foundation for addressing a fresh and related problem. In the context of VGG16, transfer learning involves leveraging the knowledge learned from training VGG16 on a large dataset (e.g., ImageNet) and applying it to a different, but related, task.

2.1 Understanding the "Deep" in Deep Learning and CNNs

Prior to delving into the functionality of CNNs, let's briefly concentrate on Convolutional Neural Networks (CNNs). We'll delve into the "deep" aspect of deep learning, focusing on the increased layers and features in a CNN compared to traditional neural networks. This will help clarify that while CNNs are a type of deep learning model, they are not synonymous with deep learning itself, as the concept of deep learning existed before the introduction of CNNs. The term "deep" in deep learning has two main implications:

- **The number of layers**: In deep learning frameworks, such as Convolutional Neural Networks (CNNs), there are typically more layers than in traditional multi-layer perceptron or standard neural networks[1]. Some CNN architectures can have up to 150 layers, making them significantly deeper. Traditional neural networks vary in layer count but often have 1 to 5 hidden layers. So, deeper architectures with 10+ layers are common in deep learning, enabling complex learning from data.
- **The number of features:** Each layer in a CNN learns multiple "features" through sets of weights[2] that connects it to the preceding layer. This aspect makes the network much deeper than a regular neural net, where each layer typically learns only one set of weights.

Despite this depth, there are powerful neural networks, including some CNNs, that consist of just a few layers. Hence, the term "deep" in deep learning reflects the fact that each layer learns multiple features [13]. It is worth noting that deep learning is not synonymous with CNNs, as the concept of deep learning existed before the introduction of CNNs.

[1]Traditional multi-layer perceptrons, also known as standard neural networks, are machine learning models consisting of interconnected layers (input, hidden, and output) that process data to learn patterns, make predictions, and perform various tasks. These models use forward propagation to compute outputs and backpropagation to update weights during training.

[2]In Deep learning, weights are parameters associated with the connections between nodes (neurons) in a neural network. They determine the strength of each connection, affecting the flow of information during training. Adjusting the weights through iterative processes like backpropagation enables the network to learn and make accurate predictions.

2.2 Model Architecture

In this section, we will provide an overview of the model architecture of CNNs, followed by a more detailed exploration of their key components. At its core, a CNN is composed of multiple layers that work collaboratively to process and extract meaningful information from the input data. There three main types of layers are:

- Convolutional Layers: These layers find patterns and features in images, starting with basic elements like edges and gradually identifying more complex shapes as they go deeper into the network [5].
- Pooling Layers: They reduce the size of the data while keeping important information intact by selecting the most significant values within small regions [41].
- Fully Connected Layers: These layers analyze the high-level relationships in the extracted features and are crucial for making final decisions, like recognizing objects in images [6].

Looking at *Figure 2.2*, we can see a graphical rapresentation of a typical CNN.

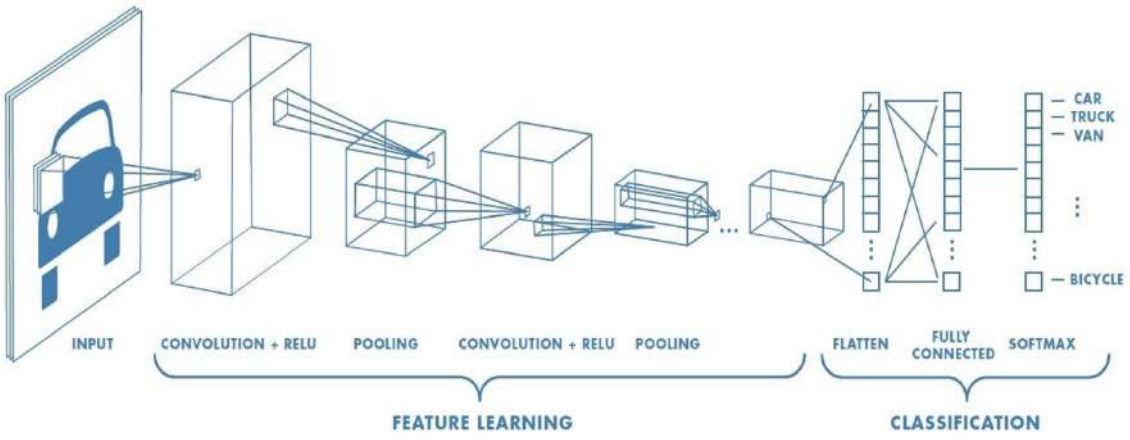

Figure 2.2: Representation of how a CNN works (source: "A Comprehensive Guide to Convolutional Neural Networks") by Sumit Saha. [31]

It's really important to note that sometimes CNNs, as in the case of the network implemented for this thesis, also incorporate an attention mechanism. An "attention mechanism" in machine learning refers to a mechanism that allows the model to focus on specific parts or aspects of the input data, giving them more weight or attention during processing. It's a way to enhance the model's ability to capture important information from the input.

2.2.1 Convolutional Layers

Francois Chollet in his book "Deep Learning with Python" [8], published by Manning Shelter Island, in 2018, clearly explains how convolutional layers are the cornerstone of CNNs. The basic operation in a convolutional layer is called convolution. In simple words: they consist of small filters, also known as kernels, which slide over the input data in a systematic manner. As they move across the input, they perform element-wise multiplication and addition to produce a feature map, highlighting distinctive patterns found in the data. These patterns can range from simple edges to more complex shapes and textures. What distinguishes a dense layer from a convolutional layer is that the former recognizes patterns across the entire input space, while the latter focuses on parts of images.

First, the model takes into account the input image as a matrix of pixel values with multiple channels (e.g., RGB images have three channels – red, green, and blue). CNNs have one or more filters and each filter is a small matrix that is randomly initialized during the training process. This means that width and height shrink as the images go deeper into the network. This characteristic gave several powerful properties to CNN. The most important advantage is translational invariance, which means that a conv layer can recognize patterns or features in input data regardless of where they are located within the input. In other words, it doesn't matter if a particular feature or pattern is in the center, left, right, top, or bottom of the input; the conv layer can still detect it.

Moreover, convolutional layers are capable of learning hierarchical representations of patterns. This means they can learn to recognize simple features (e.g., edges) in the early layers and then build upon these features to recognize more complex patterns in the deeper layers. Here's a step-by-step explanation of how convolution works inspired by "Student Notes: Convolutional Neural Networks (CNN) Introduction" [26]: Convolution operates over 3D tensors[1] and these 3D tensors represent data such as images, where each dimension corresponds to a different aspect of the data: Width, Height and Depth or Channels Note that the depth dimension represents different channels or layers of information. In the context of images, it can represent color channels (e.g., Red, Green, Blue - RGB), or in more general cases, different feature maps or channels extracted by previous layers in a CNN. The output of the convolution is still a 3D tensor. In convolutional operation, two parameters can be

[1]In deep learning, a tensor is a multi-dimensional array that can store and represent data of varying complexities and dimensions. PyTorch's official documentation gives a practical definition between a numpy array and a PyTorch Tensor: the last one can run on either CPU or GPU.

custom set. First is the size of Patches (Kernel Size), which refers to the dimensions of the filter (also called a kernel) that is used during the convolution operation (like 3x3 or 5x5). The kernel size determines the spatial extent of the local region in the input data that the convolution operation considers at a time. A larger kernel captures more extensive spatial information. The second one is the depth of the output feature map, often referred to as the number of filters or channels, which represents the number of distinct kernels or filters applied to the input data. The use of multiple filters allows the convolutional layer to learn different features and patterns at different levels of abstraction. The filter is slid across the input image in both the horizontal and vertical directions. At each position, the filter is element-wise multiplied with the corresponding region of the input image, and the results are summed up to produce a single value in the output feature map. This process is repeated for every position in the input image to generate the entire feature map. The filter's movement across the input image can be controlled by the stride parameter. The stride determines how many pixels the filter shifts at each step. A stride of 1 means the filter moves one pixel at a time, while a stride of 2 means it moves two pixels at a time, and so on. Padding is an elective method employed to maintain the spatial dimensions of the input image [24]. It encompasses the addition of additional pixels along the periphery of the input image before executing the convolution operation. Padding can be advantageous in preventing information loss at the image's edges.

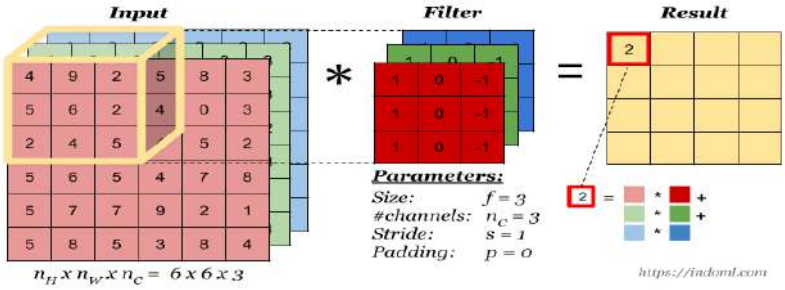

Figure 2.3: How Convolution Operation works from Student Notes: Convolutional Neural Networks (CNN) Introduction

After convolution, an activation function (commonly ReLU - Rectified Linear Unit) is applied element-wise to introduce non-linearity in the feature maps. This allows the network to learn more complex relationships in the data. Typically a convolutional layer consists of multiple filters, see *Figure 2.3* and *Figure 2.4* for visual representation. Each filter learns to detect different patterns and features from the input image. The number of filters in a convolutional layer determines the depth (number of channels) of the output feature map. Convolutional layers are

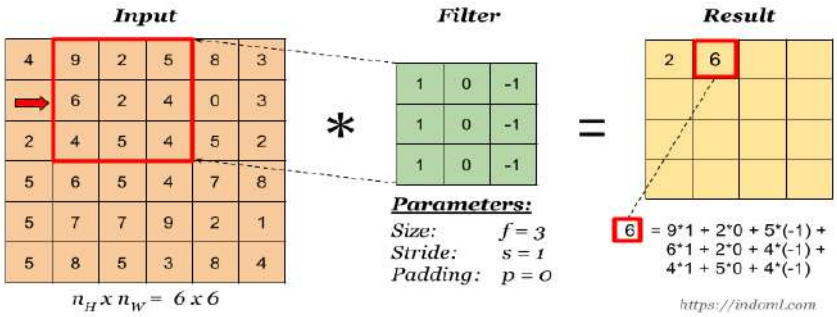

Figure 2.4: How Convolution Operation works from Student Notes pt2: Convolutional Neural Networks (CNN) Introduction

stacked together to form deep CNN architectures. By chaining multiple convolutional layers, the network can learn increasingly complex and abstract features from the input data, enabling it to recognize intricate patterns and objects in images. As the network progresses through the layers, the receptive field of filters increases, allowing the model to learn high-level features.

2.2.2 Pooling

Pooling layers come after convolutional layers and play a crucial role in Convolutional Neural Networks (CNNs). Their main purpose is to downsize the feature maps, primarily by diminishing their spatial dimensions while preserving critical details. These pooling layers contribute to enhancing the CNN's resilience to variations in input data, cutting down on the number of parameters, and mitigating overfitting.

Figure 2.5 provide a visual representation of the most commonly used pooling techniques:

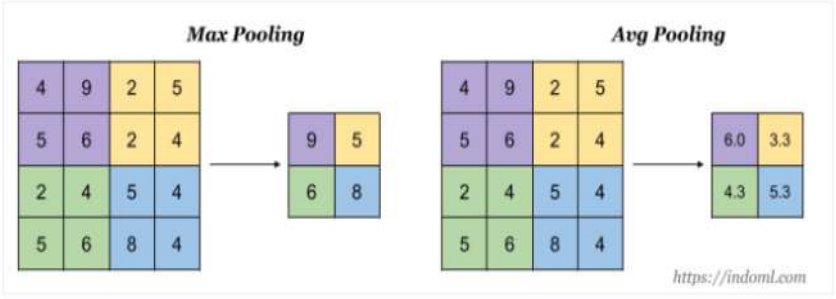

Figure 2.5: How Max and Avg Pooling Operation works from Student Notes pt2: Convolutional Neural Networks (CNN) Introduction

Max Pooling takes a small region (usually a 2x2 or 3x3 window) and outputs the maximum value from that region to the new pooled feature map. The operation is applied independently to each channel of the feature map. Here's a step-by-step

explanation of how Max Pooling works:

1. **Input Feature Map:** The input to the pooling layer is the feature map obtained from the preceding convolutional layer. The feature map is a matrix representation of learned features from the input image. In other words, a feature map refers to the output of a specific filter applied to input data, such as an image, in a convolutional neural network. It highlights significant patterns or features for further processing.

2. **Pooling Operation:** The pooling layer slides a small window (e.g., 2x2 or 3x3) over the feature map and applies the pooling operation independently to each window. In the case of Max Pooling, the maximum value within each window is extracted and placed in the corresponding position of the pooled feature map.

3. **Stride**: Similar to the convolutional layer, pooling layers can also have a stride parameter. The stride controls the step size at which the pooling window moves across the feature map. A stride of 2, for instance, means the pooling window moves two units at a time [28].

4. **Reducing Spatial Dimension**s: Since the pooling operation replaces each window with a single value (the maximum value), it effectively reduces the spatial dimensions of the feature map. For example, a 2x2 Max Pooling with a stride of 2 will halve the width and height of the feature map.

5. **Number of Channels:** The pooling operation is performed independently on each channel (the 3rd dimension of the 3D tensor) of the feature map, which means the number of channels remains unchanged after pooling.

To make the explanation more effective, see *Figure 2.6* for a visual example of the Max Pooling operation.

