# A DEEP NEURAL NETWORK BASED PRODUCT METADATA VALIDATION APPROACH FOR ONLINE MARKETPLACES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

ŞÜKRÜ ALATAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

APRIL 2023

Approval of the thesis:

A DEEP NEURAL NETWORK BASED PRODUCT METADATA VALIDATION APPROACH FOR ONLINE MARKETPLACES

Submitted by ŞÜKRÜ ALATAŞ in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç
Dean, **Graduate School of Informatics**

Prof. Dr. Altan Koçyiğit
Head of Department, **Information Systems**

Prof. Dr. Altan Koçyiğit
Supervisor, **Information Systems, METU**

**Examining Committee Members:**

Assoc. Prof. Dr. Pekin Erhan Eren
Information Systems, METU

Prof. Dr. Altan Koçyiğit
Information Systems, METU

Assist. Prof. Dr. Özgür S. Öğüz
Computer Engineering, İhsan Doğramacı Bilkent Uni.

**Date:** *19.04.2023*

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

|  |  |
|---|---|
| **Name, Last name :** | **Şükrü Alataş** |
| **Signature :** | ______________ |

## ABSTRACT

A DEEP NEURAL NETWORK BASED PRODUCT METADATA VALIDATION APPROACH FOR ONLINE MARKETPLACES

Alataş Şükrü

MSc., Department of Information Systems

Supervisor: Prof. Dr. Altan Koçyiğit

April 2023, 72 pages

As e-commerce has become increasingly popular during the pandemic, online marketplaces have seen a surge in merchants offering various products. A critical factor in the success of these marketplaces is the user experience they provide, largely dependent on features like efficient product search, fast filtering, and attractive product images. However, maintaining the data quality of product metadata and images can be challenging, especially as the number of products grows exponentially. To address this issue, this research proposes a novel approach using an AI-based automated image validation model for validating product images and an AI-based classification model to validate product metadata in an automated fashion. Our approach offers several advantages over traditional methods, including handling complex and noisy data and adapting to various challenging product categories, such as fashion items. We demonstrate the effectiveness of this approach through comparisons with traditional methods and in different settings, ultimately showing strong support for the use of AI in product metadata validation for online marketplaces.

Keywords: Metadata Validation, Deep Neural Networks, CNN, Online Marketplaces

# ÖZ

## ÇEVRİMİÇİ PAZARYERLERİ İÇİN DERİN SİNİR AĞLARINA DAYALI ÜRÜN ÜSTBİLGİSİ GEÇERLEME YAKLAŞIMI

Alataş Şükrü
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Prof. Dr. Altan Koçyiğit

Nisan 2023, 72 sayfa

Pandemi sırasında e-ticaret giderek daha popüler hale geldiğinden, çevrimiçi pazaryerlerinde çeşitli ürünler sunan satıcı sayılarında ciddi bir artış görüldü. Bu pazar yerlerinin başarısındaki kritik faktör, verimli ürün arama, hızlı filtreleme ve çekici ürün resimleri gibi özelliklere bağlı olarak sağladıkları kullanıcı deneyimidir. Ancak, özellikle ürün sayısının üstel arttığı bir ortamda, ürün üst verilerinin ve görüntülerinin veri kalitesini korumak pazar yerleri için zor olmaktadır. Bu sorunu ele almak için bu araştırma, ürün resimlerini doğrulamak için yapay zeka tabanlı otomatikleştirilmiş bir görüntü geçerleme modeli ile ürün üst verilerini otomatik bir şekilde doğrulamak için yine yapay zeka tabanlı bir sınıflandırma modeli kullanan yeni bir yaklaşım önermektedir. Yaklaşımımız, karmaşık ve gürültülü verileri işleme ve moda ürünleri gibi çeşitli zorlu ürün kategorilerine uyum sağlama dahil olmak üzere geleneksel yöntemlere göre çeşitli avantajlar sunmaktadır. Bu çalışmada, yaklaşımımızın etkinliğini, geleneksel yöntemlerle ve farklı ortamlarda karşılaştırmalar yaparak gösteriyor ve çevrimiçi pazar yerleri için ürün üst verileri doğrulamasında yapay zekanın kullanımına yönelik güçlü destek sağlıyoruz.

Anahtar Sözcükler: Üst Veri Geçerleme, Derin Sinir Ağları, CNN, Çevrimiçi Pazaryerleri

To My Family

## ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Altan Koçyiğit, for his guidance, support, and patience throughout the entire process of conducting this research and writing this thesis. His invaluable insights and constructive feedback have been invaluable to developing and completing this work.

I would also like to thank my wife, Banu Alataş, for her love, support, and understanding during the long and often stressful process of completing this thesis. Her constant encouragement and belief in me kept me motivated and inspired at every step of my academic journey.

I am also profoundly grateful to my parents, Ali and Ayşe Alataş, for their support throughout my journey with computers since the age of 14. Their belief in my abilities and constant encouragement in challenging situations has been crucial to my success.

Finally, I would like to thank my little boy, Ata Alataş, for bringing constant joy into my life, and keeping me awake at night to work on my thesis. His presence has been a constant source of motivation and joy for me.

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

# LIST OF ACRONYMS

| | |
|---|---|
| **3D** | Three Dimensions |
| **ABO** | Amazon Berkley Objects |
| **AI** | Artificial Intelligence |
| **CDN** | Content Delivery Network |
| **CMYK** | Cyan, Magenta, Yellow, Black |
| **CNN** | Convolutional Neural Network |
| **CV** | Computer Vision |
| **DNN** | Deep Neural Network |
| **GMM** | Gaussian Mixture Model |
| **HSL** | Hue, Saturation, Lightness/Luminance |
| **HSV** | Hue, Saturation, Value |
| **HTML** | Hyper Text Markup Language |
| **ILSVRC** | ImageNet Large Scale Visual Recognition Challenge |
| **ISO** | International Standards Organization |
| **JSON** | JavaScript Object Notation |
| **MMCQ** | Modified Median Color Quantization |
| **MRI** | Magnetic Resonance Imaging |
| **ReLU** | Rectified Linear Unit |
| **RGB** | Red, Green, Blue |
| **RMSProp** | Root Mean Square Propagation |
| **TPE** | Tree-structured Parzen Estimator |
| **UTF** | Unicode Transformation Format |
| **YUV** | Y - Luminance/Brightness, U - Blue Projection, V - Red Projection) |

# CHAPTER 1

# INTRODUCTION

In this chapter, we give a brief introduction about the problem we aim to solve. Then we explain our motivation, objectives, and the scope of the research. At the end of this chapter, we give the details about the organization of this document.

## 1.1. Motivation

Online marketplaces have thousands of merchants selling products on their websites. Some of them have a specific market to aim for, and some of them do not. For example, according to revenue, Wayfair.com is one of the top online marketplaces, specifically aiming at the home decoration and furniture market. Merchants willing to sell their products through Wayfair.com can sell only a certain set of home and home decorations products. On the other hand, Amazon.com is the biggest online marketplace according to revenue and has no specific market to aim for. Almost all the products can be sold on Amazon online. However, in both situations, whether an online marketplace aims at a specific market or not, there are thousands of categories and different sets of metadata attributes for every product defined. Some of these metadata have a specific set of options, like brand and color. Some of them are free text or numerical attributes, like dimensions, weight, etc. Additionally, almost every product has one or more images uploaded to a CDN system. These metadata and images are essential factors for a better customer experience and also a significant driver of the marketplace's success (Blanco et al., 2010; Kim & Lennon, 2008; Li et al., 2016; Zhao et al., 2009).

When customers are willing to buy a product that fits the concept in their mind, the search and filter facilitate the buying decision and play an essential role in their experience on the website. They want to find the best match to their search definitions. So, suppose a marketplace has a better metadata selection, and the customer wants to buy a product for the category. In that case, the customer can easily filter the products and find the best match. So, this improves the shopping experience of the customer very much. Suppose the filtered products have exciting, good-looking images and a better description. The customer is more likely willing to buy the product (Blanco et al., 2010; Kim & Lennon, 2008). This drives the marketplace to better sales figures and greater market share. On the other hand, if a marketplace would not have a good set of defined metadata and images for its products, customers cannot find the product quickly. They cannot make a buying decision easily. This results in lower revenue in the end.

Data quality is another aspect of the problem. Thus, the merchants usually upload the product metadata and images; Marketplaces use predefined product metadata and images for several products to improve the data quality and the user experience. They merge the product records from different merchants into one unified product record, and customers see one product record for a specific product. While the customer wants to see the product page, the merchants selling the same product are listed on the product page as a list. The price point is selected as the lowest between them. Otherwise, they have thousands of different metadata and images for the same product. Customers see the same product repeatedly in the search results and listing pages with different titles, descriptions, metadata, and images. This kind of experience hardens the marketplace usage for the customers who try to find the best match at a lower price point.

While marketplaces try to unify this metadata into their product databases and allow merchants to use this predefined product database, certain product categories cannot easily be unified from merchant to merchant. Fashion, clothing, and shoes are examples of these categories. Marketplaces have to rely on merchants' metadata and images for these categories. Every merchant designs these products differ from each other, so they cannot be unified.

Another aspect of the problem is different metadata standards for a product vary according to the marketplace. Every marketplace has a different set of metadata for a certain product category. For example, one marketplace may have eight predefined color names for an item of clothing, and the other has free text for color names. It is hard to align the metadata and data quality for the merchants selling products in more than one marketplace. Sometimes it is possible to convert the metadata if it can be converted programmatically, but sometimes, do not. Also, some of the metadata cannot be validated digitally. For example, the material of a product could not easily be extracted by using its images. For example, Wayfair.com is aiming home decoration market and has eight different material categories (100% Cotton, Turkish Cotton, Egyptian-Quality Cotton, Cotton Blend, Rayon from Bamboo, Terry Cloth, Linen, and Polyester) for towels, while Amazon.com have only two (Cotton and Microfiber). For both marketplaces, it is almost impossible to validate the material using images. Material metadata for some products may also be hard to validate physically because some materials can only be validated by experts or in the lab.

While data quality is a key factor for buying decisions, merchants do not always have well-written descriptions, correct attributes, or good-looking images. Even if they have good quality data, validating and converting this bulky data to different marketplaces is time-consuming. This is also a problem in the marketplaces; as mentioned before, the metadata quality is a key factor for a marketplace's success. Customers' shopping experience is highly tied to the data quality served in the marketplace. Marketplaces need to improve data quality by validating and correcting the metadata and images provided by the merchants, especially in categories that do not allow the use of unified metadata and images.

## 1.2. Objectives and Scope

The marketplaces have millions of products listed with tens of millions of metadata to validate and correct. However, they also do not have a validated ground truth that helps them train the models that can be used in the automated validation processes. In this research, we propose a new way to create a ground truth from the unrefined bulk dataset, which is our first objective. Our second objective in this research is to offer a new model to be used in the validation process that gets images of the products and validates the metadata.

Our research aims to address the problem defined in the previous section with the limitation of the data available in the field. Our proposed model needs product images, product metadata that can be defined as classes, and a human classifier that needs to classify metadata using certain rules and restrictions defined beforehand. The approach presumes the existence of these entities. Also, the performance of this study may vary according to the quality of input product images and the human subject.

## 1.3. Solution Perspective

This section focuses on providing potential solutions to the problem identified in the previous sections. These solutions are based on the findings from the literature review.

In the literature, a few pieces of research are found on product metadata/attributes validation. There are four related works in the literature. They can be separated into two groups:

The first group uses product descriptions to validate product metadata. They are using different methodologies to validate the proposed product metadata values. Although they use textual data, they are quite different approaches to the problem.

The second group of the solution uses CV and OCR techniques to understand the product images and extract some valuable information from the product images. This methodology is similar to our perspective by using product images. However, they use CV and OCR techniques, while we use CNN to identify and validate the product metadata.

Overall, these solutions aim to address the challenges identified in the problem statement, but they use different perspectives and techniques.

## 1.4. Organization

This thesis consists of five chapters and sections under them. This first chapter is the introductory chapter that gives brief information about the problem, the motivation behind the research, and the solution perspectives. The second chapter focuses on the background information about the tools and techniques used in the proposed methodology. This section also covers the related work proposed in the literature. The third chapter is about the proposed methodology. It covers all the details about the proposed solution. The fourth chapter covers the experiment we conducted to realize the methodology stated in the previous chapter. The fourth chapter consists of the results and the discussions about the experiment's results. The final chapter is the concluding chapter, which consists of a summary of the research results and future work.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

In this chapter, we give brief background information about the components of the solution space and address the previous related work in the literature.

## 2.1. Background

This section gives brief information about tools and techniques related to the solution space of the methodology proposed in the research.

### *2.1.1. Deep Neural Networks (DNNs)*

Deep neural networks (DNNs) are artificial intelligence (AI) algorithms that have gained significant attention in recent years due to their ability to achieve state-of-the-art performance in various applications. DNNs are composed of multiple layers of interconnected nodes, each processing input data and passing it on to the next layer for further processing (Goodfellow et al., 2016). This hierarchical structure allows DNNs to learn complex patterns and relationships in data, making them particularly effective for tasks such as classification, image recognition, natural language processing, and machine translation. One of the key characteristics of DNNs is their ability to learn and adapt to new data without the need for explicit programming. This is achieved through the use of backpropagation. This training algorithm adjusts the weights and biases of the network based on the error between the predicted output and the actual output. Over time, the network learns to make more accurate predictions, allowing it to generalize to new data.

Recent developments in DNNs have focused on increasing the size and complexity of the network to improve performance. These large-scale networks, known as deep learning networks, can have hundreds of layers and millions of parameters, allowing them to learn more complex patterns and relationships in data. However, these networks require significant computational resources and data to train, making them challenging to implement in practice. Despite these challenges, the potential benefits of DNNs are significant.

In the field of computer vision, DNNs have been used to achieve near-human levels of performance in tasks such as object recognition and image classification (LeCun et al., 1989; Voulodimos et al., 2018). In natural language processing, DNNs have been used to improve machine translation and enable the development of conversational AI systems. In healthcare, DNNs have been used to identify patterns in medical images and predict patient outcomes (Lundervold & Lundervold, 2019).

Overall, DNNs represent a promising approach to AI that has the potential to improve the performance of a wide range of applications. While challenges remain in terms of scalability and practical implementation, ongoing research and development in this area are likely to continue to drive advancements in the capabilities of DNNs.

### *2.1.2. Convolutional Neural Networks (CNNs)*

Convolutional neural networks (CNNs) are a type of deep neural network specifically designed to work with one or more dimensional input data. One of CNN's most well-known sub-type is using 2D input data such as images. CNNs are composed of a series of interconnected layers, each of which performs a specific function in processing the input data (LeCun et al., 1998).

At a high level, CNNs are composed of multiple layers of interconnected nodes known as neurons. These neurons receive input from the previous layer, process it using a non-linear activation function, and pass the output to the next layer. The first layer of a CNN receives input from the raw data, which in the case of image analysis, would be the image's pixel values. The first layer in a CNN is the input layer, which receives the raw input data, followed by a series of convolutional layers, which apply a series of filters to the input data, extracting features and patterns from it (LeCun et al., 1998). The network learns these filters based on the input data and the desired output during the training process. These convolutional layers are followed by a series of pooling layers, which down-sample the output of the convolutional layers, reducing the dimensionality of the data and making it easier for the network to process. This may be followed by another convolutional layer – pooling layer pair or one or more fully-connected layers, which use the output of the pooling layers to make a final decision or prediction.

One of the key advantages of CNNs is their ability to learn features from the input data rather than requiring them to be hand-crafted by the user. This allows CNNs to be applied to various tasks, such as image classification, object detection, and segmentation. Another advantage of CNNs is their ability to exploit the spatial structure of the input data, allowing them to make highly efficient use of the available computational resources. This allows CNNs to be applied to tasks involving large amounts of data, such as video analysis and medical imaging. Despite these advantages, CNNs also have some limitations, such as their reliance on large amounts of training data and the potential for overfitting if the network is not properly regularized.

CNNs are a powerful tool for image and video analysis and have demonstrated their effectiveness in various tasks. Their ability to extract features from raw data and their efficient use of computational resources make them a promising approach for solving complex real-world problems.

### *2.1.3. Color Spaces*

Color spaces are a fundamental concept in the field of color science. They are used to represent and manipulate colors in various applications. These spaces provide a standardized way to represent colors in a form that computer systems can understand and process.

One of the most commonly used color spaces is the RGB (Red, Green, Blue) color space, which is based on the human visual system and how it processes colors. Another popular color space is the CMYK (Cyan, Magenta, Yellow, Key/Black) color space, which is used in printing applications. This color space represents colors as a combination of the four subtractive primary colors. A value between 0 and 100 represents each color channel. This color space is designed to produce consistent color reproduction on printed materials. However, it is not suitable for display on electronic devices.

In addition to these popular color spaces, there are also a variety of other color spaces that have been developed for specific applications, such as the Lab color space for accurate color reproduction, the HSL (Hue, Saturation, Lightness) color space for intuitive color manipulation, HSV (Hue, Saturation, Value) and the YUV (Y – Luminance/Brightness, U – Blue Projection, V – Red Projection) color space for video compression (Schwarz et al., 1987).

#### *2.1.3.1.RGB Color Space*

The RGB color space is a three-dimensional model (see Figure 1 RGB color space in 3D representation) in which colors are created by mixing red, green, and blue light (Joblove & Greenberg, 1978). This color space is based on the primary colors of light, red, green, and blue, and it is used to create a wide range of colors on digital displays, such as TVs, computers, and smartphones.

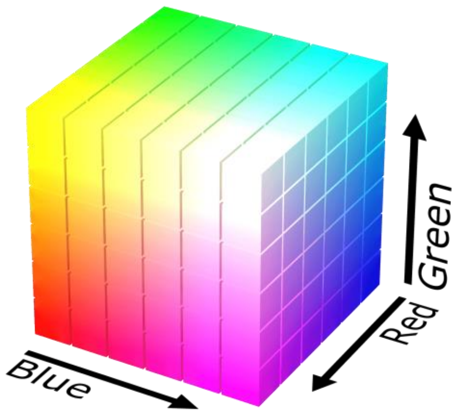


Figure 1 RGB color space in 3D representation[1]

The RGB color space is also known as the "additive" color space because the primary colors are combined to create different colors. For example, when the red and green light is mixed, they create yellow light. When red and blue light is mixed, they create magenta light. And when green and blue light is mixed, they create cyan light.

One of the key benefits of the RGB color space is that it is intuitive and easy to understand. Most people are familiar with the primary colors of light, and the concept of mixing colors is something that many people are familiar with from childhood. This makes it easy for people to use the RGB color space to create and manipulate colors on digital devices. Another benefit of the RGB color space is that digital devices and software widely support it. Virtually all digital displays and software programs that deal with color support the RGB color space, which makes it a convenient and versatile choice for many applications.

Despite these benefits, the RGB color space does have some limitations. For example, the range of colors produced in the RGB color space is relatively limited compared to other color spaces, such as the CMYK color space. This means that some colors, such as very saturated greens or purples, may be difficult to produce accurately in the RGB color space.

Additionally, the RGB color space is not well suited for printing applications. Because the RGB color space is based on the primary colors of light, the colors produced are not the same as those produced using the primary colors of pigments, such as cyan, magenta,

[1] (*RGB Color Model - Wikipedia*, 2022)

yellow, and black. This means that colors that are created in the RGB color space may not match the colors that are produced when they are printed on paper.

### *2.1.3.2.HSV Color Space*

The HSV color space, also known as the HSB color space, is a cylindrical model of color that separates hue, saturation, and value into three dimensions. This color space is often used in image editing and computer graphics due to its intuitive representation of color and its ability to easily manipulate the saturation and value of a color (Joblove & Greenberg, 1978).

The hue of color in the HSV color space is represented by a number from 0 to 360, with 0 being red, 120 being green, and 240 being blue. This angle around the color wheel allows for easy identification and manipulation of the primary and secondary colors. The saturation of a color in the HSV color space is represented by a percentage, with 0% being a shade of gray and 100% being the purest and most vibrant form of the hue. This allows for the adjustment of the vibrancy and intensity of a color. The value of a color in the HSV color space is represented by a percentage, with 0% being black and 100% being white. This allows for the adjustment of the lightness or darkness of a color (Figure 2).

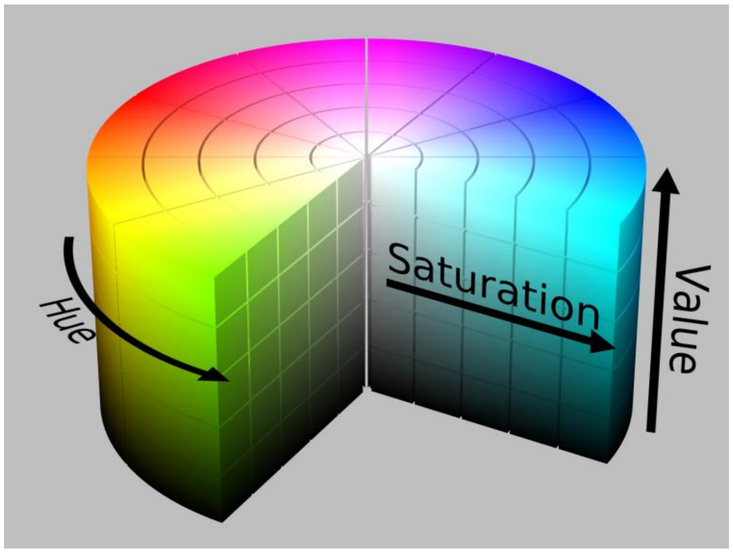


Figure 2 HSV color space in cylindrical representation[2]

Chroma and saturation are terms often used interchangeably, but they have slightly different meanings. Chroma refers to the intensity or vividness of a color. In contrast, saturation refers to the purity or richness of color. In other words, chroma measures how

[2] (*HSL and HSV - Wikipedia*, 2022)

bright or intense a color is, while saturation measures how pure or intense a color is. For example, a color with high chroma will be very bright and vivid, while a color with high saturation will be pure and rich in tone.

When we use chroma (calculated as Saturation * Value) instead of saturation, the resulting shape is a circular cone in three-dimensional space. This shape is more suitable for calculating distances between two HSV colors in Euclidean space. Because, the cylindrical representation of the HSV color space has a plane instead of a vertex for black. The circular cone representation has a point/vertex instead of a plane for the color black (Figure 3)(*HSL and HSV - Wikipedia*, 2022).

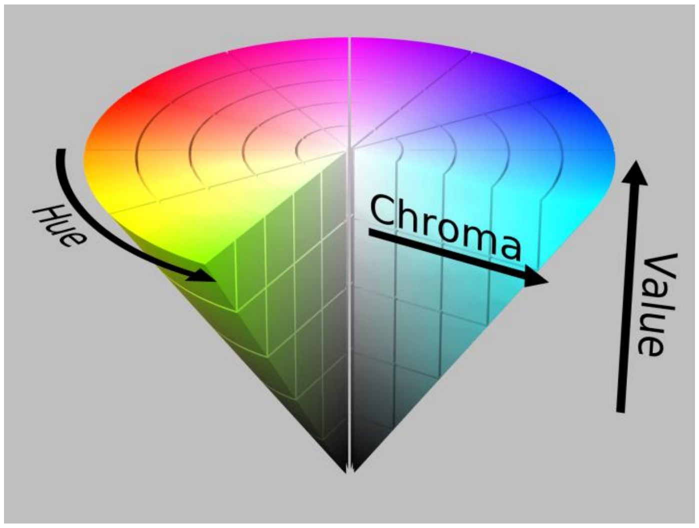


Figure 3 HSV color space in circular cone representation[3]

One advantage of the HSV color space is its ability to preserve the relationships between colors when manipulating saturation and value. In other color spaces, adjusting the saturation or value can result in a shift in the hue of a color. However, in the HSV color space, the hue remains unchanged. Another advantage is its ability to easily create complementary colors by adjusting the hue by 180 degrees. This allows for quick and easy color balancing in designs and artwork.

### *2.1.3.3.HSL Color Space*

The HSL color space, also known as the Hue-Saturation-Lightness color space, is a model used to represent colors more intuitively and perceptually uniformly than other color spaces. At its core, the HSL color space is based on the three primary colors: red, green, and blue, like RGB color space. In the HSL color space, colors are represented by hue,

[3] (*HSL and HSV - Wikipedia*, 2022)

saturation, and lightness. The hue is the basic color, red, green, or blue. The saturation represents the intensity or purity of the color, with higher saturation indicating a more intense color and lower saturation indicating a paler color. Finally, the lightness represents the brightness of the color, with higher lightness indicating a brighter color and lower lightness indicating a darker color.

While the hue value can be an angle between 0 and 360 degrees, Saturation and Lightness values can take values between 0 and 1. HSL color space can be formed as a cylinder (Figure 4) (Joblove & Greenberg, 1978).