The main benefits of pooling layers can be summarized in three crucial points. Pooling layers make CNNs more robust to small translations and distortions in the input data. Since the pooling operation considers only the maximum value in each local region, it captures the most important features, even if they are slight. Moreover, it reduced parameters. By reducing the spatial dimensions of the feature maps, pooling layers decrease the number of parameters in the subsequent layers of the network. This helps manage computational complexity and memory requirements. Lastly; as the CNN progresses through multiple pooling layers, the receptive field of

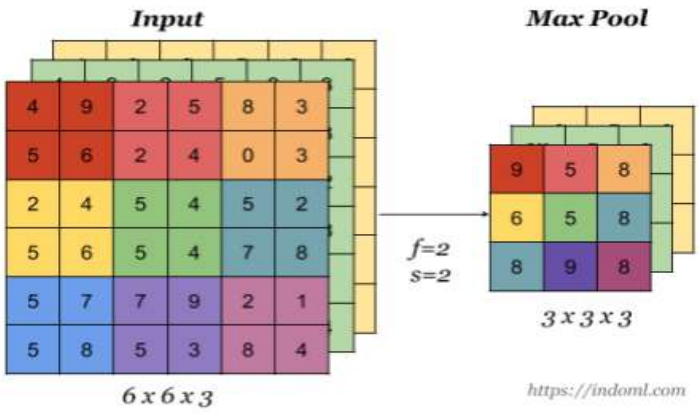

Figure 2.6: An example of Max Pooling Operation result from Student Notes pt2: Convolutional Neural Networks (CNN) Introduction

the network increases, allowing it to learn high-level and more abstract features. It is worth noting that in addition to Max Pooling, there are other types of pooling, such as Average Pooling, which takes the average value within each pooling window instead of the maximum value. While Max Pooling is the most common choice, the choice of pooling technique may vary based on the specific problem and architecture of the CNN.

2.2.3 Fully Connected Layers

After multiple convolutional and pooling layers, fully connected layers are introduced. These layers establish connections between each neuron in the previous layer and every neuron in the current layer, resembling the dense connections found in conventional neural networks. Fully Connected Layers, also referred to as Dense Layers, constitute the final elements of a Convolutional Neural Network (CNN) architecture. Unlike convolutional and pooling layers, which mainly focus on learning spatial features from the input data, fully connected layers are responsible for making predictions and performing high-level reasoning based on the extracted features.

Before passing the data through the fully connected layers, the output feature map from the last convolutional layer is typically flattened into a one-dimensional vector. This means that all the spatial information is collapsed, and the features are represented in a single continuous sequence. A fully connected layer consists of a set of neurons, each corresponding to a specific class or category in the output. These neurons are fully connected to all the elements in the flattened feature vector. The number of neurons in the fully connected layer is equal to the number of classes or categories in the classification task. Every connection between the neurons in the

fully connected layer and the elements in the flattened feature vector has an associated weight. These weights are learned during the training process and represent the importance of each feature for each class. Additionally, each neuron has a bias term, which allows for fine-tuning the prediction threshold for a particular class. Like in other layers of the CNN, a non-linear activation function (commonly ReLU or softmax) is applied to the output of each neuron in the fully connected layer. Rectified Linear Unit is often used for intermediate fully connected layers, while softmax activation is used for the final output layer in classification tasks. The softmax activation function ensures that the final outputs are probabilities, representing the likelihood of the input belonging to each class. Fully connected layers enable the CNN to use the learned features from the convolutional and pooling layers to make predictions and classify the input into specific categories. The information flow in these layers is fully connected, allowing the model to capture complex patterns and relationships in the data. The final output of the fully connected layer is usually a probability distribution over all the possible classes in the classification task.

The class with the highest probability is considered the predicted class for the input. It is important to note that not all CNN architectures have fully connected layers at the end. In some cases, the last convolutional layers output is used as the final output, particularly in tasks such as object detection, where the model needs to predict the positions and sizes of objects in addition to their classes. However, for most image classification tasks, fully connected layers are a common component to make predictions based on the extracted features.

2.2.4 Activation Function

Activation functions are a crucial component of Convolutional Neural Networks (CNNs) and other types of artificial neural networks. They introduce non-linearity to the model, allowing it to learn complex and non-linear relationships in the data. In CNNs, activation functions are applied after convolutional and fully connected layers to introduce non-linearity to the output of neurons. The purpose of activation functions can be summarized as follows:

- **Introducing Non-linearity**.
- **Enabling Gradient Flow**

Without activation functions, the output of a neuron in a neural network would be a linear function of its inputs. Linear functions can only learn linear relationships in the data, severely limiting the model's ability to represent complex patterns and make meaningful predictions. Activation functions introduce non-linearity, enabling

the neural network to approximate any arbitrary function, making it capable of learning from diverse and intricate data patterns. During the training process, neural networks optimize their weights to minimize a loss function, often using gradient-based optimization algorithms like backpropagation[1]. Activation functions play a critical role in this process by providing gradients that indicate the direction and magnitude of weight updates. Without activation functions, the gradients would be constant or too simple, leading to inefficient learning or getting stuck in local minima during optimization. Now we are going through the most commonly used activation functions in CNNs[2]:

1. ReLU (Rectified Linear Unit): ReLU is the most widely used activation function. It computes the output as the maximum between the input and zero, effectively turning off neurons with negative inputs and keeping neurons with positive inputs active [2]. ReLU is computationally efficient and helps mitigate the vanishing gradient problem.

2. Sigmoid: The sigmoid activation function compresses the input values within a range of 0 to 1. Traditionally, the sigmoid function was applied in the output layer for binary classification tasks, but its usage in hidden layers has become less common today because of the vanishing gradient issue. [35]

3. Tanh (Hyperbolic Tangent): The tanh activation function compresses the input values into a range spanning from -1 to 1. Tanh is akin to the sigmoid function but generates outputs centered around zero, which makes it somewhat more advantageous for the learning process [25].

4. Leaky ReLU: Leaky ReLU is a variation of the ReLU activation that addresses the "dying ReLU" problem, where neurons can become inactive during training and never recover [11] Leaky ReLU introduces a small negative slope for negative

[1]Backpropagation is an algorithm used in training artificial neural networks. It calculates the gradients of the model's weights by propagating the error backward from the output layer to the input layer. This process enables the network to learn and adjust its parameters during training iteratively. Note that gradients represent the slope or rate of change of a function concerning its input variables. In the context of machine learning, gradients indicate how the output of a model changes concerning its parameters, guiding optimization algorithms like gradient descent to update the model's parameters for better performance during training.

[2]"Activation Functions in Neural Networks Sigmoid, tanh, Softmax, ReLU, Leaky ReLU EXPLAINED !!!" by SAGAR SHARMA Published in 'Towards Data Science' on Sep 6, 2017 [32]

inputs, preventing neurons from dying.

5. Softmax: The softmax activation function is commonly used in the output layer of multi-class classification tasks. It takes a vector of raw scores and converts them into a probability distribution [33]. Softmax ensures that the sum of the probabilities of all classes is equal to 1.

Choosing the right activation function for a CNN depends on the specific task and the characteristics of the data. In practice, ReLU and its variants are widely used due to their simplicity, efficiency, and effectiveness in avoiding the vanishing gradient problem. For the purpose of this thesis, we choose Softmax and ReLu.

2.2.5 Attention Mechanism

Charu C. Aggarwal in his book named "Neural Networks and Deep Learning", published by Springer in 2023 [3], define attention mechanism as a computational mechanism that allows a neural network to focus on specific parts of input data (such as an image, text, or sequence) while performing a task. Instead of treating all input elements equally, an attention mechanism assigns varying degrees of importance or relevance to different elements or regions of the input. Attention mechanisms enhance the ability of neural networks to capture and process relevant information, leading to improved model performance on complex tasks. This method is widely used in sequence-to-sequence tasks, such as machine translation and natural language processing, to handle variable-length input and output sequences effectively.

Highlighting the most influential parts of the input that contribute to a prediction should be a strategically essential aspect to consider during model training [44]. Having the ability to comprehend why the model makes a particular output decision, such as classification in this thesis, offers a significant advantage in dispelling the black box phenomenon. This matter encompasses various aspects that will be further explored later in this dissertation. Despite the advantages of events, attention can be dangerous; adding attention mechanisms can make models more complex, potentially requiring more training data and computational resources. Moreover, complex attention mechanisms can lead to overfitting, especially when there is limited training data.

It can be applied in various ways, including:

- Spatial Attention: Emphasizing specific spatial regions of feature maps to highlight important visual features. Which is the one I'm going to implement in the

model [43].

- Channel Attention: Focusing on particular feature map channels to capture relevant information in each channel [20].
- Global Average Pooling (GAP) Attention: Using global average pooling to compute attention weights across spatial locations or channels [14].
- Self-Attention: Applying self-attention mechanisms, as seen in transformers, to capture dependencies across all spatial positions and channels in feature maps[34].

At the moment, our primary focus is on spatial attention because it will be the one employed in this application. As for the other types, it suffices to be aware of their existence.

It operates based on the idea that our visual system has limited processing capacity, and we cannot process all visual information simultaneously. Spatial attention allows us to prioritize processing for relevant regions and suppress processing for irrelevant regions.

While spatial attention mechanisms are extensively employed in deep networks, there has been a scarcity of research and analysis dedicated to them. This is the primary focus of the paper titled "An Empirical Study of Spatial Attention Mechanisms in Deep Networks," authored by Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai [46]. Moreover, the authors support that there is a large range for improvement in the design of spatial attention mechanisms in deep networks, and the study's findings pave the way for further exploration and optimization in this field. The work also tries to understand the effects of different attention factors and mechanisms. Key findings from the study challenge conventional beliefs about the significance of query-sensitive attention. For instance, it suggests that in self-attention, query-sensitive terms, especially query and key content, play a minor role, while deformable convolution performs effectively in image recognition. This challenges the common perception that query-sensitive attention is the main driver of success in attention mechanisms. Note that, the query is a representation of the element or position in the input sequence for which we want to calculate the attention scores. The key is a representation of all the other elements in the input sequence. It provides information about these elements that is used to assess their importance or relevance concerning the query. The key helps in determining how much attention should be given to each element in relation to the query. Together, the query and key are used to calculate attention scores or weights, which indicate the degree of relevance or importance of each element in the input sequence concerning the query.

Another crucial decision is where to integrate spatial attention within the neural network architecture. Common choices include adding it to convolutional layers or incorporating it into the attention mechanisms of transformer-based models. Later, in this thesis, we will see an implementation on a conv layer. Moreover, when working with attention is crucial to specify what convolutional layers we want to incorporate spatial attention. Note that there is the possibility to add spatial attention to one or more convolutional layers based on the task's requirements and computational resources. When attention is initialized it is fed by the input feature maps from the previous convolutional layer and, optionally, the query content.

2.3 Model Train

In the preceding sections, we've examined the primary layers of a CNN, including the attention layer, which introduces training complexity. Once again, the book "Neural Networks and Deep Learning" by Charu C. Aggarwal published by Springer in 2023 [3], explains how expanding the network's depth comes with its own set of challenges. Deeper networks tend to be more challenging to train and are notably influenced by parameter and hyperparameter selection. This is attributed to the heightened complexity of the loss function due to recursive composition as depth increases, resulting in unpredictable behaviors during gradient descent. Training the model is a crucial process of teaching a deep learning model to recognize patterns and make predictions or decisions based on input data. Remember that the goal of model training is to find the optimal set of parameters that minimize the loss function, allowing the model to make accurate predictions or decisions on new, unseen data.

The training process involves a combination of mathematical optimization, iterative updates, and fine-tuning of hyperparameters to achieve the desired level of performance. Hyperparameters determine the model's behavior, performance, and generalization ability. In other words; the choice of hyperparameters can impact the training process's efficiency. Different machine learning tasks and datasets may require different hyperparameters: what works well for one problem may not work as effectively for another. Here are some of the key hyperparameters along with brief explanations:

- Loss Function: The choice of a loss function defines how the model's performance is measured during training. It quantifies the error between the predicted values and the actual target values. Different tasks (e.g., classification, regression) require different loss functions (e.g., Mean Squared Error, Cross-Entropy) to optimize effectively.

- Optimization Algorithm: Optimization algorithms (e.g., Gradient Descent, Adam, RMSprop) determine how the model's parameters are updated based on the loss function. Each algorithm has its strengths and weaknesses, and the choice can significantly affect training speed and convergence.

In the following section, we're going deeper into Loss Function and Optimization Hyperparameters.

2.3.1 Loss Function

CNNs are trained using a loss function, which measures the disparity between the predicted outputs and the actual ground-truth labels. The goal during training is to minimize this loss, which is achieved using optimization algorithms like Stochastic Gradient Descent (SGD) or its variants. The Loss Function (also called the Cost Function or Objective Function) is a mathematical measure that quantifies the difference between the predicted output of the CNN and the true target labels. The goal during training is to minimize this loss, which reflects how well the model is performing on the training data. By minimizing the loss, CNN learns to make better predictions on unseen data. The selection of the loss function is contingent upon the particular task being addressed. Typical loss functions for various tasks encompass Mean Squared Error (MSE), which is applied in regression tasks where the output is continuous, aiming to minimize the average of the squared discrepancies between the predicted values and the actual values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

Another one can be Binary Cross-Entropy, which is used for binary classification tasks, where the output is a single value representing the probability of belonging to one of two classes. It measures the difference between the predicted probability and the true label. To name another one; Categorical Cross-Entropy, is used for multi-class classification tasks, where the output is a probability distribution over multiple classes. It measures the difference between the predicted probabilities and the true one-hot encoded labels. The categorical cross-entropy loss is used for multi-class classification problems[18]. It measures the dissimilarity between the true class probabilities (y_i) and the predicted probabilities ($\hat{y}_i$) for each class i.

$$\text{Categorical Cross-Entropy} = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \tag{2.2}$$

Where:

- C is the number of classes.
- y_i is the true probability of class i (one-hot encoded).
- $\hat{y}_i$ is the predicted probability of class i.