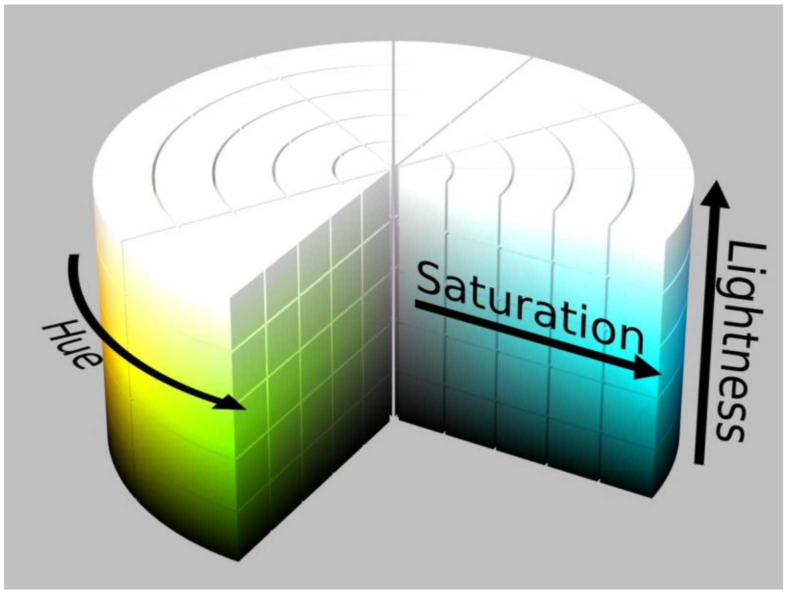


Figure 4 HSL color space in cylindrical representation[4]

When we plot hue and lightness against chroma instead of saturation, likewise HSV color space, the resulting model becomes a bi-cone that is a three-dimensional geometric shape with a circular base and two parallel, conical sides that meet at a point on top which is more suitable for calculating distances between two HSL colors in Euclidean space (Joblove & Greenberg, 1978). Because, the cylindrical representation of the HSL color space has two planes (black and white) at the top and bottom instead of a vertex. These planes have different distance values instead of one, which is misleading (Figure 4). The chroma can be calculated after the color is converted from HSL color space to HSV color space:

$$Value = L + S * \min(L, 1 - L)$$

[4] (*HSL and HSV - Wikipedia*, 2022)

$$Saturation' = \begin{cases} 0 & Value = 0 \\ 2\left(1 - \frac{L}{Value}\right) & Value \neq 0 \end{cases}$$

$$Chroma = Saturation' * Value$$

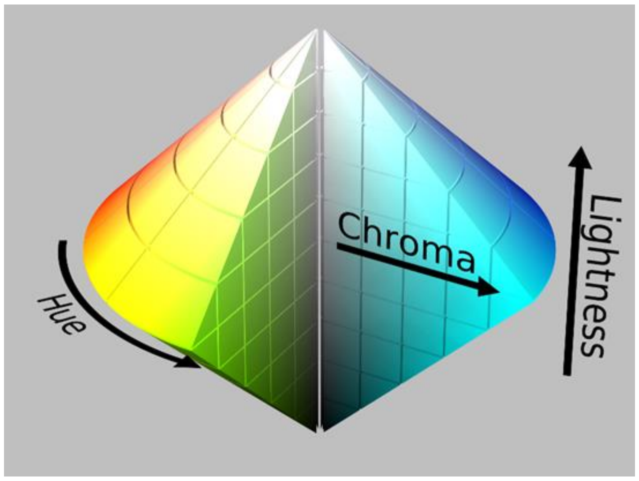


Figure 5 HSL color space in circular bi-cone representation[5]

One of the key advantages of the HSL color space is its perceptually uniform nature. This means that changes in hue, saturation, and lightness values will have the same visual impact regardless of the starting color. For example, increasing the lightness of a color by 50% will always result in the same visual change, regardless of whether the starting color is red, green, or blue.

### *2.1.4. Distance Algorithms*

Distance algorithms are a set of mathematical calculations that are used to determine the distance between two points. This distance algorithm is commonly used in various fields, such as geometry, physics, and engineering. It is also used in machine learning algorithms for data clustering and classification.

One of the most commonly used distance algorithms is the Euclidean distance, named after the Greek mathematician Euclid. The Euclidean distance is a fundamental concept in mathematics and geometry. It measures the straight-line distance between two points

[5] (*HSL and HSV - Wikipedia*, 2022)

in a Euclidean space. It is a mathematical space in which coordinates and distances represent points calculated using the Pythagorean theorem.

In addition to the Euclidean distance, there are other distance algorithms, such as the Manhattan distance and the Minkowski distance. The Manhattan distance is calculated by taking the sum of the absolute differences between the coordinates, while the Minkowski distance is a generalized version of the Euclidean and Manhattan distances.

These distance algorithms have different applications and properties. They are chosen based on the specific requirements of the problem at hand. For instance, the Euclidean distance is suitable for problems involving continuous variables, while the Manhattan distance is more suitable for problems involving discrete variables.

*2.1.4.1.Euclidean Distance for RGB Color Space*

The Euclidean distance between two points in a 3D environment, denoted by d(x, y), is calculated by taking the square root of the sum of the squares of the differences between the coordinates of the three points. For example, suppose we have two points in a cubic, A, and B, with coordinates (x1, y1, z1) and (x2, y2, z2), respectively. In that case, the Euclidean distance between them is:

$$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

The Euclidean distance is a useful measure in many applications. For example, it is used to calculate the distance of two colors between two colors in RGB color space. While Red, Green, and Blue represent a color in RGB color space, we can calculate the distance in two colors with the help of the Euclidean distance formula:

$$d(A,B) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$

As a result, when the range for each color is presumed between 0 to 255, the maximum distance in the RGB color space can be calculated as the distance between black and white, which is ~ 441.6729.

*2.1.4.2.Euclidean Distance for HSV and HSL Color Space*

The Euclidean distance for HSV and HSL Color Spaces measures the distance between two points in a circular cone / bi-cone shape. This measure uses the Hue and Chroma of the two colors and the Euclidean distance formula (*How Can I Calculate Distance from Two Pixels HSV?*, 2022).

To calculate the distance, we first need to find the chroma (C) with the help of Saturation (S) and Value (V) for the HSV color space. When using HSL color space, we need to first convert the HSL color to HSV with the formula defined in the previous section; then, we can calculate the chroma.

After calculating the chroma, then we can find the x value. y value and the z value of the coordinates of the colors with the help of the Hue (H) angle, as shown in the formula below.

$$\begin{aligned} C &= V * S \\ x &= C * \cos(H) \\ y &= C * \sin(H) \\ z &= V \end{aligned}$$

Once we have the coordinates, we can apply the Euclidean Distance formula between two points to find the distance between the two colors in the HSL color space.

$$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

When we apply the conversion on the Euclidean distance formula, the final formula is shown in the formula for HSV and HSL Color Spaces.

$$d_{HSV}(A,B) = \sqrt{(V_1 - V_2)^2 + S_1^2V_1^2 + S_2^2V_2^2 - 2S_1S_2V_1V_2\cos(H_1 - H_2)}$$

$$d_{HSL}(A,B) = \sqrt{(L_1 - L_2)^2 + S_1^2V_1^2 + S_2^2V_2^2 - 2S_1S_2V_1V_2\cos(H_1 - H_2)}$$

### *2.1.5. Quantization*

In image processing, quantization refers to reducing the number of bits used to represent the color information of an image. This is typically done to compress the image, allowing for more efficient storage and transmission of the data (Orchard et al., 1991).

There are several different approaches to quantization, each with its advantages and disadvantages. One commonly used approach is uniform quantization, where possible color values are divided into equal intervals. This approach is simple to implement and can provide good results for images with smooth color gradients. Another approach is non-uniform quantization, where the range of possible color values is divided into intervals of varying sizes. This approach can provide better results for images with complex color patterns. However, it is more computationally intensive and can result in larger file sizes.

One major limitation of quantization is that it can cause a loss of image quality, resulting in visible distortion or artifacts in the processed image. To mitigate this, advanced techniques such as dithering and error diffusion can be used to distribute the quantization error across the image in a more visually pleasing manner.

### *2.1.6. CV and Image Processing*

Computer vision is a rapidly growing field that has numerous applications in a wide range of industries. It involves using advanced algorithms and techniques to enable computers to interpret and understand visual data from the world around them (Vandoni, 1996).

The origins of computer vision can be traced back to the 1950s and 1960s, when researchers began to explore the potential of using computers to analyze and interpret visual data. One of the early pioneers in this field was David Marr, who developed the first computational vision model (Marr, 1982).

Since then, the field of computer vision has continued to evolve and expand, driven by advances in technology, algorithms, and hardware. Some of the key developments in this domain have included the development of image recognition algorithms, using deep learning techniques, and using high-performance computing resources.

One area of research in computer vision focuses on object recognition. This involves developing algorithms to identify and classify objects in an image or video based on their visual characteristics. For example, a computer vision system might be trained to recognize different types of vehicles, animals, or household objects.

Another important area of research in computer vision is image segmentation. This involves dividing an image into separate regions based on its visual characteristics, such as color, texture, or shape. This can help to identify and isolate specific objects or regions of interest within an image.

Today, computer-vision technology is used in many applications, including surveillance, medical imaging, and self-driving cars. In surveillance, computer-vision systems are being used to monitor public spaces, detect suspicious behavior, and alert authorities in real-time. In medical imaging, computer-vision systems analyze medical images, such as X-rays and MRIs, and identify abnormalities or conditions, such as cancer (Lundervold &

Lundervold, 2019). Furthermore, in self-driving cars, computer-vision systems are being used to provide the vehicle with the ability to perceive and navigate its environment.

Despite the many advances that have been made in computer-vision technology, there are still significant challenges and obstacles that must be overcome in order for this field to continue to grow and thrive. Some key challenges in this domain include more sophisticated algorithms, more efficient and scalable hardware, and better integration with other AI technologies.

At the same time, many opportunities and potential benefits come with the development of computer-vision technology. Some of these include the potential to improve safety and security, the potential to improve the accuracy and efficiency of medical diagnoses, and the potential to enable new forms of automation and transportation.

## 2.2. Related Work

In this section, we are elaborating on similar works that aim to address the challenges we stated as the problem statement. While lots of different research has been conducted for product metadata extraction and product recommendation (Deng et al., 2022; Ghani et al., 2006; Ghosh et al., 2023; Kumar & Saladi, 2022a, 2022b; Lin et al., 2021; Miami & Zeng, 2015; Petrovski & Bizer, 2017; Qiu et al., 2015; Rezk et al., 2019; Q. Wang et al., 2020a, 2020b; Wong et al., 2008; Yang et al., 2022a, 2022b; Zhang et al., 2022; Zheng et al., 2018), there are very few works conducted in the product metadata validation area. We have searched the literature and found two groups of similar work; the first group of research uses textual descriptions of the product to validate the product metadata. The second group uses product images with CV and OCR techniques to validate the product metadata. These studies are summarized in the following part:

**CAVE: Correcting Attribute Values in E-commerce Profiles**

This study is one of the recent papers about this area, published in October 2022. In this paper, the authors propose a system named CAVE (Correcting Attribute Values in E-commerce) for product metadata validation (Sabeh et al., 2022). This paper aims to validate, correct, and enrich product metadata using the Question Answering (QA) models, including BERT, DistilBERT, RoBERTa, ALBERT, and XLNET. CAVE is trained on datasets generated from the Amazon Review Dataset.

The proposed model learns information from titles and product attributes tables placed in the product description's HTML, using encoder and language models. Then, they use these data to validate and correct attribute values. The model also can enrich existing product descriptions with new attribute values extracted from titles. The work was presented at the 31st ACM International Conference on Information & Knowledge Management Conference (CIKM '22). The published conference proceeding does not have any evaluation for the proposed model. The overview of the proposed architecture is shown in Figure 6.

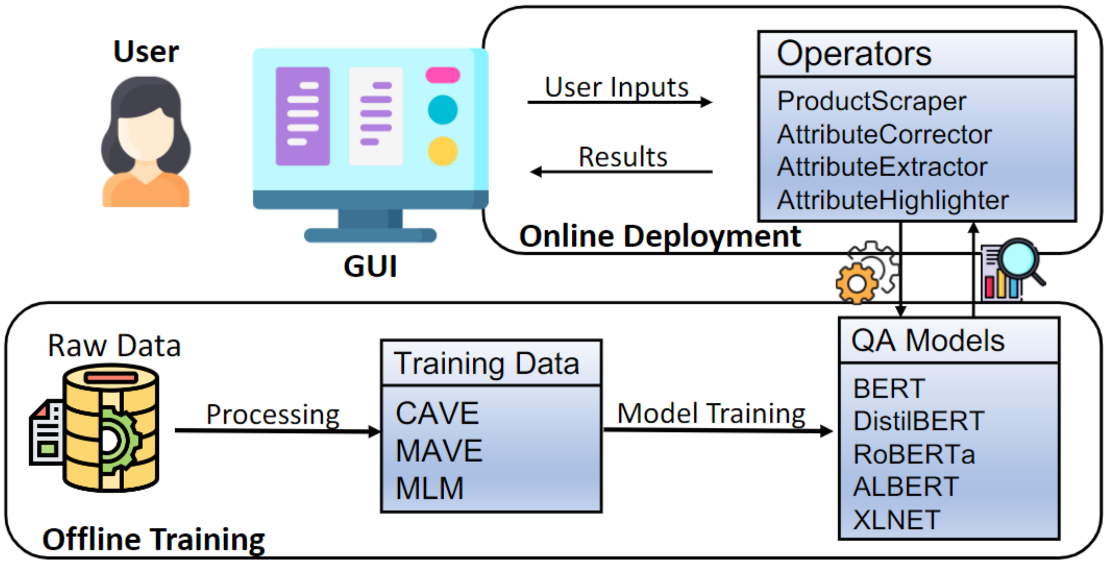


Figure 6 CAVE model architecture

The proposed architecture process Amazon Review Dataset and extract product metadata question-answer data from it. Then, it trains the QA models with these training data. After that, the user sends the product information via GUI. The model scrapes the product attributes from the HTML table in the product description and the product title from the product data and corrects them using QA models.

**E-commerce Product Attribute Value Validation and Correction Based on Transformers**

A transformer-based approach was proposed in this research to validate product attribute values using the product profile and suggest correct values when errors are detected (le Yu Haozheng Tian & Velkoski, 2022).

This approach can be applied to all textual metadata types. It uses a RoBERTa-based Natural Language Inference (NLI) model that has been extended for e-commerce product metadata value validation by comparing structured product information to the most relevant content selected from unstructured product profiles. In addition, the model is also used to recommend correct values. This feature reduces manual effort in real-life scenarios. As a result, the model has achieved PRAUC scores of 0.889 for contradictory, 0.864 for neutral, and 0.950 for entailed examples in a total of 8247 examples testing data.

**Automatic Validation of Textual Attribute Values in E-commerce Catalog by Learning with Limited Labeled Data**

Authors propose to develop an automatic validation approach that verifies the correctness of textual attribute values for products using product descriptions (Y. Wang et al., 2020).

They have proposed a meta-learning latent variable approach called MetaBridge to validate the textual attribute values of products from various categories as an NFI task in the few-shot learning setting. This approach involves a meta-learning latent variable model that simultaneously processes signals obtained from product profiles and textual attribute values. According to them, this model can significantly reduce annotation costs by effectively utilizing labeled data from limited categories. Regarding the PRAUC score, their model MetaBridge showed a 3.66% improvement over Meta-SGD and 3.33% compared to BERT.

### An Automated Computer Vision System for Extraction of Retail Food Product Metadata

An automation method was proposed to enhance the extraction of unstructured product metadata from food product label images using computer vision, machine learning, optical character recognition, and natural language processing (Gundimeda et al., 2019).

The overall process flow is shown in Figure 7. The process starts with background removal that uses computer vision similar to our baseline methods. Then, the automatic image quality classification step uses image data to extract product metadata with OCR combining other techniques. The output of this step is processed with NLP in the attribute extraction step. After that, the product metadata is validated with the extracted data against existing metadata.

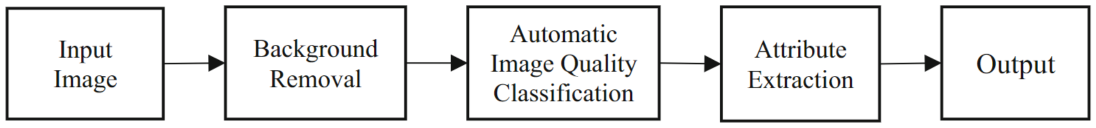


Figure 7 The process flow for the automated system for extraction of retail food product metadata

This approach also includes an automatic image quality classification system to identify images and a technique to improve the quality of images using traditional computer vision algorithms to enhance text detection and OCR and NLP-based metadata extraction accuracy.

To assess the performance of their proposed solution, the authors conducted an experiment using a real dataset with 352 food products from 53 brands, which contained 955 images (including front, back, and side view product images). After the experiment, it was found that their approach had a 0.9810 accuracy score on nutrition metadata validation and 0.9879 on net weight/volume validation.

# CHAPTER 3

# METHODOLOGY

In this chapter, we elaborate on our proposed methodology in detail. First, we give the problem definition and explain the research questions. Then, we expound the methodology into two main concepts, and we explain these concepts at every step.

## 3.1. Problem Definition

Improving the data quality of product metadata is a time-consuming job for the merchants and also for the marketplaces. The first reason for that is the count of the products. The marketplaces have millions of different products to be sold in their databases. These results in tens of millions of metadata attributes plus millions of product images to validate and correct. It is a costly decision to make this validation by the labor workforce. As a remediation, marketplaces try to develop internal automation tools to improve the data quality of the metadata and validate product images. These tools get input data from the uploaded information by the merchants and try to validate the metadata. For example, a tool can determine whether a product is placed in the wrong category with image verification. It gathers and processes all the images previously served in a category and finds out that the newly uploaded product is in the wrong category. Alternatively, a tool can easily correct grammar mistakes in a product description uploaded by a merchant.

These tools primarily bear the data already in the marketplace, uploaded by merchants, like product images. The data quality of the product images is a common problem for these tools used to improve the metadata. Merchants can upload images shot in improper lighting or angle. Products can be placed in the frame improperly (Figure 8).

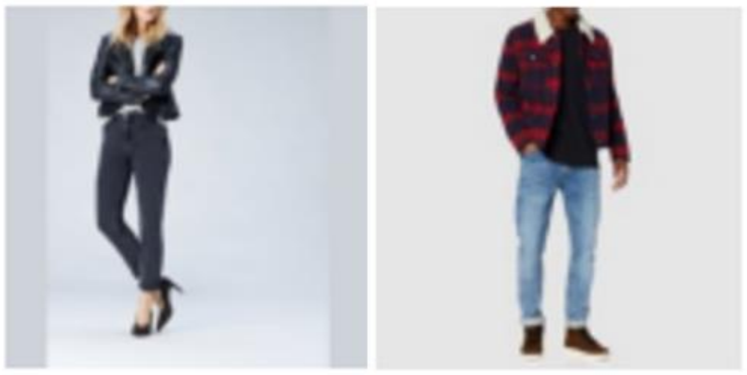


Figure 8 Improper shoe product image samples

Also, some of the product images created by the merchants include improper graphical elements like logos, campaign badges, size charts, or different kinds of text (product description, company contact information, etc.). These problems allow the internal tools to output incorrect results. So, these tools can easily miss details or make a wrong decision. For example, some of the images already served by the marketplace may have some defects, like containing size charts, shots from an improper angle, etc., that cause the tool to miss the defects on the newly uploaded images. Alternatively, the tool can correct all the grammar mistakes in the description. However, it may not handle the wrong description of the product or misleads the customers.

The last problem is human error and human subjectivity. Even if a marketplace has enough human resources to handle the validation of every product in its database, people can easily miss a detail on the product and outputs a worse-quality in data. Also, some metadata attributes have a vague description by nature. For example, color is one of the top filter data for clothing products. It defines the main visual conception of the product and provides guidance to the buying decision.

Despite this, color is a very subjective matter and has many different naming and classifications. For example, while RGB color space has more than 16 million colors, many marketplaces try to classify these colors into 8 to 10 color classes. In this circumstance, one person may classify a clothing product to one color; one other is to another. Some product categories also have different visual conceptions by its nature. For example, while shoe category has all the problems mentioned before.

Additionally, the conception of the color of a shoe varies from the other fashion/clothing categories. A small bow can determine the shoe's color on the top, or the primary color of a shoe varies from person to person who focuses on a different part of the shoe. These examples are shown in Figure 9. The shoe on the left is considered a white shoe by the customers of the small white bow. Different personnel may classify the shoe on the right as black, white, red, or multi-color.

Figure 9 Shoe product image samples

As stated before, the marketplaces have millions of products and tens of millions of product metadata and images uploaded. In terms of validation and correcting, the labor workforce cannot handle this bulk data manually. They need to automate this process. But when using automation, there are data quality-related problems with uploaded images and metadata. The product images can contain unrelated graphical items, improper posing, or other quality-related problems that mislead the tools. Also, input metadata contains human error and subjectivity. Suppose a classification model is trained with this data. In that case, the resulting model may suffer from these problems, and the results may not show the actual accuracy. This situation reveals a paradox: the tools created for validation and correction also need validated input data to work properly. As a result, It is harder to serve correct metadata for every product in marketplaces with millions of products inside. This directly affects customer experience and the success of a marketplace.

In this research, we aim to offer a new methodology to validate and correct metadata attributes of products by using their images, even if the metadata and the product images are not of good quality.

## 3.2. Methodology

We aim to offer a new model in this research to lessen the aforementioned problems. First, we use machine learning models to automatize the process. It is crucial to handle a large amount of bulk data. Second, we offer to create a ground truth with manual sampling and validation. This dataset is used to compare our classification model's output. We also create an image validation model with the sampling output and manual validation step. We use this model to validate raw images after. So that we can remove unrelated images from the dataset with the help of this model, and finally, we can use validated bulk images to create a classifier to improve the metadata. And we compare the results of this model with the run made with ground truth.

Our methodology has two main steps: validation model preparation and classification. In the first step, the ground truth is prepared by sampling and manual validation. Then image validation model is trained with the outputs (Figure 10).

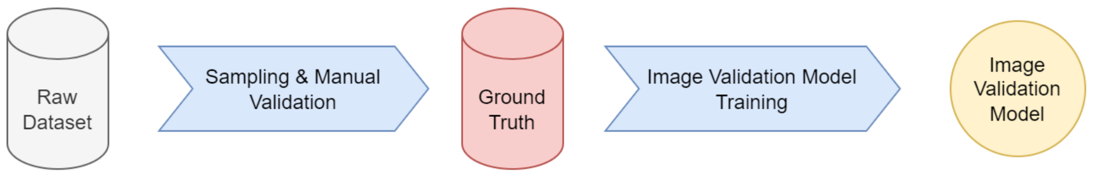


Figure 10 Methodology First step overview

In the second step, the artifact of the first step is used with the raw dataset to create a validated dataset in the beginning. Then metadata classification model is used on ground truth and validated dataset separately. In the final, the two results of the model are compared (Figure 11).

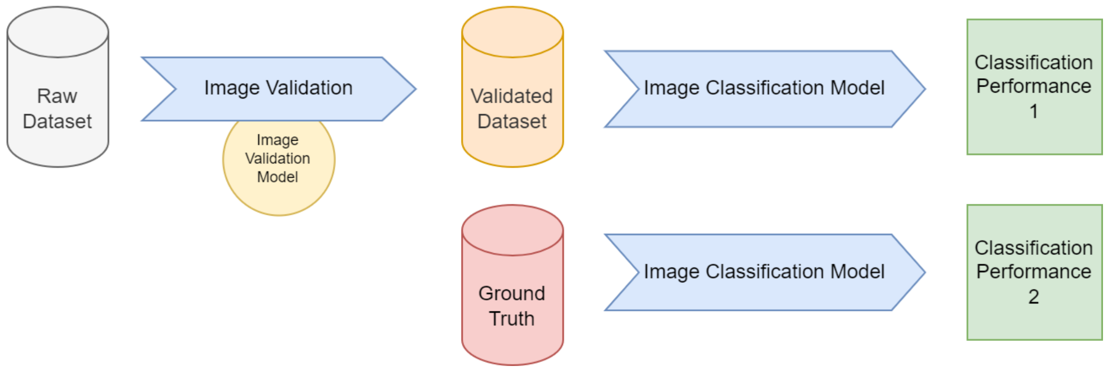

Raw Dataset
Image Validation
Image Validation Model
Validated Dataset
Image Classification Model
Classification Performance 1
Ground Truth
Image Classification Model
Classification Performance 2

Figure 11 Methodology Second step overview

### 3.2.1. *Sampling and Manual Validation*

In this step, we aim to create a ground truth dataset with sampling and manual validation. This is a manual process conducted by an expert. In the beginning, the number of the target image is determined. This number can change according to metadata and the properties of the validation model to be created after the selection. We offer to select an equal number of images for each class of metadata. This improves the resulting validation model accuracy and decreases the model's bias. If we select more images for a class, it may create a bias in this class in the resulting model. Also, the resulting model may miss the details of the other classes more.

Then, a set of rules must be defined preliminarily before the selection begins. These rules are used to define the properties of a valid image to be selected and the class of the image. These rules help us to decrease the subjectivity of the human workforce. Every selected image and the class of the image needs to comply with these rules.