One benefit of using the loss function is that it heavily penalizes incorrect predictions. This emphasis on correctness helps the model focus on assigning high probabilities to the correct classes. Categorical Cross-Entropy can be sensitive to class imbalances, which is not the case. The logarithmic nature of Categorical Cross-Entropy can cause sensitivity to extreme predictions. Very confident but incorrect predictions can lead to disproportionately high losses[18].

2.3.2 Optimization

Optimization is the process of updating the parameters (weights and biases) of the CNN in order to minimize the chosen loss function. The goal is to find the optimal values for the model's parameters that make it perform well on the training data and generalize well to unseen data[10] The process of optimization involves an iterative approach using gradient-based algorithms. The most commonly used optimization algorithms in CNNs include:

- **Stochastic Gradient Descent (SGD):** During every iteration, SGD chooses a random subset of the training data known as a mini-batch. It employs this mini-batch to calculate the gradients of the loss function concerning the model parameters. Subsequently, SGD adjusts the parameters in the direction opposite to the gradients to reduce the loss.
- **Adam (Adaptive Moment Estimation):** Adam is an algorithm that adapts the learning rate for each parameter based on its past gradient [10].

Adam maintains a running average of both the first-order moments (the mean) and the second-order moments (the uncentered variance) of the gradients. This helps the optimizer adaptively adjust the learning rates for each parameter during training. The update step for the parameter θ at time step t in Adam is given by:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{2.3}$$

Where:

- θ_t is the parameter at time step t.
- η is the learning rate, determining the step size of the update.
- $\hat{m}_t$ is the biased first-moment estimate (mean of gradients) at time step t: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, where m_t is the raw first-moment estimate and β_1 is the exponential decay rate for the first moment.
- $\hat{v}_t$ is the biased second raw moment estimate (uncentered variance of gradients) at time step t: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$, where v_t is the raw second-moment estimate and β_2 is the exponential decay rate for the second moment.
- ϵ is a small constant to prevent division by zero.

Adam adapts the learning rates for each parameter based on their past gradients. This enables faster convergence and efficient updates, especially for features with sparse gradients or in high-dimensional spaces. Doing this requires the storage of past gradients' first and second moments for each parameter, which can be memory-intensive for large models and datasets. Moreover, it uses bias correction to counteract the initial bias of the moving averages. This is particularly helpful during the early stages of training when the moving averages are close to zero. Unfortunately, Adam has several hyperparameters such as the learning rate (η), and the exponential decay rates (β_1 and β_2). Poorly tuned hyperparameters can result in suboptimal convergence.

While Adam performs well in practice, its convergence guarantees are not as well-established as some other optimization algorithms like SGD with momentum.

2.4 Classical CNN Architectures

Thanks to the growing importance of CNNs, the studies in computer vision over the years have created various architectures. Each architecture is designed to tackle specific challenges in image recognition, object detection, segmentation, and other related tasks. In this introduction, I'll briefly outline some popular CNN architectures.

2.4.1 LeNet -5

LeNet-5 is one of the earliest CNN architectures, developed by Yann LeCun et al. in 1998. It was designed to recognize handwritten digits in images and played a significant role in popularizing CNNs for image recognition tasks. LeNet-5 consists of seven layers, see *Figure 2.7*, including convolutional, pooling, and fully connected

layers.

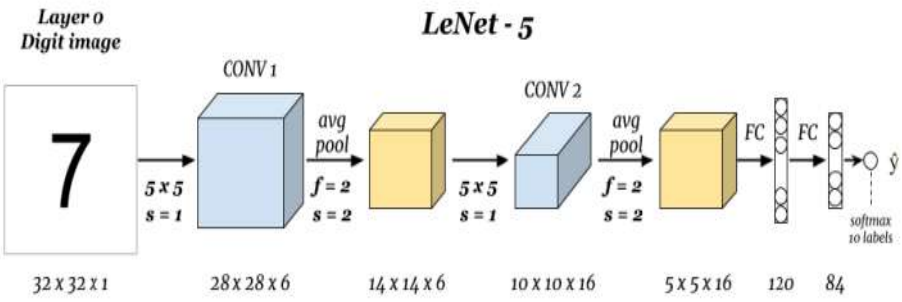

Figure 2.7: A graphical rapresentation of LaNet -5 from "Student Notes: Convolutional Neural Networks (CNN) Introduction" [26]

1. **Input Layer:** The input to LeNet-5 is a grayscale image of size 32x32 pixels. At the time of its development, this size was suitable for recognizing handwritten digits.

2. **First Convolutional Layer (C1):** The first layer is a convolutional layer with six filters, each of size 5x5. The filters move across the input image with a stride of 1 and apply element-wise multiplication with the corresponding pixels in the receptive field. The result is then summed up to create a feature map. The output of this layer is six feature maps with a size of 28x28, resulting from a reduction in spatial dimensions due to the 5x5 convolutional operation.

3. **First Average Pooling Layer (S2):** The first pooling layer is an average pooling layer with a filter size of 2x2 and a stride of 2. Average pooling replaces the 2x2 regions in each feature map with their average value, effectively reducing the spatial dimensions by half. The output of this layer is six feature maps with a size of 14x14.

4. **Second Convolutional Layer (C3):** The second convolutional layer has 16 filters, each of size 5x5. Like in the first layer, the filters convolve over the output of the first pooling layer (S2) and generate 16 feature maps of size 10x10.

5. **Second Average Pooling Layer (S4):** The second pooling layer is also an average pooling layer with a filter size of 2x2 and a stride of 2. It reduces the spatial dimensions further to 5x5, resulting in 16 feature maps of size 5x5.

6. **Fully Connected Layer (C5):** The fifth layer is a fully connected layer with 120 neurons. Each neuron is connected to every element of the 5x5x16 feature maps obtained from the second pooling layer. The fully connected layer learns high-level abstract features from the lower-level features learned by the

convolutional layers.

7. **Fully Connected Layer (F6):** The sixth layer is another fully connected layer with 84 neurons. It is connected to the 120 neurons of the previous layer (C5) and learns to represent even higher-level features.

8. **Output Layer:** The final layer is the output layer with ten neurons, corresponding to the ten possible digits (0 to 9). The output layer employs a softmax activation function to convert the output values into probabilities, indicating the likelihood of the input image representing each digit.

LeNet-5's architecture might appear simple compared to modern CNNs, but its design and success marked the beginning of using CNNs for image recognition tasks [4]. Despite its age, LeNet-5's principles and concepts still influence many contemporary CNN architectures

2.4.2 Vgg-16

VGG16 is a deep CNN architecture with 16 layers, hence the name "16." It gained popularity for its uniform structure, where most of the layers consist of 3x3 convolutional filters and 2x2 max pooling layers. The VGG16 architecture, visualized in *Figure 2.8*, has a straightforward and easy-to-understand design, making it widely used as a baseline model for various computer vision.

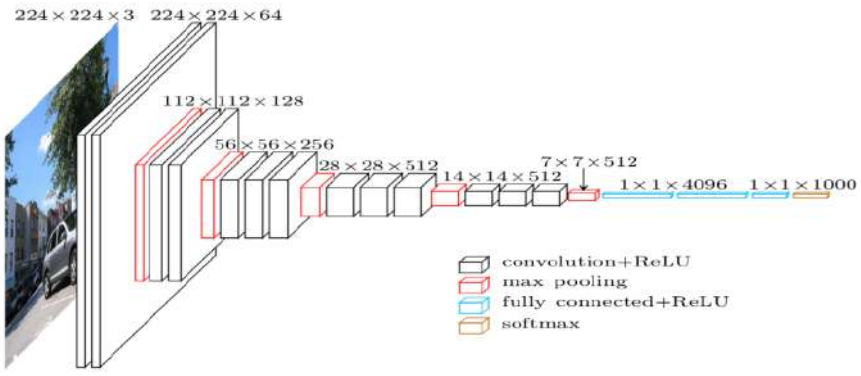

Figure 2.8: A graphical rapresentation of Vgg16 from "Student Notes: Convolutional Neural Networks (CNN) Introduction" [26]

1. **Input Layer:** The input to VGG16 is an RGB image of size 224x224 pixels.

2. **Convolutional Layers:** The model consists of 13 convolutional layers, with varying numbers of filters in each layer (ranging from 64 to 512). Each convolutional layer uses a small 3x3 filter with a stride of 1 and same padding.

3. **Max Pooling Layers:** After every two convolutional layers, there is a max pooling layer with a 2x2 filter and a stride of 2. Max pooling reduces the spatial

dimensions, making the model more computationally efficient and increasing the receptive field of the filters.

4. **Fully Connected Layers:** Following the convolutional layers, there are three fully connected layers with 4096 units each. These layers gradually reduce the spatial dimensions to a 1x1x4096 feature vector.

5. **Output Layer:** The final layer is the output layer, which consists of 1000 units representing different classes (assuming it's used on the ImageNet dataset, which has 1000 classes). The activation function for the output layer is usually softmax to convert the logits into probabilities.

VGG19 is an extension of VGG16 with 19 layers. It has a similar architecture to VGG16 but includes four additional convolutional layers, making it deeper. The extra layers allow VGG19 to capture more complex patterns and features, potentially leading to improved performance in certain tasks. However, the additional layers also make VGG19 computationally more expensive [23]. The architecture of VGG19 is almost identical to VGG16, with the only difference being the number of convolutional layers. The main difference is in the number of filters in each layer, which is also increased to maintain the same pattern of doubling the number of filters after each max pooling layer.

2.4.3 ResNet

ResNet, short for "Residual Network", is a deep convolutional neural network architecture proposed by Kaiming He et al. from Microsoft Research in 2015[16]. It addressed the challenge of training very deep neural networks by introducing the concept of residual blocks, enabling the successful training of networks with hundreds or even thousands of layers.

The main insight behind ResNet is the use of residual connections, also known as skip connections or shortcut connections. These connections allow the model to learn residual mappings, which makes it easier for the network to learn identity functions and enables better optimization during training. The skip connections directly connect the output of one layer to the input of a later layer, allowing the gradient to bypass several layers and flow directly during backpropagation. This helps alleviate the vanishing gradient problem, where the gradients become too small to effectively update the weights in very deep networks. Here are the key components of the ResNet architecture:

1. **Identity Block:** The building block of ResNet is the identity block, which is used to learn the residual mapping. It consists of a series of convolutional layers

with small 3x3 filters, followed by batch normalization and ReLU activation. The output of the final convolutional layer is added element-wise to the input of the block (shortcut connection), and the result is passed through another ReLU activation. The purpose of this block is to learn the residual (difference) between the input and output features.

2. **Convolutional Block:** The convolutional block is similar to the identity block but includes an extra convolutional layer with a 1x1 filter at the beginning. This additional layer helps to adjust the dimensions of the input and output if they are not the same. The remaining layers are the same as in the identity block, with a series of 3x3 convolutional layers, batch normalization, and ReLU activation.

3. **Skip Connections (Shortcut Connections):** The key innovation in ResNet is the skip connections, which allow the gradient to flow more efficiently during training. In the identity block, the input is directly added to the output, creating a shortcut connection. In the convolutional block, a 1x1 convolutional layer is used to adjust the dimensions of the input before adding it to the output.

4. **ResNet Architectures:** ResNet comes in different variants, depending on the depth of the network. The original ResNet paper proposed ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The numbers indicate the total number of layers in each variant (including convolutional, batch normalization, and ReLU layers).

2.5 Interpretability of the Model: Into the black box

The concept of understanding the interpretability of models, especially when dealing with complex ones like CNNs, holds great significance for various reasons. The rapid adoption of machine learning algorithms has surpassed the development of a clear framework to comprehend them. This has eventually led to the issue of the "black box problem", which suggests that these algorithms lack transparency and are challenging to interpret or explain, particularly in ways that humans can easily grasp. Maya Krishnan[1], in her article "Interpretability in Deep Learning", questions the prevailing consensus about the existence and importance of the black box problem. The terms such as "interpretability", "applicability", and related concepts lack

[1] "Against Interpretability: a Critical Examination of the Interpretability Problem in Machine Learning" published in 2020 [17]

precise definitions when applied to algorithms. This poses a challenge in using these concepts effectively to address the issues that have prompted the need for interpretability. Additionally, the absence of clear definitions makes it hard to determine whether specific technical features can provide formal definitions for these concepts.

A book titled "Interpretability in Deep Learning" authored by Ayush Somani [40], Alexander Horsch, and Dilip K., delves into understanding the functioning of deep learning and artificial neural networks (ANNs). These networks resemble computerized brains capable of learning and performing tasks akin to humans, such as recognizing faces, deciphering handwriting, and driving vehicles proficiently. Nevertheless, the enigma arises when these networks behave like black boxes, concealing the mechanisms behind their decisions. Imagine a self-driving car employing complex deep learning algorithms suddenly failing to slow down at a roundabout, potentially leading to dangerous situations. The opaqueness inherent in deep learning models curtails their usability in critical domains, necessitating a comprehensive grasp of their functionality, especially in contexts as crucial as healthcare. To tackle this challenge, researchers are working on enhancing the transparency and comprehensibility of Artificial Intelligence, termed as "eXplainable Artificial Intelligence (XAI)." This pursuit aims to imbue AI with fairness, transparency, and reliability. The book delves into the significance of understanding AI mechanisms, particularly in fields like medicine.

The ambiguity surrounding AI terminology becomes apparent, emphasizing the need for precise definitions. The authors also explore strategies to ensure the dependability, impartiality, and adherence to ethical principles in AI systems. The authors utilize the Five Ws—Who, What, Where, When, Why, and How—to systematically probe the intricacies of the black box problem. They highlight the challenge of making AI comprehensible, particularly in intricate fields like medicine. When introducing new systems, understanding their functioning and potential impacts is imperative. An inadequately trained AI system can lead to errors and susceptibility to manipulation, underscoring the need for a balance between optimal performance and interpretability. This equilibrium is depicted graphically in *Figure 2.9*, illustrating the pivotal roles of both aspects in cultivating responsible, robust, and accountable AI systems.