After that, the sampling stage begins. We offer random sampling to avoid bias. First, a set of images was randomly selected from the raw dataset. The first image is analyzed against the rules defined. If the image is valid according to the rules, then the class selection is made according to the rules. The resulting image and the class are recorded. If an image cannot comply with the rules, then this image is also recorded as an invalid image to be used in the validation model to be created later. The predefined class and the resulting classes are also compared to calculate the accuracy of the predefined classes. The sampling stage is finished when the selected images have reached the predefined number.

### *3.2.2. Image Validation Model Training*

In this step, we use valid and invalid image samples from the previous step to create a validation model. This model validates input images and removes the invalid images from the dataset. This helps us decrease the effect of invalid images, including additional graphical items, improper posing, angle, etc. As a result, the classification model gets only valid input images. We do not add the misclassified images into the process in this step. These images are only used to find the ratio of the misclassification of the raw dataset.

### *3.2.3. Image Validation*

In this step, we use the model created in the previous step to validate bulk raw image data. This process classifies input images as valid or invalid. Then we remove invalid images to clear the dataset and improve the data quality. As a result, a dataset that has validated images has resulted in this step.

### *3.2.4. Image Classification Model Training*

In this step, we use images to train a classification model to classify products against selected metadata. But in the first step, we train the model with a ground truth dataset created in the early stages of the process. We record the resulting metrics of the model. Then, we train the model again with the predefined classes. After that, we compare the results with the ground truth model results. In the comparison, we use the misclassification statistics from the previous stage and the ground truth classification metrics to understand the efficiency of our classification model in terms of statistics.

# CHAPTER 4

# EXPERIMENTAL WORK

In this chapter, we give the details about the experiment we performed to realize the effectiveness of the model we proposed in the previous chapter. At the beginning of the chapter, we elaborate on the dataset and the features we used in the experiment. Then we give details about the steps performed in the experiment, implementation details, and experimental settings. And finally, we explain the results and findings after the experiment.

## 4.1. Dataset

In this experiment, we use the Amazon Berkeley Objects (ABO)[6] dataset, a large-scale dataset designed to train 3D models of products with real images (Collins et al., 2021). ABO dataset contains product catalog images, metadata, and 3D models of products sold on Amazon websites worldwide. The ABO dataset contains 147,702 products and 398,212 product images, along with the metadata of the products.

The metadata of the products is placed into 16 files encoded with UTF-8 and gzip-compressed. Every line corresponds to a product as a JSON object. Every product metadata JSON has common sub-sections listed in Table 1.

[6] https://amazon-berkeley-objects.s3.amazonaws.com/index.html

Table 1 ABO dataset metadata table

| Metadata Section | Description |
|---|---|
| brand | Brand name |
| bullet_point | Important features of the products |
| color | Color of the product as text (available in different languages) |
| color_code | Color of the product as HTML color code |
| country | Country of the marketplace, as an ISO_3166-1 alpha-2 code |
| domain_name | The domain name of the marketplace where the product is found. |
| fabric_type | Description of product fabric |
| finish_type | Description of product finish |
| item_dimensions | Dimensions of the product (height, width, length) |
| item_id | The product reference id. |
| item_keywords | Keywords for the product |
| item_name | The product name |
| item_shape | Description of the product shape |
| item_weight | The product weight |
| main_image_id | The main product images. |
| marketplace | Retail website name (Amazon, AmazonFresh, AmazonGo) |
| material | Description of the product material |
| model_name | Model name |
| model_number | Model number |
| model_year | Model year |
| node | Location of the product in the category tree |
| other_image_id | Other available images for the product |
| pattern | Product pattern |
| product_description | Product description as HTML |
| product_type | Product type (category) |
| spin_id | Reference to the 360º View image sequence. |
| style | Style of the product |
| 3dmodel_id | Reference to the 3d model of the product. |

There are 576 different types of products listed in the dataset. The top 20 product types according to the count of the item listed in (Table 2)

Table 2 ABO dataset product type distribution

| Product Type | Count |
|---|---|
| CELLULAR_PHONE_CASE | 64853 |
| SHOES | 12965 |
| GROCERY | 6546 |
| HOME | 5264 |

Table 2 (cont.) ABO dataset product type distribution

| Product Type | Count |
|---|---|
| HOME_BED_AND_BATH | 3082 |
| HOME_FURNITURE_AND_DECOR | 2255 |
| CHAIR | 2100 |
| BOOT | 2009 |
| SANDAL | 1845 |
| FINERING | 1540 |
| HEALTH_PERSONAL_CARE | 1449 |
| FINENECKLACEBRACELETANKLET | 1377 |
| ACCESSORY | 1362 |
| SOFA | 1199 |
| OFFICE_PRODUCTS | 1152 |
| FINEEARRING | 1137 |
| PET_SUPPLIES | 1064 |
| SPORTING_GOODS | 972 |
| TABLE | 935 |
| HARDWARE_HANDLE | 860 |

Each product has one or more images in various sizes. Also, some of the products have 3d models and images taken from different angles.

## 4.2. Product and Metadata Selection

The preliminary step in the experiment is the selection of the product category and selecting the metadata attribute to improve. It needs to have two characteristics: selected metadata could be extracted from the images of the product. The second one is it needs to be metadata that can be represented as classes.

The dataset has 576 different kinds of products and their metadata attributes. We select the "SHOES" category and the "color" metadata to improve. Shoes have the characteristics and the problems we defined in the previous chapter, that is, a fashion product that is hard to merge with other products and has a different visual perception from other fashion products, and every shoe has various forms, sizes, and concepts. Especially the color of the shoes does not always mean the main or dense color of the product. Usually, the color of a shoe is defined by a part of the shoe. So, it is hard to find the color of shoes by primitive techniques like using the color histogram. For example, the shoes shown in (Figure 12) have an intense white color spectrum, while it is pink shoes.

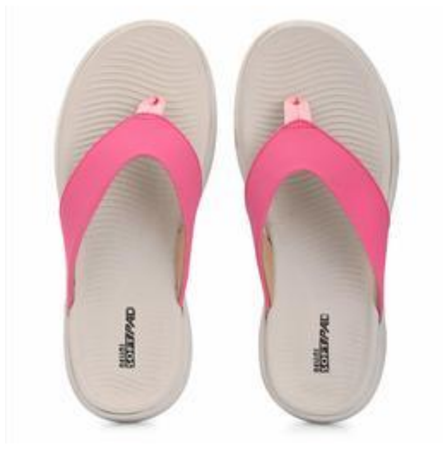


Figure 12 Sample shoe image in pink color

The color attribute, in general, also suffers from human subjectivity more than other metadata attributes. It is hard to classify some colors close to the primary colors' transition points. Visual perception of a color may vary according to light, posing, and other factors in the product image.

## 4.3. Experiment Design

In this section, we detail the experiment step by step. There are two main concepts in our experiment. The first step is the dataset processing steps using different kinds of tools and techniques. These steps are marked as blue in Figure 13. Every step results in a new dataset marked as orange in Figure 13. The second concept is the metadata improvement model runs on every resulting dataset, marked as green in Figure 13. On every new run for improvement, we use image data to improve the selected metadata with the help of CNN. We also use baseline algorithms to compare the results of our proposed model. We also compare the results of those runs to see the differences between datasets resulting from the processing phases.

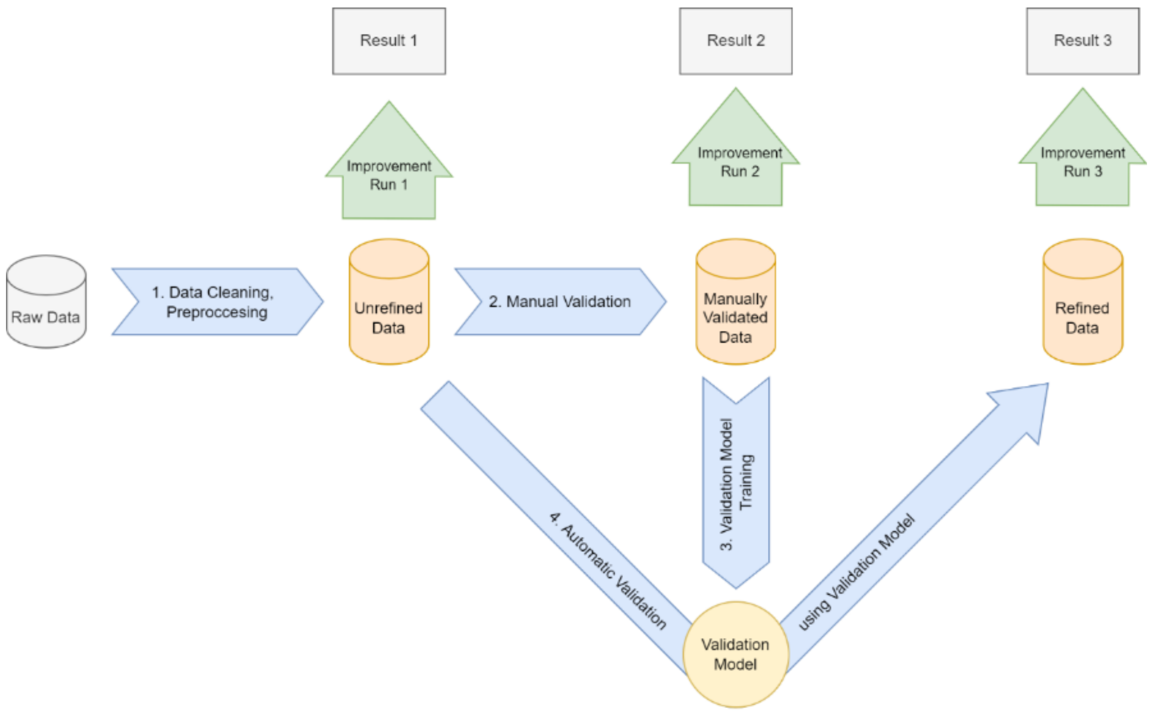


Figure 13 Experiment overview

### *4.3.1. Dataset Processing Steps and Artifacts*

This concept has four main steps, three resulting datasets, and a validation model artifact.

#### *4.3.1.1.Data Cleaning, Preprocessing Step*

This step uses the raw images and their metadata and results in the unrefined dataset used in the first improvement run. This step covers the removal of incomplete and inconsistent data first. While the improvement run uses product images to improve the metadata, the products with no image data are removed. Then, metadata conversion may apply. Suppose the metadata is a free text or numerical kind of metadata. In that case, a conversion needs to be conducted to convert text or numerical data into classes needed by the improvement run. After that metadata conversion, an image refinement is made on the image data. While there is a specific image size restriction for the CNN models, image resizing and scale may be needed according to the image data. Also, color correction and image preprocessing (like brightness and contrast modifications) may need to be according to the input image data.

After these steps, the unrefined dataset is ready to use in the first improvement run. This dataset has unrefined metadata and image data for a certain product category.

#### *4.3.1.2.Manual Validation Step*

In this step, a manual validation process is conducted by an expert. The expert randomly selects an equal number of samples for each class, according to considered metadata. The validation process covers both image data and metadata. The process starts with image validation. The expert validates the image data with a ruleset defined before the beginning of the validation process to minimize the subjectivity of the expert. These rules need to cover the characteristics of a valid image, like pose, quality parameters, etc. The expert selects the images that only fit the rules defined in the ruleset. Then it validates metadata. The expert selects only data with valid metadata according to the valid image according to the predefined ruleset.

After these steps, the manually validated dataset is ready for the second improvement run. This dataset includes a manually validated small subset of the unrefined metadata and image data for a certain product category.

#### *4.3.1.3.Validation Model Training*

This step covers creating an image validation model from the manually validated dataset's image data. A predefined CNN model, AlexNet, is used for training the model. The input dataset is split into training, validation, and test datasets with 60%, 20%, and 20% ratios similar to the improvement run. The input dataset includes the dropped invalid images. Then the model training is conducted with the hyperparameters defined.

The resulting model can classify images into two categories: valid and invalid. A valid class is used to express that the image is valid, according to the predefined ruleset, that can be used in the further metadata validation process. In contrast, an invalid class expresses that the image is invalid and cannot be used in the following metadata validation process. We can use this general image validation model to validate the unrefined dataset's image data in the next step. This model helps us to remove invalid images from the dataset. After the training, the validation model is used in the next step in inference mode.

#### *4.3.1.4.Automatic Validation Using Validation Model*

In this step, we use the general validation model to validate the images in the unrefined dataset. The model is used to classify all the images into two categories, valid and invalid. We drop invalid image data and metadata accordingly. Only valid images and related metadata are selected into the refined dataset with the help of the validation model.

After this step, the refined dataset is ready for the last improvement run. This dataset includes an automatically validated subset of the unrefined metadata and image data for a certain product category.

### *4.3.2. Metadata Improvement Runs*

Metadata improvement runs are applied to the resulting dataset three times, to the unrefined dataset, manually validated dataset, and refined dataset. CNN and baseline methods are applied separately to the input dataset on every run. All the results are collected first and then compared together. The inline schema of every run is shown in (Figure 14).