What *Figure 2.9* wants to express is that in the realm of Deep Learning, there's a trade-off between model interpretability and raw performance. This trade-off reflects the balance between having a model that can be easily understood and one that achieves exceptional predictive accuracy but is challenging to dissect. In simple words; Sometimes it's better to have an interpretable model than a model that

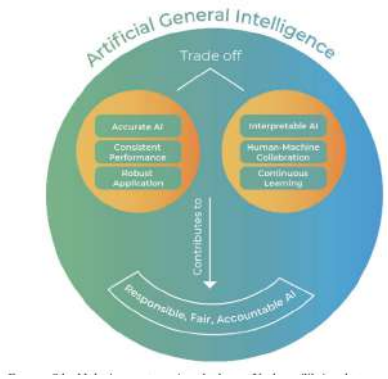

Figure 2.9: AI Payoff illustration from "Interpretability in Deep Learning" book. Authored by Ayush Somani, Alexander Horsch, and Dilip K. Prasad. It is published by Springer Cham. [40]

performs very well but can't be understood from within.

Sometimes, things get complicated because there are problems that can't be easily put into clear rules, like ethical questions. This makes it hard to fully understand or explain those things. The book provides examples like the difficulty of creating a fair criminal risk assessment tool aligned with human justice notions or ensuring safety while exploring scientific problems. It is really important to find out if they have any unfair biases. These biases could come from the data they were trained on and continue to cause unfairness. Interpretable Deep Learning (IDL) emerges as a valuable tool to uncover hidden relationships within intricate datasets, precluding unjust biases toward specific groups. The discussion underscores the potential of algorithms to yield harmful discriminatory outcomes when employed in automated decision-making processes. Initiatives such as the Association for Computing Machinery's (ACM) transparency and accountability statement and the Defense Advanced Research Projects Agency's (DARPA) eXplainable Artificial Intelligence (XAI) program strives to bestow models with clarity and accuracy. On the regulatory front, the European Union's General Data Protection Regulation (GDPR) plays a pivotal role, mandating explanations for automated decision-making. This regulatory landscape resonates across sectors handling personal data, including finance, social networks, and healthcare, necessitating accountability through measures like auditability, mitigation of adverse effects, ethical trade-offs, and compensation for unforeseen negative outcomes.

In the context of this project, several favorable aspects come to light in support of Convolutional Neural Networks (CNNs). According to Francois Chollet, the author of "Deep Learning with Python" by Manning Shelter Island in 2018 [8], the issue of the black box is mitigated when we delve into the realm of convolutional networks. Specifically, Francois highlights how since 2013, many methods have been developed

to visualize what happens inside the network. What is learned by ConvNets can now be visualized.

Francois mentioned three methods to visualize ConvNets:

1. Visualize Intermediate ConvNet Output (Intermediate Activation): As the input image passes through each layer, intermediate activations are produced. These intermediate activations represent the feature maps generated by each filter in a particular layer. Each feature map highlights a specific aspect of the input image that the corresponding filter is sensitive to.
2. Visualize ConvNet Filters: Convolutional filters are used in ConvNets to detect specific patterns or features in an image, such as edges, textures, or more complex structures. By visualizing the filters, I can gain insights into the network's feature extraction process.
3. Visualize Heatmap of Class Activation in an Image: Class activation maps (CAMs) are used to visualize which parts of an input image are most important in making a prediction for a specific class. This method helps to understand which regions of the input image contributed the most to the network's decision. By overlaying a heatmap on the input image, with hotter areas indicating higher activation for a particular class, it is possible to see where ConvNet is focusing its attention to make a classification decision.

Remember, as mentioned in *section 2.2.5*, that another crucial method to let the model begin more understandable is using an attention mechanism. To achieve this goal, which is to break the black box, enter it, and understand it, for this thesis, we implemented a classification model that internally incorporates an attention model. After training the model, we then examined what happens within the attention layers plotting the results and observing how attention changes in different scenarios. After the practical application that we will see in the next chapter, it will become evident how attention serves as a valuable strategy to make the model's functioning understandable, not only to data scientists but also to non technical stakeholders.

Black Box problem in corporate adoption

The black box issue is particularly relevant in industries such as fashion, where businesses heavily rely on these models for brand recognition and decision-making. In this context, let's explore the importance of interpretability through the lens of a fashion-focused CNN and its adoption within fashion enterprises.

Fashion companies, often structured in a hierarchical manner, tend to follow a top-down approach when adopting complex solutions. Consider a scenario where a

CNN is designed for brand recognition within the fashion sector. Such a solution holds the potential to impact diverse areas of decision-making across the company. To ensure its effective implementation, it's imperative that key stakeholders, particularly the Global Leadership Team (GLT) and GLT-1, are deeply involved in understanding, approving, and endorsing the solution. By involving these top-tier decision-makers at the outset, the adoption process becomes smoother for other managers and operational staff. Complex solutions like the CNN model, which enable strategic decision-making, are more likely to be embraced if they are first embraced by high-level executives. This initial buy-in builds trust in the decisions made by the model and encourages its usage across the organization. When decisions are made based on the model's recommendations, they are viewed with less skepticism, as the rationale behind each choice is clear.

In contrast, when organizations skip this top-down involvement and adopt complex models from the bottom-up, they often face challenges in garnering trust and understanding from higher levels of management. This can result in a lack of confidence in the model's decisions, hindering its effective implementation and leading to wasted resources in its development. Without the endorsement and comprehension of decision-makers at the strategic level, the intricate insights generated by the CNN model may go unnoticed or underutilized. To address this, the process of integrating complex models should emphasize the interpretability of the black box. This involves methods and tools that provide transparency into the decision-making process of the CNN model. Visualizations, feature importance scores, and other interpretability techniques can aid in showcasing how the model arrives at its conclusions. These insights bridge the gap between the technical intricacies of the model and the decision-makers understanding, enabling them to make informed choices based on the model's output.

In conclusion, while interpretable models may not consistently attain an high accuracy comparing to their complex counterparts, the preference often leans decidedly toward choosing interpretability over models that remain enigmatic or difficult to fathom.In other words, having a model that one can fully comprehend and explain is frequently deemed more desirable than possessing a highly accurate but opaque one.

Chapter 3

Application of a CNN to Runway Fashion Show Images

In this chapter, we present the practical implementation of CNNs to tackle the task of brand classification for a large dataset of runway fashion show images.

The model developed with the primary objective of accurately assigning each outfit image to its corresponding brand can have several valuable applications in the fashion industry and beyond. To achieve this, we leverage a VGG16 architecture enhanced with attention mechanisms, specifically using heatmap visualization. We explore transfer learning, initializing the VGG16 model with pre-trained weights from a large-scale image recognition task; the ImageNet. By leveraging pre-trained weights, we aim to capture meaningful image features that can be applied to our specific domain; runway fashion show images.

Additionally, we enhance the standard VGG16 architecture with attention mechanisms, allowing the model to focus on distinctive regions of the input image, hopefully, it should potentially improve its discriminative power. Lately, the practical implementation of the CNN-based will be fully detailed, viewing the model training, and testing phases, and conclude with a comprehensive visualization of the model's results using heatmaps.

3.1 Runway Dataset

In the context of fashion trend analysis and fashion show monitoring, one of the main challenges we encountered was the lack of datasets containing images grouped by brand and related to various fashion shows. To overcome this issue, we created an ad hoc dataset aimed at collecting detailed data on fashion brand shows. We identified the website "nowfashion.com" as a valuable source of information and images. The creation of a specific dataset for our purpose became essential for several reasons: The absence of preexisting datasets containing images organized by brand and fashion shows made fashion trend analysis and brand performance evaluation challenging. Collecting data from various unstructured sources could lead to inconsistent and low-quality data, while creating an ad hoc dataset allowed us to ensure the accuracy and consistency of the collected data.

To gather data from "nowfashion.com," we developed a web scraping script. This script allowed us to automatically extract fashion show images, brand-related information, dates, locations, and other relevant data. The collected data was then organized and saved into a CSV file as in *Figure 3.1*. We shared our dataset in Kaggle repository (https://www.kaggle.com/datasets/martarello elena/runways-from-major-brands-since-2011).

In *Figure 3.1* there is a sample of the final output.

collection_url	image_url	image_alt	image_name	image_path	image_size	image_width	image_height	image_format	image_mode	brand
https://door11.com/chanel-couture-spring-summe...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Couture-Spring-Summer-2023-Paris-Fashio...	chanel-couture-spring-summer-2023-paris/Chanel...	403519	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-couture-spring-summe...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Couture-Spring-Summer-2023-Paris-Fashio...	chanel-couture-spring-summer-2023-paris/Chanel...	602415	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-couture-spring-summe...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Couture-Spring-Summer-2023-Paris-Fashio...	chanel-couture-spring-summer-2023-paris/Chanel...	584394	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-couture-spring-summe...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Couture-Spring-Summer-2023-Paris-Fashio...	chanel-couture-spring-summer-2023-paris/Chanel...	517650	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-couture-spring-summe...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Couture-Spring-Summer-2023-Paris-Fashio...	chanel-couture-spring-summer-2023-paris/Chanel...	586667	1706	2560	JPEG	RGB	chanel
...	...	...	...	...	...	...	...	...	...	...
https://door11.com/chanel-ready-to-wear-spring...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Ready-To-Wear-Spring-Summer-2011-Paris-...	chanel-ready-to-wear-spring-summer-2011-paris-...	319313	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-ready-to-wear-spring...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Ready-To-Wear-Spring-Summer-2011-Paris-...	chanel-ready-to-wear-spring-summer-2011-paris-...	400631	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-ready-to-wear-spring...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Ready-To-Wear-Spring-Summer-2011-Paris-...	chanel-ready-to-wear-spring-summer-2011-paris-...	336001	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-ready-to-wear-spring...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Ready-To-Wear-Spring-Summer-2011-Paris-...	chanel-ready-to-wear-spring-summer-2011-paris-...	300959	1706	2560	JPEG	RGB	chanel
https://door11.com/chanel-ready-to-wear-spring...	https://media.door11.com/wp-content/uploads/20...	NaN	Chanel-Ready-To-Wear-Spring-Summer-2011-Paris-...	chanel-ready-to-wear-spring-summer-2011-paris-...	474234	1706	2560	JPEG	RGB	chanel

Figure 3.1: An extract of a CSV file

Here is a breakdown of the columns in the DataFrame:

1. ‘collection_url‘: This column contains URLs related to a collection.
2. ‘image_url‘: This column contains URLs images related.
3. ‘image_name‘: This column contains the name of the image file.
4. ‘image_path‘: This column contains the relative path to the image file

within the collection or website directory.

5. ‘image_size‘: This column represents the size of the image in bytes.
6. ‘image_width‘: This column contains the width of the image in pixels.
7. ‘image_height‘: This column contains the height of the image in pixels.
8. ‘image_mode‘: This column might be used to store the color mode of the image, and in this sample, it is RGB (Red, Green, Blue).

The amassed data quantity was quite captivating. Potentially, we could have engaged with 132,153 images across 1,404 distinct brands. Regrettably, this proposition proved unattainable within the confines of this project. Primarily, the computational prowess required to effectively manage the substantial 30GB of photos loomed large. The device we employed, an M1 MacBook Air, was not originally designed for data science tasks. Furthermore, the vexing issue of class imbalance necessitated resolution. Among the 1,404 brands, image distribution was uneven due to disparate origins or staggered entries into the realm of ready-to-wear fashion. One approach to addressing this imbalance is oversampling the less-represented category. Regardless of the technique employed, generating novel synthetic photos entails increased gigabyte consumption. While undersampling could have been an option, it might have introduced its own set of complications. This decision hinges upon the weight of the selected sample in training the data, as indicated by the predicaments outlined. As a result, the chosen path led to concentrating on a handful of brands with the most substantial image count, due to these challenges. In the upcoming sections, we will explore the selection of the brands used and how their photos were preprocessed.

3.2 Data Statistic

The loaded data frame, as presented at the conclusion of *section 3.1*, contains the paths where the images are stored. Before proceeding to load images that might eventually go unused, we examined the brands with the highest photo counts, namely:

1. Valentino: 1140
2. Fendi: 940
3. Rick Owens: 880
4. Yohji Yamamoto: 880
5. Louis Vuitton: 880
6. Versace: 854
7. Giorgio Armani: 841
8. Hermes: 840
9. Dries Van Noten: 840
10. Chanel: 840

We opted to focus on these specific brands: ['Rick Owens', 'Dries Van Noten', 'Versace', 'Louis Vuitton', 'Chanel', 'Giorgio Armani']. This choice stems from various reasons, some more objective than others. Certainly, personal taste plays a significant role[1]. Another rationale behind this selection is the quantity of photos associated with each brand. These brands exhibit a well-balanced class distribution, contributing to a favorable data balance.

Compared to what was stated in *chapter 1*, the various artistic directions and designers who have contributed to the creation of collections are a very important factor. This is because each collaboration and director brings personal aspects to their work that differentiate their work from that of others. The selected brands generally maintain consistent artistic directors; however, every year and for some releases, they involve different collaborations with various artists and designers.

3.3 Images Preprocessing

According to A. Famili, Wei-Min Shen, Richard Weber, and Evangelos Simoudis; image data preprocessing refers to the procedures conducted on raw image data before its utilization in analyses or its input into machine learning algorithms. These authors in their paper "Data Preprocessing and Intelligent Data Analysis" [12] explain how this process encompasses multiple stages designed to improve the image data's quality and appropriateness for subsequent tasks. This step encompasses actions such as resizing, standardization, augmentation, and noise reduction, all aimed at guaranteeing that the data is in a fitting state for analysis or model training. Some of these steps are detailed in the next section.