The input image data and selected metadata attribute data are combined first. Then this combined dataset is shuffled on a random basis. Then the resulting dataset is split into training, validation, and testing sub-datasets with 60%, 20%, and 20% consecutively for the CNN training and testing. This dataset is also used with baseline methods in a single, not separated manner.

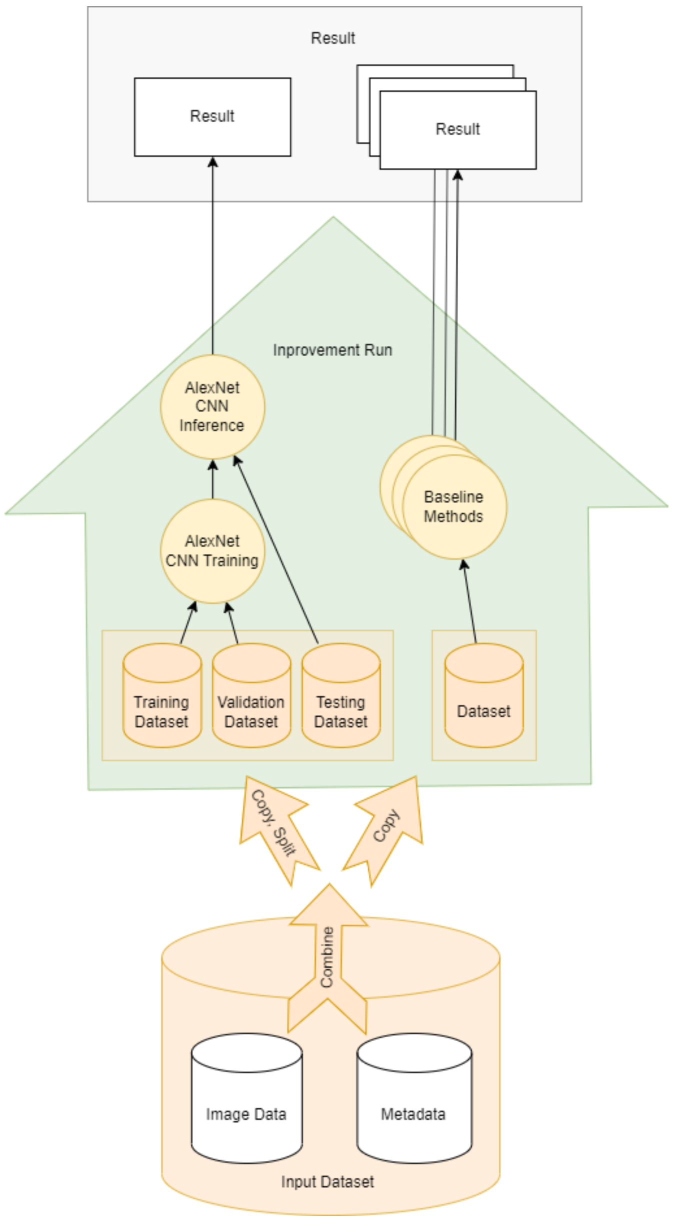


Figure 14 Experiment validation step in detail

After each run, we combined the results and discussed the outputs. Also, we compare the results of different improvement runs with different datasets mentioned before. We report accuracy, precision, recall, and f1 scores for every class on the dataset for CNN and baseline methods.

#### *4.3.2.1.Convolutional Neural Network (CNN)*

We use the AlexNet model as CNN on every improvement run (Krizhevsky et al., 2017). The model's top-5 error rate in 1000 different output classes is 15.3% which is 10.8 lower (better) than the best-performing model to date.

The alternative models are ResNet18 and VGG-Net11 (He et al., 2015; Simonyan & Zisserman, 2014). We take a random subset of 2000 images from the raw dataset and split it into training, validation, and testing datasets with the 60%, 20%, and 20% rule. Then, we run these three candidate models with the selected dataset in different hyperparameter settings. After that, we take the best-performing model according to the validation dataset accuracy and select AlexNet with a 0.7345 accuracy score. ResNet18 model performed 0.7267, and VGG-Net11 performed 0.7326 accuracy score in their best-performing hyperparameter settings.

The original paper of AlexNet uses images with 224 x 224 pixels in width and height consecutively, with RGB channels. It has five convolutional layers, three max-pooling layers, and three fully connected layers at the end. We use PyTorch implementation[7] , which is slightly different from the original architecture (Krizhevsky & Inc, 2014) (Table 3)

[7] https://github.com/pytorch/vision/blob/main/torchvision/models/alexnet.py

Table 3 CNN AlexNet PyTorch Implementation Architecture

| Layer | Details |
|---|---|
| Convolutional | 3 channels, 224x224, kernel:11x11, stride:4, padding:2 |
| ReLU | |
| Max Pooling | 64 channels, 55x55, kernel:3x3, stride:2 |
| Convolutional | 64 channels, 27x27, kernel:5x5, padding:2 |
| ReLU | |
| Max Pooling | 192 channels, 27x27, kernel:3x3, stride:2 |
| Convolutional | 192 channels, 13x13, kernel:3x3, padding:1 |
| ReLU | |
| Convolutional | 384 channels, 13x13, kernel:3x3, padding:1 |
| ReLU | |
| Convolutional | 256 channels, 13x13, kernel:3x3, padding:1 |
| ReLU | |
| Max Pooling | 256 channels, 13x13, kernel:3x3, stride:2 |
| Adaptive Avg. Pool | 256 channels, 6x6 |
| Flatten | |
| Dropout | |
| Linear | 9216 |
| ReLU | |
| Dropout | |
| Linear | 4096 |
| ReLU | |
| Linear | 4096 |
| SoftMax | 11 |

The input image size should be greater or equal to 224x224 pixels with three channels. The first convolutional layer is applied with an 11x11 kernel size, 64 output channels, a stride of 4 pixels, and padding of 2 pixels. ReLU and max pooling are applied to the output of the first convolutional layer with kernel size 3x3, stride 2 pixels.

The second convolutional layer is applied with a 5x5 kernel size, 192 output channels, and padding of 2 pixels. Then, ReLU and max pooling are applied with kernel size 3x3 pixels and stride 2 pixels. After that, three consecutive convolutional are applied with kernel size 3x3, and padding is 1 pixel. There are ReLU between them, not max pooling layers. The first of these three is applied with 384 output channels, and the remaining two are applied with 256 output channels. After the last one, max pooling is applied with kernel size 3x3 pixels with stride 2 pixels.

At the end of the convolutional part, an average adaptive pool is applied with 6x6 pixels. Then the fully connected part is started with two sequential composite layers consisting of one dropout layer connected with a fully connected layer with 4096 output and ReLU function. In the end, one fully connected layer is placed to connect 4096 features to the resulting classes with the softmax function.

This model converts 224x224 pixel RGB images into probability distributions of classes at the end. Then we use the argmax function to select the resulting class of the model.

#### *4.3.2.2.Baseline Methods*

To compare the performance of our approach, we developed baseline methods, which are functions that get input images and output a class of metadata with mathematical calculations. These methods use different statistical functions to aggregate image data into a classification. They are used as a baseline in the methodology to compare and prove the effectiveness of the proposed CNN model.

These methods are closely related to the selected metadata attribute. For example, suppose the selected metadata is color information. In that case, baseline methods aggerate the color information from the pixels of the images into one color class. Or, if the selected metadata is the pattern information, the patterns in the image are collected by the method and output a single pattern class.

## 4.4. Performance Evaluation Metrics

We use precision (regular, macro avg, weighted avg), recall (regular, macro avg, weighted avg), f1-score (regular, macro avg, weighted avg), and accuracy metrics while we measure the performance of the CNN and baseline models. We use the confusion matrix to compare the true and the predicted classes. We also use a loss-epoch and accuracy-epoch plot to see the effectiveness of the CNN model through each epoch.

Additionally, we measure the timing of each model. We report the measurements for each model, including CNN and baseline models, separately for each run in the result section. After the experiment, we compare the results of each run.

## 4.5. Baseline Algorithms Selection

We use three quantization algorithms and four pairwise distance algorithms as baseline algorithms. All the algorithms get input of all pixels of the input image as an array and output a probability distribution of 11 possible classes. The image is classified as multi-color if two or more colors have the possibility of more than 25%. If not, we select the highest probability as the main color class. The general perspective of a baseline algorithm is shown in Figure 15.

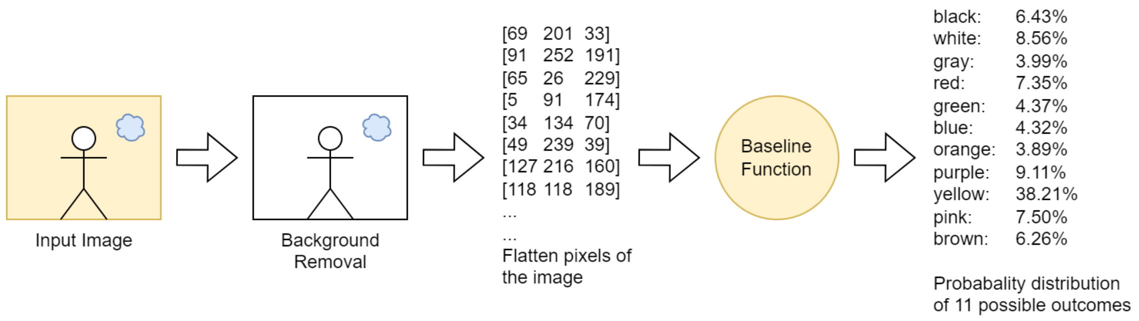


Figure 15 Baseline algorithm general perspective

The background removal from the images before processing is conducted with the help of an algorithm proposed by Suzuki & be (1985). The OpenCV library implementation of the algorithm is used.

If needed, we use the softmax function when converting distance vectors into probability distributions of the possible 11 main color classes. However, we convert the distance vector by dividing them by the max distance of the color space. The distance vector conversion and softmax function formula:

$$z_i = \frac{d_{max}}{d_i}$$

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{11} e^{z_j}}$$

We generate a 4x4 pixels image in RGB color space to use in the examples for explaining the baseline methods. The example image pixel values are shown in Table 4. We use three classes only to simplify the example calculations, black (R:0, G:0, B:0), white (R:255, G:255, B:255), and gray (R:128, G:128, B:128).

Table 4 Example 4x4 image pixel RGB values

| | **#1** | **#2** | **#3** | **#4** |
|---|---|---|---|---|
| **#1** | (69, 201, 33) | (34, 134, 70) | (129, 100, 73) | (139, 173, 72) |
| **#2** | (191, 252, 191) | (49, 239, 39) | (145, 172, 240) | (24, 41, 25) |
| **#3** | (65, 26, 229) | (127, 216, 160) | (251, 2, 169) | (124, 68, 122) |
| **#4** | (105, 91, 174) | (118, 118, 189) | (206, 31, 245) | (165, 191, 208) |

### *4.5.1. Quantization Based Baseline Algorithms*

In general, quantization algorithms summarize the colors of an image and decrease the palette usage of a digital image. In this way, they are used as an image compression technique. This study uses quantization algorithms as an image's main color identification algorithm. They take the image and try to output a single-color point for the whole image. So, they summarize the whole image into one color. We take that one color as the primary color of the image and find the pairwise distance of this color and the color classes. Then we convert these distances into a probability distribution of 11 possible outcomes. We use Modified Median Color Quantization (MMCQ), OctTree Quantization and Modified MMCQ (pngquant) Quantization algorithms (Gervautz & Purgathofer, 1988).

MMCQ quantization aggregates 16 pixels of the example image to one RGB (68, 204, 36) color. The Euclidean distances of this color to the color classes and the probability distribution of example three classes are shown in Table 5.

Table 5 MMCQ quantization example result

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Distances** | 218.03 | 292.46 | 133.57 |
| **Result** | 19.00% | 11.00% | 69.00% |

OctTree quantization aggregates 16 pixels of the example image to one RGB (155, 181, 224) color. The Euclidean distances of this color to the color classes and the probability distribution of example three classes are shown in Table 6.

Table 6 OctTree quantization example result

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Distances** | 327.05 | 128.21 | 112.93 |
| **Result** | 5.00% | 37.00% | 59.00% |

Modified MMCQ (pngquant) quantization aggregates 16 pixels of the example image to one RGB (125, 134, 145) color. The Euclidean distances of this color to the color classes and the probability distribution of example three classes are shown in Table 7.