3.3.1 Background removal

The first crucial transformation needed is removing the image's background, which means eliminating the surroundings from an image, leaving only the main subject, in this case, the model. Please note that the model we will develop later in this chapter is applied twice: once to the images with backgrounds and then to the images without backgrounds. The objective is to determine whether the presence of a background has any impact on the classification. By removing the background, the focus is shifted solely to the clothing items showcased on the runway. What we expect here, is that the model should be more focused on outfits. In other words; the fashion analysis could be more effective without distractions from the surrounding

[1] I have a strong affinity for Versace and the style of Rick Owens. Chanel and Giorgio Armani, in my view, are indispensable as well.

environment. Backgrounds in runway images can vary greatly from one photo to another, depending on the location of the show and the photographer's positioning. Removing the background creates a consistent and uniform presentation of clothing items, making it easier to compare different designs side by side.

To pursue this objective, we used the rembg library[1]. The rembg library has garnered significant attention from developers alike due to its straightforwardness and effectiveness in background removal. It finds applications in various domains, including graphic design, image editing, and computer vision tasks. This open-source library is continually maintained and updated by the community, ensuring that users can easily harness its capabilities for background removal.

This library use a deep learning architecture known as U2-Net. The author let some references, in particular, the paper "U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection" by Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, and Martin Jagersand from the University of Alberta, Canada [27]. Rembg library utilizes the U2-Net model, specifically designed for salient object detection, as a component of its background removal process.

The model starts by taking in the binary data of an image file. This data includes information about the picture's colors and other details. The U2-Net's design is like a set of nested layers, each with a specific job. These layers work together to understand the image better, focusing on both the tiny details and the big shapes. As the image goes through these layers, they look at it from different angles. This helps the model recognize objects of all sizes and complexities. The layers combine all the information they gather from different angles and levels. This way, they create a full picture of what's in the image, taking into account both local details and the overall scene. With all this combined information, the U2-Net gets a deep understanding of the image. It can spot the edges, shapes, textures, and other signs that show where important objects are. Using the information it has gathered, the U2-Net performs object segmentation. This means it creates a binary mask that distinguishes between the salient objects (foreground) and the less significant areas (background). Pixels corresponding to the salient objects are marked as "1", while pixels corresponding to the background are marked as "0". The binary mask generated by the U2-Net is the output of the model. This mask essentially highlights the regions of the image where the salient objects are located. At this point, the map can be used to put on

[1]You can find the GitHub repository for the library at the following link: https://github.com/danielgatis/rembg.

top of the original image to show where the important objects are, or it can be used to cut out the important objects from the background like rembg does. In *Figure 3.2* there are some results:

Figure 3.2: Background removal: some results

3.3.2 Transforming the Images

As already mentioned, data preprocessing involves transforming and preparing the data to be suitable for feeding into a machine-learning model. For the purpose of this project, the process applied to images concerns different steps. Look at the code below, in the *Figure 3.3*.

```
# Define the preprocess_image function to load, resize, and normalize the pixel values of the images.
def preprocess_image(image_path):
    image_path = './imgs/images/images/' + image_path
    image = load_img(image_path, target_size=(img_height, img_width))
    image = img_to_array(image)
    image = image / 255.0
    return image
```

Figure 3.3: First custom CNN with self-attention layer

Images are typically represented as tensor array in machine learning because they are efficient and easy to work with. Tensor arrays are multidimensional arrays that can store data of numerical type. This makes them ideal for storing image data, which is typically multidimensional. The function built for prepossessing performs

the normalization of the pixels. Normalization is the process of scaling the data to a specific range of values. In this case, the data is being scaled to a range of values between 0 and 1. Moreover normalizing the data can help to ensure that all of the features in the data are given equal weight. This can prevent the machine learning model from overfitting the training data. Second, normalization can help to improve the efficiency of machine learning models. We can reduce the amount of computation that is required by the machine learning model. This can be especially important for large datasets.

3.3.3 Data Augmentation

Data augmentation is a method employed in machine learning and data science to expand a dataset's size by implementing different alterations and adjustments to the original data. These modifications introduce variety and unpredictability into the dataset, potentially enhancing the effectiveness and resilience of machine learning models. Data augmentation methods are frequently utilized with images, text, and various other forms of data. The network built for this project has some layers born to make data augmentation. The following bullet points briefly explain them:

1. RandomFlip Layer: This layer applies random horizontal flipping to the input images, which helps the model learn from different orientations of the same objects.

2. andomRotation Layer: Here is another data augmentation technique, this layer applies random rotation to the input images. It rotates the images by a small angle, up to 0.1 radians, to expose the model to variations in object orientations. The output of the shape is: (None, 224, 224, 3)

3. RandomTranslation Layer: The last data augmentation layer is now run, which performs random horizontal translation on the input images. It moves the pixels in the image horizontally by up to 20This helps the model learn to recognize objects even when they are slightly off-center.

3.4 The model

The network we implemented has a backbone part and a custom part. In this section we are going into the customization made for this network. The full model structured is display is *figure 3.4*, and the focus of the attention mechanism is display in *Figure 3.5*. For the backbone part, we used a pretrained VGG16 model without the top classification layer, which is in TensorFlow. Note, that the top classification layer in a deep learning model, like VGG16, consists of the final layer that performs

predictions or classifications. This layer utilizes the high-level features acquired by earlier layers to classify input data into distinct categories or classes. By eliminating the top classification layer, the specific classification functionalities of the model get eliminated while preserving its feature extraction capabilities. This enables the addition of a new customized classification layer tailored to the particular task. Initially, this model was applied to the original images. Subsequently, the same model was applied to images with the background removed. This approach aims to explore potential variations in the model's performance under different conditions.

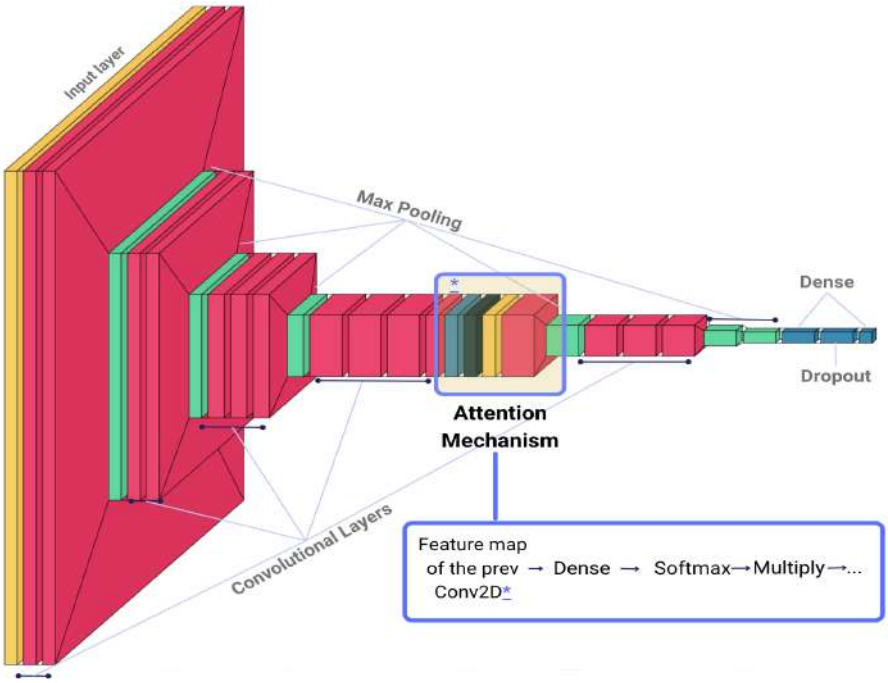

Figure 3.4: The structure of the Neural Netwrok implemented: backbone model (VGG16) and the custom part

The architecture consists of several layers and some of them were already explained in *section 2.2*. Here the full of layers:

1. Input Layer
2. 13 convolutional layers and pooling layers (backbone Model)
3. Attention Mechanism
4. Element-wise Multiplication with Attention Map
5. Remaining Feature Extraction Layers (backbone Model)
6. Global Max Pooling Layer (global_max_pooling2d_2)

7. Classification layers

Next, only some of the aforementioned layers will be explained. Element-wise Multiplication with Attention Map, the Remaining Feature Extraction Layers (Backbone Model) and the dropout layer explained in the next section. In the application we are discussing, their operation is not different from what has already been explained in *chapter 2*.

3.4.1 Attention Mechanism

In this section, we won't explain so much what attention is but rather where it is positioned within the network, how it is structured, and more generally, the steps it takes. For an introduction to the topic, please refer to *section 2.2.5*.

During the fine-tuning process, we considered the option of placing the attention mechanism at the penultimate position or in the sixth-to-last layer of the model architecture. The task at hand involved the decision of whether to position the attention mechanism closer to the input or closer to the output in the neural network architecture. It's crucial to acknowledge that this decision can be highly context-dependent, influenced by the particular requirements of the task and the chosen architectural design. However there are some shared best practices, for example, placing attention mechanisms closer to the input can help with large input data, such as high-resolution images, where not all parts of the input are equally important for the task. In this way the network focuses on relevant parts of the input data, effectively performing feature selection. Attention at the input stage allows the network to process and attend to salient information early in the forward pass, potentially reducing the computational cost of processing irrelevant information later in the network. On the other hand, placing attention mechanisms closer to the output allows the network to access the full context and make decisions based on a global view of the input data. Moreover; attention mechanisms can learn complex interactions between different parts of the input and output sequences, enabling the model to capture intricate patterns in the data. The attention mechanism is inserted before the sixth-to-last layer of the backbone model. Specifically, it's added as a layer called attention, after a convolutional layer. Look at the *Figure 3.5* to have reference for the attention mechanism structure.

- Convolutional Layer (conv2d_2), which applies a convolution operation to the feature maps from block4_conv3. The purpose of this convolutional operation is to generate attention scores for each spatial location in the feature maps, appling 512 filters. These attention scores are learned during training and represent the importance or relevance of each region in the feature maps.

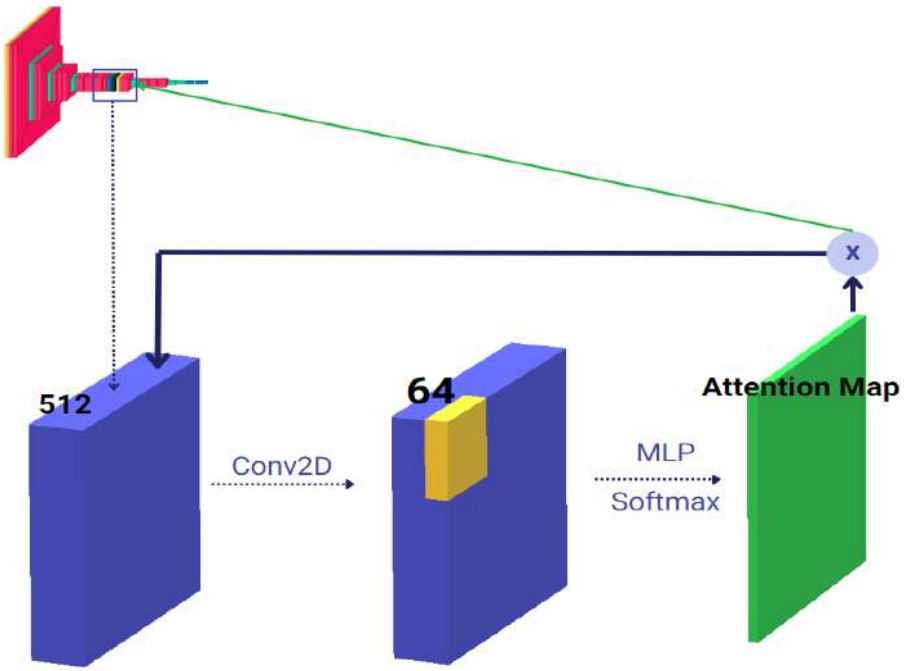

Figure 3.5: Scheme of how Attention Mechanism works. A stream which is part of *Figure 3.5*

- Dense Layer (dense_6): Following the convolutional layer, there is a single neuron Dense layer called dense_6. Importantly, this Dense layer does not have an activation function. The role of this Dense layer is to combine information from the convolutional layer and transform it into a format suitable for further processing. It helps in aggregating the attention scores produced by the convolutional layer.

- Softmax Activation (softmax_2): After the Dense layer, the model applies a softmax activation function, resulting in the layer named softmax_2. Before applying attention to the input, the parameter 'axis=[1, 2]', in the code, indicates that the Softmax computation is performed separately along the height and width axes of the image (i.e., along dimensions 1 and 2 of the tensor). The softmax activation normalizes the attention scores to create a probability distribution. This distribution assigns probabilities to each location in the feature maps, indicating how likely each location is to contain relevant information. The softmax ensures that the sum of the probabilities across all locations is equal to 1. Finally, the Multiply() operation is used to multiply the original input by the normalized attention map. This means that the calculated attention determines how much each pixel of the input image contributes to the final output of the network. The final output will be a weighted version of the

input, where pixels that receive higher attention in the attention map will have a greater impact on the output.

Basically the convolutional layer generated an attention map using the attention mechanism. This attention map assigns importance scores to different spatial locations within the feature maps extracted from a specific layer (block4_conv3). In this step, the model takes the attention map and performs element-wise multiplication with the feature map. Element-wise multiplication means that it multiplies each element (pixel) in the attention map by the corresponding element in the feature map. In other words, each value in the attention map is used as a weight to scale the corresponding value in the feature map. The result of this element-wise multiplication effectively emphasizes certain parts of the feature map while de-emphasizing others. When these high values are multiplied by the corresponding elements in the feature map, they have the effect of amplifying the information in those regions. In other words, the features in the high-attention areas are highlighted and become more influential in subsequent processing. Conversely, locations in the attention map with low values indicate regions that are less important. When multiplied with the feature map, they have the effect of reducing the influence of the information in those regions. Features in the low-attention areas are downplayed or suppressed. The purpose of this element-wise multiplication is to allow the model to dynamically adjust its focus on different parts of the feature map based on what it has learned during training. It's a mechanism for the model to selectively attend to relevant features and ignore less informative or noisy parts of the feature map.

In *Figure 3.6*, we can find an example of the output of the attention map, specifically for the model applied to both images with the background and those without. What we can notice at first glance is how the model seems to focus more on the outfit in the model without the background, whereas when the model is presented with an input with the background, it appears more distracted. *Figure 3.6* shows just an example of the product photos. However, we have made an effort to plot a sufficient number of photos, subjecting them to a qualitative analysis. What follows is that generally, the model without a background tends to focus better and with more intensity on photos without the background. Where by intensity, we mean that each pixel has a higher number.

After the element-wise multiplication step with the attention map, the model proceeds with the remaining Feature Extraction Layers of the backbone Model. Following the attention mechanism, the model resumes its feature extraction process using the remaining layers in the backbone model. These layers consist, again, of convolutional and pooling operations. It's essential to note that the weights of these

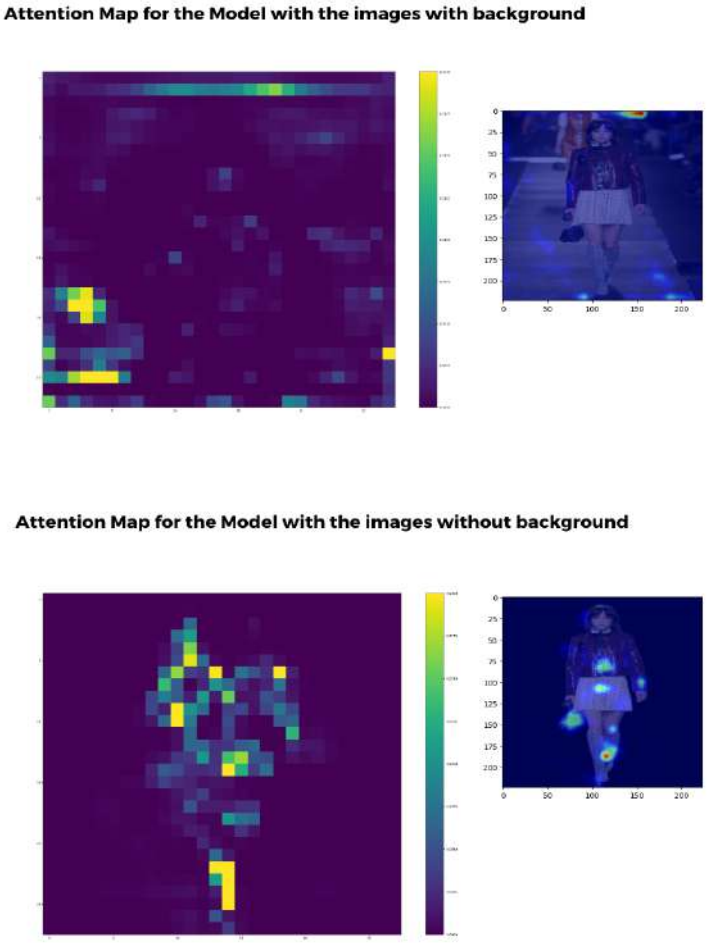

Figure 3.6: Visualizations of the Attention Maps for the same image using the two different models trained with (top) and without (bottom) the background

layers are frozen, meaning they are not updated or modified during the training process. This is a crucial aspect of transfer learning, where pre-trained models are used as a starting point. By keeping these weights fixed, the model retains the knowledge learned from a previous dataset (in this case, the ImageNet dataset) and adapts it to the current task. As the data flows through these layers, higher-level features are extracted, which are more abstract and semantically meaningful. These features capture increasingly complex patterns and information in the input images.

3.4.2 Classification

The classification part of our network refers to the component of a model that is responsible for assigning input data to specific categories or classes. Several layers are set to perform classification, the fist one is a Global Max Pooling. This layer reduces the spatial dimensions of the feature maps to a single vector. Global max pooling selects the maximum value from each feature map, resulting in a fixed-size

representation. The output of the Global Max Pooling Layer is a fixed-size representation, typically a one-dimensional vector. Each element in this vector corresponds to the maximum value from a specific feature map. The number of elements in the vector is equal to the number of feature maps. The resulting fixed-size representation is a condensed and informative summary of the feature maps, making it suitable for further processing in subsequent layers, which is a dense layer. Moving to the end of this network, there is another fully connected layer with 512 units and ReLU activation.

Finally the Dropout Layer, is a crucial component of neural networks, particularly for addressing the problem of overfitting. Overfitting occurs when a model learns to memorize the training data rather than generalize from it. Dropout helps mitigate this issue. In this model, a dropout rate of 20% is specified, meaning that approximately 20% of the input units are randomly deactivated during each training step. The key idea behind dropout is to introduce some level of randomness or uncertainty into the model during training. By randomly deactivating units, dropout prevents the network from relying too heavily on any specific set of neurons for making predictions. This, in turn, encourages the model to learn more robust and generalized representations of the data. Dropout acts as a form of regularization because it discourages the network from fitting the training data too closely. During inference or when making predictions, dropout is typically turned off, and all units are used, ensuring that the model's predictions are consistent and deterministic.

In this neural network architecture, the last layer is responsible for producing the final predictions or outputs. Here, there are 6 units, which correspond to the six different classes (brands).

The model architecture has a total of 15'275'463 parameters. Parameters in a neural network are the tunable values that the model learns during training to make accurate predictions. They include weights and biases associated with each layer and neuron in the network. Among these parameters, 560'775 parameters are trainable (weights that will be updated during training). Trainable parameters are weights and biases that the model updates and adjusts during the training process to minimize the loss and improve its performance on the given task. These parameters are typically found in the layers specific to the current task, such as the fully connected layers or output layers. The remaining 14'714'688 parameters are non-trainable (weights from the frozen backbone model). Non-trainable parameters are weights and biases that come from a pre-trained backbone model and are not updated during the training of the current model. These parameters capture knowledge learned from a previous dataset (e.g., ImageNet) and are transferred to the current

model through a process known as transfer learning. Non-trainable parameters are often found in the layers of the backbone model, and they help the model leverage existing knowledge to solve a new, related task.

3.5 Model Train

The first step in building a neural network model is defining its architecture. So, the model is created using the Keras functional API. Before compiling the model, we set a learning rate scheduler. This function is responsible for adjusting the learning rate during training. Here, it starts with an initial learning rate of 0.001 and reduces it by 10% after the first 100 epochs. This dynamic learning rate adjustment can help improve training stability and convergence.

Once the model architecture is defined and the learning rate scheduler is in place, the following step is to compile the model. As one training parameter, I've chosen the Adam optimizer, a popular optimization algorithm that adapts the learning rate during training to converge faster. Then, I've set the categorical cross-entropy loss function.

The training steps takes into account the train data, validation data, epochs, and batch size. The batch size specifies the number of training examples that are used together to compute one update of the model's parameters. In other words; when training a neural network, I don't typically update the model's parameters after every individual training example. Instead, I update the parameters based on a group of examples, known as a "batch". Batch training is a technique that allows to perform gradient descent and update the model's weights using multiple training examples at once. The noise in individual examples gradients can cancel each other out in a batch. Averaging gradients across a batch has a slight regularization effect, similar to adding a small amount of noise to the training process. This can help prevent overfitting. In this case, 64 training examples are used in each batch. Batch training helps manage memory and can speed up training through parallel processing on GPUs. Larger batch sizes require more memory to store the gradients and intermediate results for backpropagation. This can become a concern for large models and limited hardware resources. With larger batches, updates occur less frequently, which can slow down the training process. Smaller batches can update the model more frequently. Another crucial parameter to set is the number of epochs. An epoch is a complete pass through the entire training dataset. This parameter determines how many times the model will see and learn from the entire dataset. In this case, the model is trained for 100 epochs. The goal of using epochs is to improve the model's ability to generalize and make accurate predictions on new, unseen data. Training for more

epochs can potentially improve model performance, but there's a risk of overfitting if the model starts fitting noise in the training data. After a certain number of epochs, the model's performance might plateau. Training beyond this point might yield minimal improvements or even worsen performance due to overfitting. Multiple epochs allow the model to see the data from different perspectives, learning various patterns and relationships. This improves the model's ability to generalize and make accurate predictions on unseen data. Complex relationships in the data might require several epochs for the model to learn effectively. Longer training times enable the model to capture intricate patterns. As epochs progress, the model refines its weights to minimize the loss. This helps the model approach convergence, where the training loss stabilizes, indicating that the model has learned as much as it can from the data.

After training the identical model on two different sets of images—one set with backgrounds and another set without backgrounds (as described in section 3.1.2)—it's essential to visualize and compare the results using appropriate plots. This comparison will provide insights into the impact of including backgrounds on the model's performance.

3.6 Model Evaluation

Model evaluation is a crucial step in the machine learning and deep learning workflow. It involves assessing the performance and effectiveness of a trained machine-learning model using various metrics and techniques. The goal of model evaluation is to determine how well the model generalizes to unseen data and whether it meets the desired criteria for a specific task.

3.6.1 Validation and Training Accuracy

Model *with background*

The training accuracy starts from a relatively low value and gradually increases as the model learns from the training data. It exhibits a steady upward trend, indicating that the model is improving and fitting the training data better over time. The training accuracy reaches a high value, indicating that the model has learned the training data well. The validation accuracy follows the training accuracy. It exhibits an upward trend, indicating that the model is learning useful patterns from the training data that generalize to the validation data. The validation accuracy typically plateaus or fluctuates as the model encounters data it hasn't seen during training. A large gap between training and validation accuracy can indicate overfitting, where the model memorizes the training data but doesn't generalize well to

new data. In contrast, if both training and validation accuracies are low, it might indicate underfitting, suggesting that the model is too simple to capture the underlying patterns in the data. Here, a small gap between training and validation accuracy suggests that the model is not overfitting the training data excessively. It means that the model's performance on unseen data (validation data) is consistent with its performance on the training data, indicating that it has learned to generalize effectively.

Model *without background*

The training accuracy starts from a relatively low value, indicating that the model is initially performing poorly on the training data. As training progresses, there is a gradual improvement in both training and validation accuracy. This suggests that the model is learning and adapting to the data over time. Eventually, both training and validation accuracy values appear to converge to a stable range. This suggests that the model has learned the underlying patterns in the data to some extent and is not overfitting. The gap between training and validation accuracy remains relatively small until epoch 40. Validation accuracy reaches a point where it is around 0.70 and the training accuracy is around 0.90, this could indicate overfitting. In other words, while the model starts with a low accuracy, it gradually improves and converges to a stable range with consistent performance on both the training and validation datasets. However, the relatively large gap between training and validation accuracy suggests overfitting. However, further fine-tuning or architectural adjustments could potentially lead to even better performance.

3.6.2 Validation and Training Loss

Model *with background*

The training loss starts at a relatively high value (around 1.84) and gradually decreases over the initial epochs. *Figure 3.6* show clearly this trend indicates that the model is learning and improving its fit to the training data. As training progresses, both the training and validation loss continue to decrease, which is a positive sign. The model appears to be generalizing well to the validation data as the gap between training and validation loss is not widening significantly. Towards the end of the training, the training loss reaches a very low value (around 0.38), and the validation loss also decreases but at a slower rate, reaching a level of 0.69. This could be a sign that the model is starting to overfit the training data more.

Model *without background*

As we can see in *Figure 3.6* the training loss starts at a relatively high value

(around 1.905) and gradually decreases over the initial epochs. This indicates that the model is learning and improving its fit to the training data. The validation loss follows a similar trend initially but starts to increase after a certain point (around the 40th epoch). This divergence between training and validation loss suggests that the model might be overfitting the training data. Overfitting becomes more apparent in the latter part of training as the training loss continues to decrease while the validation loss increases or remains stable. This is a clear sign of overfitting. Towards the end of training, the training loss reaches a very low value (close to 0.3), but the validation loss remains relatively high (around 1.7 to 1.8), indicating a significant gap between training and validation loss.

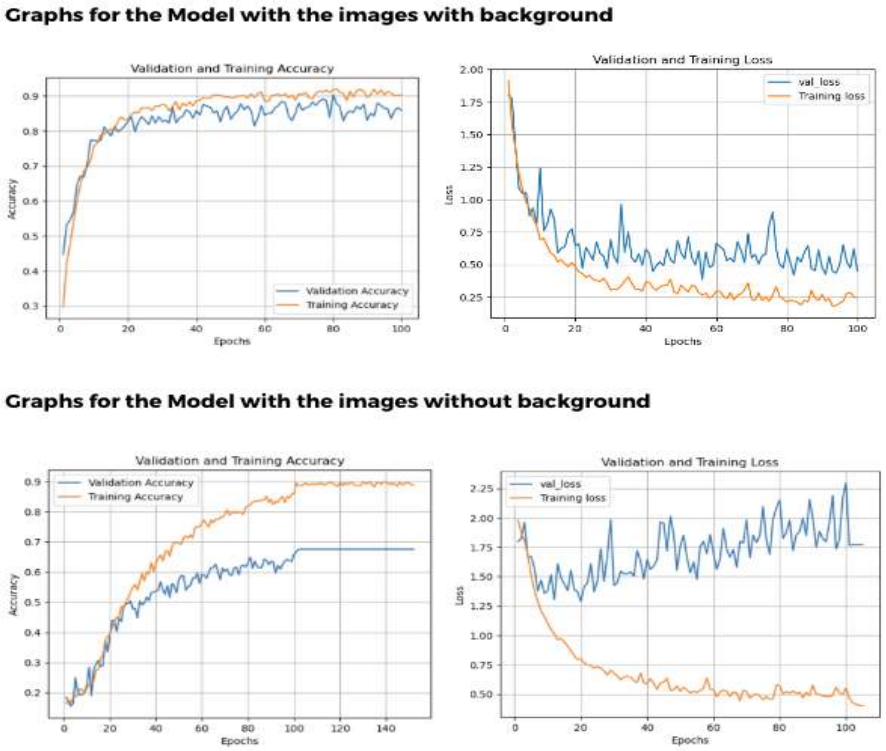

Figure 3.7: Validation-Training Accuracy and Validation-Training Loss for the model with the images with and without background

3.6.3 Confusion Matrix and other Classification Metrics

The confusion matrix provides a detailed breakdown of the model's performance in individual classes. Each row in the matrix represents the actual class, while each column represents the predicted class. This means that the diagonal elements (from top-left to bottom-right) represent the true positives, indicating the number of instances that were correctly classified for each class. Otherwise, off-diagonal elements represent misclassifications. The precision scores for each class vary, a metric I'll comment on later, measures the accuracy of the positive class predictions Then,

recall measures the ability of the model to correctly identify all positive instances. The F1-score is the harmonic mean of precision and recall and provides a balanced measure of model performance. Lastly, for each class, I'll compute the "support", that indicates the number of true instances for each class. Finally, I investigate if the model performs well across all classes, with slightly more weight given to the larger classes in the weighted average. That is done using the macro-average and weighted-average F1-score.