Table 7 Modified MMCQ quantization example result

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Distances** | 234.68 | 209.90 | 19.28 |
| **Result** | 0.00% | 0.00% | 100.00% |

### *4.5.2. Pairwise Distance Based Baseline Algorithms*

The pairwise distance algorithm calculates the pairwise distance of each image pixel with the color classes. This distance is calculated with the Euclidean distance formula for the RGB color space. For the HSL color space, we use the circular bi-cone distance formula extracted with the help of Euclidean distance. This algorithm outputs a matrix with color classes in one of the axes while all the pixel colors are in the other. We use four techniques to aggregate this matrix into the probability distribution of the main color classes.

#### *4.5.2.1.Pairwise Distance Quantiles*

In the first technique, we use quantiles. We sort the distances against each main color class. Then we take the distance value at 0.25, 0.5, and 0.75 quantiles and convert them into probability distribution of the main color classes. We use the linear calculation of the quantiles stated as "Definition 7" by Hyndman & Fan (1996).

The pairwise distances of the example image sorted by each class are shown in Table 8.

Table 8 The pairwise distances of the example image sorted by each class

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Q1** | 53.69 | 90.56 | 60.44 |
| | 155 | 120 | 61.72 |
| | 178.75 | 138.62 | 62.62 |
| | 186.75 | 164.12 | 63.34 |
| **Q2** | 215 | 204.62 | 72.69 |
| | 222.62 | 229.5 | 93.62 |
| | 233.25 | 231.62 | 108.31 |
| | 239.5 | 236.5 | 110.62 |
| **Q3** | 247.12 | 264.25 | 121.5 |
| | 252.12 | 267.25 | 133.5 |
| | 297.25 | 270.25 | 152.75 |
| | 302.5 | 294.5 | 156.75 |
| **Q4** | 321.5 | 298.75 | 162.75 |
| | 327 | 299 | 170.25 |
| | 329 | 312.5 | 170.88 |
| | 369.5 | 390 | 180.75 |

The distances for 0.25, 0.50, and 0.75 quantiles and the resulting probability distributions are shown in Table 9.

Table 9 Pairwise distance quantiles of the example image

| | black | white | gray |
|---|---|---|---|
| **0.25 Distance** | 207.94 | 194.50 | 70.35 |
| **0.25 Result** | 1.52% | 1.76% | 96.72% |
| **0.50 Distance** | 243.31 | 250.38 | 116.06 |
| **0.50 Result** | 10.79% | 10.25% | 78.96% |
| **0.75 Distance** | 307.25 | 295.56 | 158.25 |
| **0.75 Result** | 16.87% | 17.85% | 65.28% |

#### *4.5.2.2.Pairwise ArgMin*

We take the minimum distance for each pixel to the main color classes in the second pairwise distance technique. Then we count them and convert them into class probability distribution of the main color classes using the percentage. We called this algorithm "Pairwise ArgMin."

The pairwise distance of each pixel to the class points, the argmin class of each pixel, the count of selected classes, and the result are shown in Table 10.

Table 10 Pairwise ArgMin result of the example image

| | black | white | gray | ArgMin |
|---|---|---|---|---|
| **1** | 215 | 294.5 | 133.5 | gray |
| **2** | 369.5 | 90.56 | 152.75 | white |
| **3** | 239.5 | 298.75 | 156.75 | gray |
| **4** | 222.62 | 236.5 | 63.34 | gray |
| **5** | 155 | 312.5 | 110.62 | gray |
| **6** | 247.12 | 299 | 162.75 | gray |
| **7** | 297.25 | 164.12 | 93.62 | gray |
| **8** | 252.12 | 204.62 | 62.62 | gray |
| **9** | 178.75 | 270.25 | 61.72 | gray |
| **10** | 329 | 138.62 | 121.5 | gray |
| **11** | 302.5 | 267.25 | 180.75 | gray |
| **12** | 321.5 | 229.5 | 170.88 | gray |
| **13** | 233.25 | 231.62 | 72.69 | gray |
| **14** | 53.69 | 390 | 170.25 | black |
| **15** | 186.75 | 264.25 | 60.44 | gray |
| **16** | 327 | 120 | 108.31 | gray |
| | | | | |
| **Count** | 1 | 1 | 14 | |
| **Result** | 6.25% | 6.25% | 87.50% | |

*4.5.2.3.Pairwise Mean*

In the third one, we take the average distance value of all the distances from each pixel to the main color classes. And we convert them into probability distribution of the main color classes. This algorithm is called “Pairwise Mean.”

The mean distance of the pairwise distances is shown in Table 10, and the result of the algorithm is shown in Table 10.

Table 11 Pairwise mean result of the example image

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Mean Distance** | 245.66 | 238.25 | 117.66 |
| **Result** | 10.95% | 11.58% | 77.46% |

*4.5.2.4.Array Mean ArgMin*

In the last pairwise algorithm, we take the mean value of all pixels in the image called the center point. Then, we calculate the distance between this center point and the main color points. And convert these distances into probability distribution of the main color classes. We called this algorithm “Array Mean ArgMin.”

The mean of the pixels of the example image is RGB (121, 128, 140). This color’s pairwise distances and the result are shown in Table 12.

Table 12 Array Mean ArgMin result of the example image

| | **black** | **white** | **gray** |
|---|---|---|---|
| **Distance** | 225.0 | 217.5 | 13.9 |
| **Result** | 0.00% | 0.00% | 100.00% |

## 4.6. Model Implementation and Experiment Settings

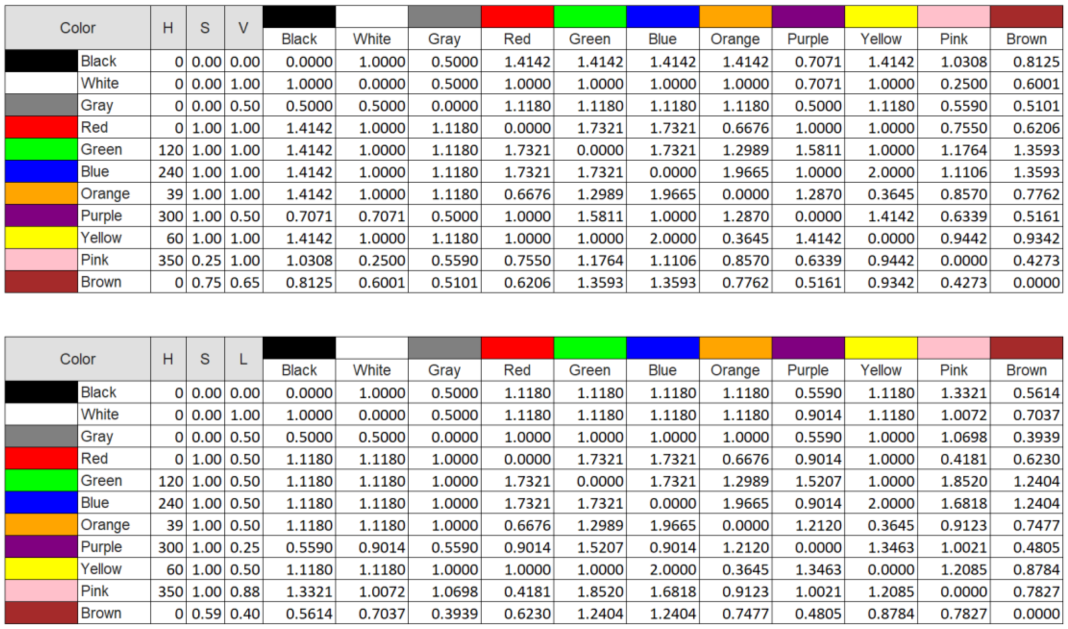
### *4.6.1. Implementation Details*

We implemented the experiment model in Python Jupiter notebooks and published it in the GitHub repository[8]. We use PyTorch and other common Python libraries (NumPy,

[8] https://github.com/alatas/MSThesis

matplotlib, etc.) in the implementation. The steps of the experiment are implemented in separate Jupyter notebooks. Each notebook can be run on a local personal computer or Google Colab environment. There is an option to use Google Drive as the persistent storage while using Google Colab in the notebooks.

We use 11 color classes "black," "white," "gray," "red," "green," "blue," "orange," "purple," "yellow," "pink," "brown," and "multi-color." The distances between selected color classes in HSV and HSL color spaces are shown in Figure 16.



| Color | | H | S | V | ⬛ Black | ⬜ White | ▩ Gray | 🟥 Red | 🟩 Green | 🟦 Blue | 🟧 Orange | 🟪 Purple | 🟨 Yellow | ▨ Pink | 🟫 Brown |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⬛ | Black | 0 | 0.00 | 0.00 | 0.0000 | 1.0000 | 0.5000 | 1.4142 | 1.4142 | 1.4142 | 1.4142 | 0.7071 | 1.4142 | 1.0308 | 0.8125 |
| ⬜ | White | 0 | 0.00 | 1.00 | 1.0000 | 0.0000 | 0.5000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.7071 | 1.0000 | 0.2500 | 0.6001 |
| ▩ | Gray | 0 | 0.00 | 0.50 | 0.5000 | 0.5000 | 0.0000 | 1.1180 | 1.1180 | 1.1180 | 1.1180 | 0.5000 | 1.1180 | 0.5590 | 0.5101 |
| 🟥 | Red | 0 | 1.00 | 1.00 | 1.4142 | 1.0000 | 1.1180 | 0.0000 | 1.7321 | 1.7321 | 0.6676 | 1.0000 | 1.0000 | 0.7550 | 0.6206 |
| 🟩 | Green | 120 | 1.00 | 1.00 | 1.4142 | 1.0000 | 1.1180 | 1.7321 | 0.0000 | 1.7321 | 1.2989 | 1.5811 | 1.0000 | 1.1764 | 1.3593 |
| 🟦 | Blue | 240 | 1.00 | 1.00 | 1.4142 | 1.0000 | 1.1180 | 1.7321 | 1.7321 | 0.0000 | 1.9665 | 1.0000 | 2.0000 | 1.1106 | 1.3593 |
| 🟧 | Orange | 39 | 1.00 | 1.00 | 1.4142 | 1.0000 | 1.1180 | 0.6676 | 1.2989 | 1.9665 | 0.0000 | 1.2870 | 0.3645 | 0.8570 | 0.7762 |
| 🟪 | Purple | 300 | 1.00 | 0.50 | 0.7071 | 0.7071 | 0.5000 | 1.0000 | 1.5811 | 1.0000 | 1.2870 | 0.0000 | 1.4142 | 0.6339 | 0.5161 |
| 🟨 | Yellow | 60 | 1.00 | 1.00 | 1.4142 | 1.0000 | 1.1180 | 1.0000 | 1.0000 | 2.0000 | 0.3645 | 1.4142 | 0.0000 | 0.9442 | 0.9342 |
| ▨ | Pink | 350 | 0.25 | 1.00 | 1.0308 | 0.2500 | 0.5590 | 0.7550 | 1.1764 | 1.1106 | 0.8570 | 0.6339 | 0.9442 | 0.0000 | 0.4273 |
| 🟫 | Brown | 0 | 0.75 | 0.65 | 0.8125 | 0.6001 | 0.5101 | 0.6206 | 1.3593 | 1.3593 | 0.7762 | 0.5161 | 0.9342 | 0.4273 | 0.0000 |

| Color | | H | S | L | ⬛ Black | ⬜ White | ▩ Gray | 🟥 Red | 🟩 Green | 🟦 Blue | 🟧 Orange | 🟪 Purple | 🟨 Yellow | ▨ Pink | 🟫 Brown |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⬛ | Black | 0 | 0.00 | 0.00 | 0.0000 | 1.0000 | 0.5000 | 1.1180 | 1.1180 | 1.1180 | 1.1180 | 0.5590 | 1.1180 | 1.3321 | 0.5614 |
| ⬜ | White | 0 | 0.00 | 1.00 | 1.0000 | 0.0000 | 0.5000 | 1.1180 | 1.1180 | 1.1180 | 1.1180 | 0.9014 | 1.1180 | 1.0072 | 0.7037 |
| ▩ | Gray | 0 | 0.00 | 0.50 | 0.5000 | 0.5000 | 0.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.5590 | 1.0000 | 1.0698 | 0.3939 |
| 🟥 | Red | 0 | 1.00 | 0.50 | 1.1180 | 1.1180 | 1.0000 | 0.0000 | 1.7321 | 1.7321 | 0.6676 | 0.9014 | 1.0000 | 0.4181 | 0.6230 |
| 🟩 | Green | 120 | 1.00 | 0.50 | 1.1180 | 1.1180 | 1.0000 | 1.7321 | 0.0000 | 1.7321 | 1.2989 | 1.5207 | 1.0000 | 1.8520 | 1.2404 |
| 🟦 | Blue | 240 | 1.00 | 0.50 | 1.1180 | 1.1180 | 1.0000 | 1.7321 | 1.7321 | 0.0000 | 1.9665 | 0.9014 | 2.0000 | 1.6818 | 1.2404 |
| 🟧 | Orange | 39 | 1.00 | 0.50 | 1.1180 | 1.1180 | 1.0000 | 0.6676 | 1.2989 | 1.9665 | 0.0000 | 1.2120 | 0.3645 | 0.9123 | 0.7477 |
| 🟪 | Purple | 300 | 1.00 | 0.25 | 0.5590 | 0.9014 | 0.5590 | 0.9014 | 1.5207 | 0.9014 | 1.2120 | 0.0000 | 1.3463 | 1.0021 | 0.4805 |
| 🟨 | Yellow | 60 | 1.00 | 0.50 | 1.1180 | 1.1180 | 1.0000 | 1.0000 | 1.0000 | 2.0000 | 0.3645 | 1.3463 | 0.0000 | 1.2085 | 0.8784 |
| ▨ | Pink | 350 | 1.00 | 0.88 | 1.3321 | 1.0072 | 1.0698 | 0.4181 | 1.8520 | 1.6818 | 0.9123 | 1.0021 | 1.2085 | 0.0000 | 0.7827 |
| 🟫 | Brown | 0 | 0.59 | 0.40 | 0.5614 | 0.7037 | 0.3939 | 0.6230 | 1.2404 | 1.2404 | 0.7477 | 0.4805 | 0.8784 | 0.7827 | 0.0000 |