Model *with background*

In *table 3.1* you can see the confusion Matrix.

	rick owens	**dries van noten**	**versace**	**louis vuitton**	**chanel**	**giorgio armani**
rick owens	143	6	9	1	0	2
dries van noten	5	148	10	5	3	3
versace	16	4	156	5	3	5
louis vuitton	5	7	9	152	4	2
chanel	3	1	2	0	153	1
giorgio armani	5	2	18	1	8	130

Table 3.1: Confusion Matrix for the model with the images with background

For all the other classes look at *table 3.2*. The precision scores are relatively high, ranging from 0.76 to 0.93. This suggests that when the model predicts a fashion category, it is often correct. The recall scores are also generally high, ranging from 0.79 to 0.96, indicating that the model is effective at capturing most of the items in each category. The F1-scores for all classes are good, ranging from 0.79 to 0.92. Both macro and weighted averages of precision, recall, and F1-score are approximately 0.86, which demonstrates that the model's performance is consistent across different categories, and it is not biased toward any particular class.

The model appears to be capable of effectively distinguishing between different fashion items.

	Precision	Recall	F1-Score	Support
rick owens	0.88	0.68	0.77	161
dries van noten	0.61	0.66	0.64	174
versace	0.56	0.63	0.59	189
louis vuitton	0.78	0.63	0.70	179
chanel	0.73	0.71	0.72	160
giorgio armani	0.61	0.75	0.67	164
Accuracy			0.68	1027
Macro Avg:	0.70	0.68	0.68	1027
Weighted Avg:	0.69	0.68	0.68	1027

Table 3.2: Classification Metrics for the Model with the images with background

Model *without background*

In *table 3.3* you see the confusion Matrix.

	rick owens	**dries van noten**	**versace**	**louis vuitton**	**chanel**	**giorgio armani**
rick owens	110	10	24	6	7	4
dries van noten	2	115	12	5	12	28
versace	5	24	119	7	7	27
louis vuitton	2	17	31	113	10	6
chanel	5	9	9	9	114	14
giorgio armani	1	13	16	4	7	123

Table 3.3: Confusion Matrix for model with the images without background

Looking at the other metrics in *table 3.4*, the model's performance can be considered reasonably good, but there is room for improvement in certain areas. The precision scores for each class vary, with the highest precision (0.88) for "rick owens" and the lowest precision (0.56) for "versace". This indicates that the model is better at precisely identifying some classes than others. The recall scores also vary across classes, ranging from 0.63 to 0.75. Again, "rick owens" has the highest recall, while "versace" has the lowest. The F1-scores for different classes vary but generally fall between 0.59 and 0.77. The macro-average and weighted-average F1-scores are both around 0.68, which indicates that the model performs reasonably well across all classes, with slightly more weight given to the larger classes in the weighted average. The model seems to be decent to classify fashion outfits into different categories, but there are variations in its performance across classes. It performs better for some classes than others, and there is room for improvement, especially for classes with lower precision and recall. Further fine-tuning, data augmentation, or architectural adjustments may help improve its overall performance.

	Precision	Recall	F1-Score	Support
rick owens	0.81	0.89	0.85	161
dries van noten	0.88	0.85	0.87	174
versace	0.76	0.83	0.79	189
louis vuitton	0.93	0.85	0.89	179
chanel	0.89	0.96	0.92	160
giorgio armani	0.91	0.79	0.85	164
Accuracy			0.86	1027
Macro Avg:	0.86	0.86	0.86	1027
Weighted Avg:	0.86	0.86	0.86	1027

Table 3.4: Classification Metrics Model for the model with the images without background

3.6.4 Attention Mechanism Evalutation

After examining the attention maps generated by the model to see if they indeed focus on relevant regions, we notice that, generally, those generated by the model without the background tend to focus on much more salient points. The salient points of comparison once again come from a qualitative analysis. Clearly, a massive qualitative evaluation has been conducted, which required a significant amount of time to complete. However, we have also provided some numerical summary

A more comprehensive approach to understanding how attention behaves across different brands is achieved by visualizing the attention map, where we aggregate the attention scores from all the images. Looking at *3.7* we can see this result for the model applied to the images with the background. Again in *3.8* there are the results for the model applied to the images with the background.

The aim of these two representations is to investigate not only how attention has been allocated among different parts when comparing images with and without the background, but also to facilitate the comparison between classes in order to identify potential variations in attentional focus among the brands. When scrolling through the heatmap readings across brands within the same application, it appears that there isn't a significant difference in attention values for the model with the background. It's worth noting that for both *Figures 3.8 and 3.9* the values have been recalculated on a logarithmic scale to highlight those parts of the matrix with values very close to zero. As expected, attention on the images with the background has shifted significantly towards the margins, reducing the focus on the outfit itself. However, we still notice a faint silhouette in the center of the heatmap, indicating that despite the noise generated by the background, the model has paid attention to the runway models. Some brands show a greater focus on the background, such as Rick Owens and Louis Vuitton in *Figure 3.8*. Chanel and Giorgio Armani do

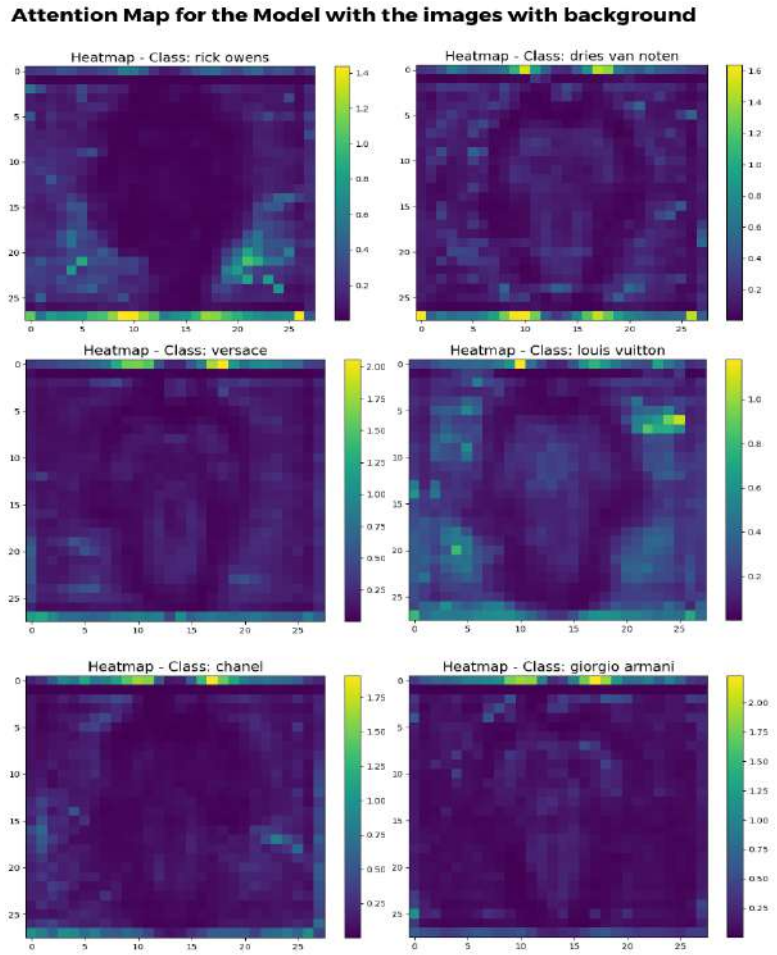

Figure 3.8: Model with the background: sum of all attention value by pixels for all the images divided by brand

not appear to have a strong focus on the human figure. This aspect would deserve further investigation in any potential development of this thesis.

Turning our attention to the results of the model applied to photos without the background, we can clearly deduce that the higher attention values are concentrated on the model. The silhouette of the model stands out distinctly. There are differences between brands, with noticeable variations. For example, in the case of Chanel in *Figure 3.9*, it appears to have generally lower attention, except for the lower part of the photo.

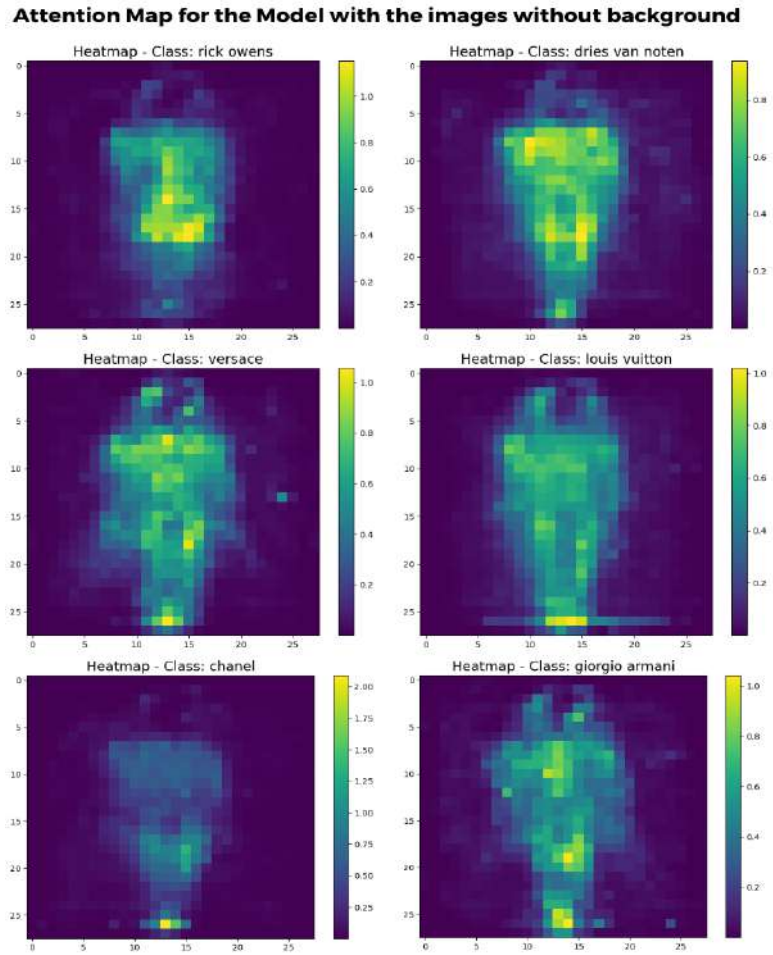

Figure 3.9: Model without the background: sum of all attention value by pixels for all the images divided by brand

Reviewing the aggregated heatmaps by brand, specifically those for Rick Owens, Chanel, and Versace, we have chosen significant photos that represent the key focal points where the attention layer has most frequently concentrated. For all the brands, photos from various collections spanning multiple years were selected. This serves to demonstrate how certain trends for some brands appear to remain unchanged. For example, looking at *Figure 3.10* we notice that for Chanel, the lower area appears to be significant. Indeed, Chanel often opts for above-the-knee skirts, leaving the legs exposed, while the shoes in all this have a contrasting color compared to the models, who appear to have fair skin. Again in *Figure 3.10*, it appears that Rick Owens also tends to favor concentrations of faux leather in the central part of the model's body. Generally, Rick Owens features models with diverse physiques and colors, which makes the evaluation of the outfit more heterogeneous when considering the model as well. For Versace, we found it interesting to evaluate the upper part of the outfit, although in *Figure 3.10* the heatmap appears to concentrate on other parts as well. In general, Versace seems to have some attention directed towards

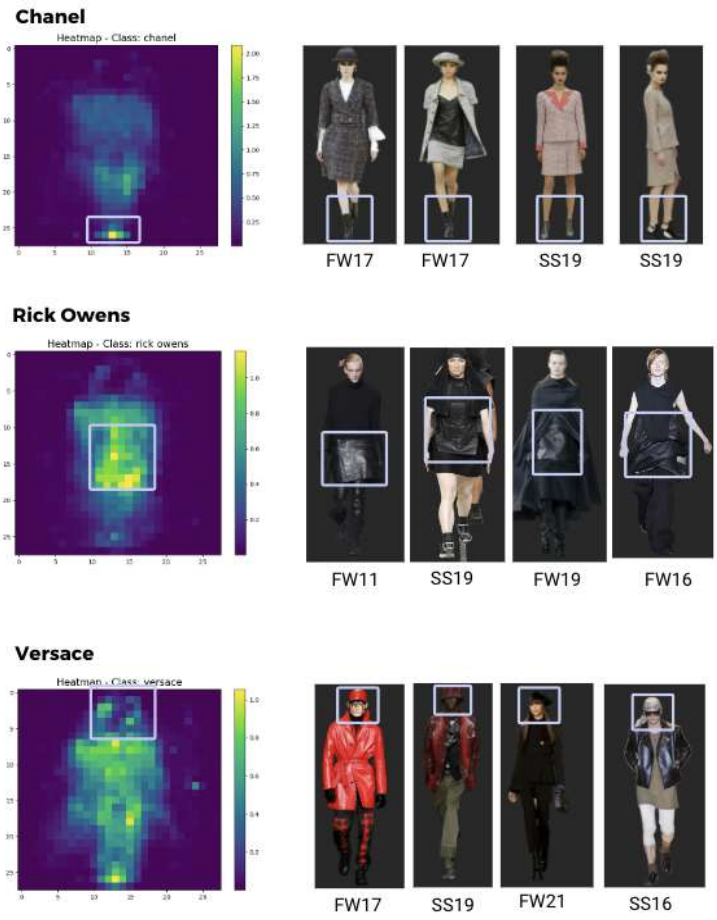

Figure 3.10: Model without the background: Contextualization of the sum of all attention value by pixels for some images divided by brand

the models' heads. This could be due to the hats worn by the models, which are present in nearly all collections, but also perhaps because of the models' hairstyles, which are styled in a distinctive manner. In contrast to Chanel, which also features many hats in its runway shows, those of Versace seem to be more prominent in the classification. Evaluating the highlights of heatmaps across hundreds of photos is a time-consuming and detail-oriented process. The provided images are representative examples of what has been analyzed, but it's important to note that this is only a small selection, and the complete process can be much more in-depth and detailed. Understanding the focal points in heatmaps can provide valuable insights into the attention and key elements of fashion collections.

For the sake of accuracy and completeness of results, we report below an example, in *Figure 3.11*, of poorly captured attention. However, the model doesn't always seem to focus well, not even in the model without the background.

Figure 3.11: A bad example of attention

In conclusion, our analysis of the attention maps generated by the model has revealed several interesting insights. Moreover this work has provided valuable insights into the model's attention distribution, offering implications for different brands and background conditions. This work contributes to the broader understanding of attention mechanisms in computer vision and highlights the importance of considering attention allocation in various practical applications

Chapter 4

Conclusion and Other Considerations

In the development of this project, we focused on creating a multiclassification model designed for brand classification. This challenge served as a significant test to evaluate the effectiveness of the Convolutional Neural Network (CNN) in identifying the strength of the visual identity of various brands. To make our CNN even more powerful, we implemented an attention layer, allowing the model to focus on crucial features in the images. We explore the complex structure of a neural network and gain understanding of potentiality.

The two models obtained have achieved satisfactory results, making them already usable in well-structured business contexts. On these two promising basic classification models, it would be interesting to carry out further optimization processes. The potential of these two models highlights the strength of the tools employed: CNNs and attention mechanisms. We have noted the promising capability of classifying images and extract intricate details and information. However, their implementation necessitates substantial data labeling efforts and computational resources, which represent the associated weaknesses to consider in their deployment.

4.1 Discussion

Both models, one trained with images containing backgrounds and the other without backgrounds, show promising performance in classifying fashion items into six different categories.

The model with the background appears to outperform the model without the

background in the classification task. The presence or absence of background in the images seems to have a noticeable impact on classification performance. The background in fashion images often contains valuable contextual information. For instance, the surroundings, scenes, or settings in which the fashion items are photographed can provide clues about the brand or style. The model with background takes advantage of this context to make more informed predictions. Having the background might help the model distinguish between similar-looking items from different brands. It can identify distinctive backgrounds associated with specific brands or fashion styles, contributing to better discrimination between classes. In contrast, the model without background faces challenges in cases where contextual cues from the background are crucial for classification. It relies solely on the appearance of the fashion item itself, which can be limiting, especially when multiple brands produce visually similar items. In other words, we can infer that in some cases, the brands may not be easily distinguishable solely based on the visual appearance of their fashion items. This could imply that certain brands might not have a distinct visual identity that sets them apart from others, at least in the context of the dataset and the features the model relies on. Some brands may produce fashion items with styles, designs, or aesthetics that are visually similar to those of other brands. This similarity in styles can make it challenging for a model to differentiate between them based solely on visual features. Look at *Figure 4.1*, to better understand the point.

Figure 4.1: Some classes (brand) which have similar outfits in different years and collection

Fashion brands often follow or adapt to current market trends, which can lead to a convergence of styles and designs across different brands. This can result in fashion items that look alike, further complicating brand identification. The visual features available in the dataset may not capture the finer details or nuances that

distinguish one brand from another. Certain brand-specific elements, such as logos, labels, or unique stitching patterns, might not be prominent in the images. As mentioned earlier, context, including background information, can be important in brand identification. If context is removed, the model loses valuable cues for distinguishing brands. Brand identity can sometimes be subjective and based on factors beyond visual appearance, such as brand reputation, values, and associations. These subjective aspects are not typically captured in image-based models. The point is, that while some fashion brands may have a strong and easily recognizable visual identity, others may share commonalities that make them less distinguishable in certain contexts. The ability of a model to identify brands accurately depends on the distinctiveness of visual features, dataset composition, and the importance of context in brand recognition.

4.2 Retrospective Consideration and Possible Improvement

The models can benefit from further fine-tuning by adjusting hyperparameters, learning rates, and optimization algorithms to address the potential overfitting issue indicated by the gap between training and validation performance. Fine-tuning can involve experimenting with various hyperparameters, including learning rates, dropout rates, and batch sizes, through multiple model runs. User feedback and domain experts' input are crucial for identifying model strengths and weaknesses, including areas of consistent misclassification or bias. Expanding the training dataset with augmented data, such as random rotations and scaling, can enhance generalization, though this was initially limited by machine capabilities. To mitigate overfitting in the model without backgrounds, techniques like dropout layers and L2 regularization can be added. Dropout introduces randomness during training, preventing overreliance on specific neurons, while L2 regularization encourages smaller weights to improve generalization. Experimenting with different model architectures, including layers and units, can also boost performance. More high-quality labeled data specific to fashion categories can be beneficial. Ensembling, combining predictions from both models, may enhance overall performance.

In conclusion, pursuing these avenues is essential to determine if improvements are achievable, considering the challenge posed by distinguishing certain fashion brands due to limited distinctiveness in style and visual identity.

Bibliography

[1] URL: https://www.ibm.com/topics/deep-learning.

[2] Abien Fred Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. arXiv: 1803.08375 [cs.NE].

[3] Charu C. Aggarwal. *Neural networks and deep learning: A textbook*. Springer, 2023.

[4] Richmond Alake. *Understanding and implementing lenet-5 CNN Architecture (Deep Learning)*. Mar. 2022. URL: https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342.

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[6] S.H. Shabbeer Basha et al. "Impact of fully connected layers on performance of convolutional neural networks for image classification". In: *Neurocomputing* 378 (2020), pp. 112–119. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.10.008. URL: https://www.sciencedirect.com/science/article/pii/S0925231219313803.

[7] Audrey Aurelia Chang et al. "Fashion Trend Forecasting Using Machine Learning Techniques: A Review". In: *Data Science and Intelligent Systems*. Ed. by Radek Silhavy, Petr Silhavy, and Zdenka Prokopova. Cham: Springer International Publishing, 2021, pp. 34–44. ISBN: 978-3-030-90321-3.

[8] François Chollet. *Deep learning with python*. Manning Publications Co., 2018.

[9] Keith Dinnie. "Luxury fashion branding — Trends, tactics, techniques". In: *Journal of Brand Management* 16 (2009), pp. 413–415. URL: https://api.semanticscholar.org/CorpusID:167879546.

[10] Sanket Doshi. *Various optimization algorithms for training neural network*. Aug. 2020. URL: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[11] Arun Kumar Dubey and Vanita Jain. "Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions". In: *Applications of Computing, Automation and Wireless Systems in Electrical Engineering*. Ed. by Sukumar Mishra, Yog Raj Sood, and Anuradha Tomar. Singapore: Springer Singapore, 2019, pp. 873–880. ISBN: 978-981-13-6772-4.

[12] A. Famili et al. "Data preprocessing and intelligent data analysis". In: *Intelligent Data Analysis* 1.1 (1997), pp. 3–23. ISSN: 1088-467X. DOI: https://doi.org/10.1016/S1088-467X(98)00007-9. URL: https://www.sciencedirect.com/science/article/pii/S1088467X98000079.

[13] Yanming Guo et al. "Deep learning for visual understanding: A review". In: *Neurocomputing* 187 (2016). Recent Developments on Deep Big Vision, pp. 27–48. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.09.116. URL: https://www.sciencedirect.com/science/article/pii/S0925231215017634.

[14] Gousia Habib and Shaima Qureshi. *Gapcnn with Hypar: Global average pooling convolutional neural network with novel NNLU activation function and hybrid parallelism*. Sept. 2022. URL: https://www.frontiersin.org/articles/10.3389/fncom.2022.1004988/full.

[15] Ali Hassanzadeh and Tara Namdar. *MILLENNIALS BRAND LOYALTY IN THE FASHION INDUSTRY & THE ROLE OF BRAND IDENTITY*.

[16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[17] Maya Krishnan. "Against Interpretability: A critical examination of the interpretability problem in machine learning". In: *Philosophy and Technology* 33.3 (2019), pp. 487–502. DOI: 10.1007/s13347-019-00372-9.

[18] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[19] T. S. Lee, C. S. Leung, and Zhiming Zhang. "Fashion Brand Image Marketing: Brand Image and Brand Personality". In: *Research journal of textile and apparel* 4 (2000), pp. 60–67. URL: https://api.semanticscholar.org/CorpusID:167956934.

[20] Anusha Lihala. *Attention and its different forms*. Mar. 2019. URL: https://towardsdatascience.com/attention-and-its-different-forms-7fc3674d14dc.

[21] N. E. Mills. "Intellectual Property Protection for Fashion Design: An Overview of Existing Law and a Look Toward Proposed Legislative Changes". In: *Shidler Journal of Law, Commerce & Technology* 5 (2009), p. 24. URL: https://digitalcommons.law.uw.edu/wjlta/vol5/iss5/5.

[22] Mayank Mishra. *Convolutional Neural Networks, explained*. Sept. 2020. URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[23] Derrick Mwiti. *Keras loss functions: Everything you need to know*. Sept. 2023. URL: https://neptune.ai/blog/keras-loss-functions.

[24] Anh-Duc Nguyen et al. "Distribution Padding in Convolutional Neural Networks". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 4275–4279. DOI: 10.1109/ICIP.2019.8803537.

[25] Keith B. Oldham, Jan C. Myland, and Jerome Spanier. "The Hyperbolic Tangent tanh(x) and Cotangent coth(x) Functions". In: *An Atlas of Functions: with Equator, the Atlas Function Calculator*. New York, NY: Springer US, 2009, pp. 289–296. ISBN: 978-0-387-48807-3. DOI: 10.1007/978-0-387-48807-3_31. URL: https://doi.org/10.1007/978-0-387-48807-3_31.

[26] Benny Prijono and Pkab Berkata. *Student notes: Convolutional Neural Networks (CNN) introduction*. Apr. 2018. URL: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/.

[27] Xuebin Qin et al. "U2-Net: Going deeper with nested U-structure for salient object detection". In: *Pattern Recognition* 106 (2020), p. 107404. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107404. URL: https://www.sciencedirect.com/science/article/pii/S0031320320302077.

[28] Rachid Riad et al. "Learning strides in convolutional neural networks". In: *CoRR* abs/2202.01653 (2022). arXiv: 2202.01653. URL: https://arxiv.org/abs/2202.01653.

[29] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV].

[30] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[31] Sumit Saha. *A comprehensive guide to Convolutional Neural Networks - the eli5 way*. Nov. 2022. URL: https://medium.com/towards-data-science/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[32] Sagar Sharma. *Activation functions in neural networks*. Nov. 2022. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[33] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. "ACTIVATION FUNCTIONS IN NEURAL NETWORKS". In: *International Journal of Engineering Applied Sciences and Technology* 04 (May 2020), pp. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[34] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *CoRR* abs/1803.02155 (2018). arXiv: 1803.02155. URL: http://arxiv.org/abs/1803.02155.

[35] K. Shibata and K. Ito. "Gauss-sigmoid neural network". In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*. Vol. 2. 1999, 1203–1208 vol.2. DOI: 10.1109/IJCNN.1999.831131.

[36] Shaghayegh Shirkhani et al. "Study of AI-Driven Fashion Recommender Systems". In: *SN Computer Science* 4.5 (2023). DOI: 10.1007/s42979-023-01932-9.

[37] Carlos Silva et al. "Cattle Brand Recognition using Convolutional Neural Network and Support Vector Machines". In: *IEEE Latin America Transactions* 15.2 (2017), pp. 310–316. DOI: 10.1109/TLA.2017.7854627.

[38] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[39] Jing Theng So, Andrew Grant Parsons, and Sheau Fen Yap. "Corporate branding, emotional attachment and brand loyalty: The case of luxury fashion branding". English. In: *Journal of Fashion Marketing and Management* 17.4 (2013), pp. 403–423. ISSN: 1361-2026. DOI: 10.1108/JFMM-03-2013-0032.

[40] Ayush Somani, Alexander Horsch, and Dilip K. Prasad. *Interpretability in deep learning*. Springer, 2023.

[41] Manli Sun et al. "Learning Pooling for Convolutional Neural Network". In: *Neurocomputing* 224 (2017), pp. 96–104. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.10.049. URL: https://www.sciencedirect.com/science/article/pii/S0925231216312905.

[42] Jerry Wei. *Alexnet: The architecture that challenged CNNS*. Sept. 2020. URL: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[43] Linchuan Xu et al. *A Novel Global Spatial Attention Mechanism in Convolutional Neural Network for Medical Image Classification*. 2020. arXiv: 2007.15897 [eess.IV].

[44] Sergey Zagoruyko and Nikos Komodakis. "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". In: *CoRR* abs/1612.03928 (2016). arXiv: 1612.03928. URL: http://arxiv.org/abs/1612.03928.

[45] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. arXiv: 1311.2901 [cs.CV].

[46] Xizhou Zhu et al. *An Empirical Study of Spatial Attention Mechanisms in Deep Networks*. 2019. arXiv: 1904.05873 [cs.CV].