Figure 16 HSV and HSL distance matrices

We compared HSV and HSL color spaces before the implementation to find a more suitable color space for the experiment with the matrices shown in Figure 16. HSL color space has lower distance values for close color classes. For example, HSL has 0.4181, HSV has a 0.7550 value for the Red/Pink distance, while both color spaces have a 1.0000 distance value between Black/White. Similarly, HSV has a 0.9342 distance value, while HSL has a 0.8784 distance value between Yellow/Brown. According to this comparison, we decided to use HSL color space instead of HSV color space. Also, we use RGB color space, the most common color space. In the implementation, notebooks have the option to switch between color spaces. The distance algorithm used in baseline algorithms changes according to the color space used. The rest of the experiment is not changing according to the color space.

The HTML color names define these RGB and HSL color points. The RGB and HSL values of these classes are shown in (Table 13).

Table 13 Class center points in RGB and HSL color spaces

| Color | RGB Value | HSL Value |
|---|---|---|
| Black | [0, 0, 0] | [0, 0.0, 0.0] |
| White | [255, 255, 255] | [0, 0.0, 1.0] |
| Gray | [128, 128, 128] | [0, 0.0, 0.5] |
| Red | [255, 0, 0] | [0, 1.0, 0.5] |
| Green | [0, 255, 0] | [120, 1.0, 0.5] |
| Blue | [0, 0, 255] | [240, 1.0, 0.5] |
| Orange | [255, 165, 0] | [39, 1.0, 0.5] |
| Purple | [128, 0, 128] | [300, 1.0, 0.25] |
| Yellow | [255, 255, 0] | [60, 1.0, 0.5] |
| Pink | [255, 192, 203] | [350, 1.0, 0.88] |
| Brown | [165, 42, 42] | [0, 0.59, 0.40] |

The dataset has text color names and HTML color codes for each product. We examine the HTML color codes provided with the random sampling method. We took 100 random samples and checked them with the actual images. Most HTML codes are close to the actual primary color of sample images. But, when we use the HTML color provided, we need to convert them to a color class with a method like Euclidean distance. Also, most marketplaces do not have a well-defined color point for the metadata. So, we decided to use text color names and a conversion matrix to convert text color names into color classes. The conversion table is shown in Table 14.

Table 14 Text color attribute to the color classes conversion table

| Color Classes | Text Color Attribute |
|---|---|
| "multi-color" | "multi" |
| "black" | "black", "asphalt", "caviar", "graphite", "Schwarz" |
| "white" | "white", "ivory" |
| "gray" | "gray", "grey", "chrome", "silver", "steel", "charcoal", "nickel", "aluminum", "anthracite", "ash", "dove", "fog", "iron", "pewter", "platinum", "slate", "sliver", "smoke", "stainless" |
| "red" | "red", "rose", "bordeaux", "burgundy", "maroon", "merlot", "autumn", "berry", "brick", "burgandy", "cherry", "garnet", "mahogany", "maron" |
| "green" | "green", "mint", "olive", "alligator", "aloe", "cadet", "emerald", "lagoon", "lemongrass", "sage", "seafoam", "sod", "teal", "turquoise" |
| "blue" | "blue", "navy", "aqua", "denim", "azure", "blau", "bule", "sapphire", "sky" |
| "orange" | "orange", "fire", "flame", "fawn", "pumpkin", "rust" |
| "purple" | "purple", "amethyst", "fuchsia", "heather", "lavender", "lilac", "magenta" |

Table 14 (cont.) Text color attribute to the color classes conversion table

| Color Classes | Text Color Attribute |
|---|---|
| "yellow" | "yellow", "gold", "amber", "brass", "butter", "canary", "citrine", "flax" |
| "pink" | "pink", "blush", "champagne", "linen" |
| "brown" | "brown", "beige", "biege" "braun", "bronze", "camel", "caramel", "sand", "tan", "walnut", "acorn", "antique", "barnwood", "chestnut", "chocolate", "cognac", "ecru", "hemp", "khaki", "oak", "saddle", "taupe", "wenge" |

The product is ignored and removed from the dataset if the color name is not defined, or not in the list stated in Table 14. Additionally, if a text includes different two-color classes (like "navy black", or "brown black"), it is defined as multi-color.

The experiment is run with two multi-color options. In the first option, we consider a multi-color class like the other color classes. In the second option, we do not use the multi-color as a color class and force the models to choose one of the main color classes. We select the color with the highest probability in baseline algorithms when using the single-color option. We report the results with multi-color and single-color options separately.

We use 60% training, 20% validation, and 20% testing separation while training the CNN models.

### *4.6.2. Hyperparameter Tuning*

On hyperparameter tuning, we use a two-phased approach. In the first phase, we use a grid search with a broad set of possible parameters. After that phase, we select the best-performing set of hyperparameters from the first phase and run the second phase with a new set of hyperparameters close to the first phase's output. For the first phase, all possible hyperparameters are shown in Table 15.

Table 15 Hyperparameter set for the first phase

| Hyperparameter | Possible Values |
|---|---|
| **learning_rate** | [0.1, 0.01, 0.001, 0.0001, 1e-05, 1e-06] |
| **optimizer** | ["Adam", "RMSprop", "SGD"] |
| **dropout** | [0.05, 0.1, 0.15, 0.2 ] |
| **batch_size** | [16, 32, 64, 128] |

We use the Optuna[9] framework in hyperparameter tuning (Akiba et al., 2019). In the first phase, we use the "GridSampler" sampler, which suggests all combinations of parameters in the given search space during the study. The first phase has 288 (7 x 3 x 4 x 4) available combinations of all possible hyperparameter values shown in Table 15. We select the best-performing set of hyperparameters according to the testing dataset accuracy score. Then, we select a new set of hyperparameters for the second phase with the help of four plots drawn by the outputs of the first phase. These plots are the "Intermediate Values Plot," "Optimization History Plot," "Parallel Coordinate Plot," and "Hyperparameter Importance Plot." The Intermediate Values Plot shows the accuracy score after each epoch for all the trials. The Optimization History Plot shows the best accuracy score after each trial. The Parallel Coordinate Plot shows each trial's set of selected hyperparameters and output accuracy. The hyperparameter importance plot shows the importance of each hyperparameter in percentage according to the accuracy score (see Appendix A for the details).

In the second phase, we use "TPESampler," which uses the Tree-structured Parzen Estimator algorithm. This sampler fits one Gaussian Mixture Model (GMM) (represented as 'l(x)') to the set of parameter values associated with the best objective values. Then it fits another GMM (represented as 'g(x)') to the remaining parameter values. In the end, TPE chooses the parameter value x that maximizes the ratio l(x)/g(x) (Bergstra et al., n.d., 2013; *Optuna.Samplers.TPESampler — Optuna 3.1.0 Documentation*, n.d.; Ozaki et al., 2020, 2022). We run the second phase for a total of 500 trials. On each trial, Optuna selects a new hyperparameter set and runs the model to achieve a better accuracy score on the testing dataset. Ultimately, we select the best-performing set of hyperparameters according to the accuracy score on the testing dataset.

Additionally, we use the "prune" feature on the Optuna framework on both phases to release resources for the trials are already have worse accuracy scores on the run. Optuna decides that the current trial could be pruned according to the current accuracy score on each epoch. We use the "median pruner" algorithm to decide on prune for the current trial. The median pruner decides to prune the current trial if the current trial's best intermediate result is worse than the median of intermediate results of previous trials at the same step.

We conducted two hyperparameter tuning sessions for two models. In the first model, we use the manually validated image dataset to tune hyperparameters for Unrefined Dataset Improvement Run, Manually Validated Dataset Improvement Run, and Refined Dataset Improvement Run. We run these sessions for 120 epochs in total. We use the automatic validation dataset in the second model to tune hyperparameters for Automatic Validation Model Training Run. We run these experiments for 60 epochs in total.

---

[9] https://optuna.org/

We conducted these two tuning sessions separately in two color spaces, RGB and HSL.

## 4.7. Results

We conducted eight hyperparameter sessions and 26 experiments over three different datasets. We define the results under five sections. In the first section, we give the results of the hyperparameter tuning sessions. And we give the results of the Automatic Validation Model; then, we give the results of the runs of three different runs on different color spaces and models.

### *4.7.1. Hyperparameter Tuning Results*

The results of the phases for each hyperparameter tuning session are shown in Table 16 (see Appendix A for the details).

Table 16 Hyperparameter Tuning Results

**Dataset Improvement Run Hyperparameter Tuning**

| | **RGB Color Space** | | **HSL Color Space** | |
|---|---|---|---|---|
| | First Phase | Second Phase | First Phase | Second Phase |
| Best Accuracy: | 0.8163 | 0.8265 | 0.7193 | 0.75 |
| Batch Size: | 16 | 16 | 32 | 16 |
| Dropout: | 0.15 | 0.15 | 0.1 | 0.11 |
| Learning Rate: | 0.0001 | 0.0001 | 0.0001 | 0.00008 |
| Optimizer: | RMSProp | RMSProp | RMSProp | RMSProp |

**Automatic Validation Model Hyperparameter Tuning**

| | **RGB Color Space** | | **HSL Color Space** | |
|---|---|---|---|---|
| | First Phase | Second Phase | First Phase | Second Phase |
| Best Accuracy: | 0.9558 | 0.9676 | 0.9205 | 0.9264 |
| Batch Size: | 16 | 16 | 32 | 16 |
| Dropout: | 0.15 | 0.13 | 0.05 | 0.07 |
| Learning Rate: | 0.00001 | 0.00005 | 0.0001 | 0.00006 |
| Optimizer: | RMSProp | RMSProp | Adam | Adam |

### *4.7.2. Automatic Validation Model Results*

After the manual validation process with an expert, we extracted 1092 valid and classified images for 11 classes. We tried to take a hundred validated image samples for each class. However, the purple and orange classes did not have enough validated 100 images at the end. Finally, we found 1092 valid, 609 invalid, and 715 misclassified images in 2416 images. If we ignore the multi-color class, there is a total of 654 misclassified, 555 invalid images, and 992 valid images in 2201 total images.

The automatic validation model training run is conducted on the same CNN AlexNet model. The model performed a 0.9294 accuracy score on RGB color space and a 0.9059 accuracy score on HSL color space on the testing dataset between “invalid” and “valid” classes. RGB model run is completed in 414.54 seconds, while the HSL model run is completed in 786.25 seconds. The summary is shown in Table 17, and some of the sample misclassified images on Figure 17. (see Appendix A for the details)

After the model was finalized, we compared RGB and HSL accuracy scores. We run the RGB model on the unrefined dataset. In the manually validated dataset, the invalid image ratio was 25.20%. The model marked 5744 images out of 21454 images. The invalid ratio is 26.77%, which is consistent with the manually validated dataset's invalid ratio.

Table 17 Automatic validation model run results summary

| | | **Precision** | **Recall** | **F1 Score** | **Support** |
|---|---|---|---|---|---|
| RGB Color Space | **invalid** | 0.9391 | 0.8640 | 0.9000 | 125 |
| | **valid** | 0.9244 | 0.9674 | 0.9455 | 215 |
| | **accuracy** | | | 0.9294 | |
| | **macro avg** | 0.9318 | 0.9157 | 0.9227 | 340 |
| | **weighted avg** | 0.9298 | 0.9294 | 0.9287 | |

| | | **Precision** | **Recall** | **F1 Score** | **Support** |
|---|---|---|---|---|---|
| HSL Color Space | **invalid** | 0.8974 | 0.8400 | 0.8678 | 125 |
| | **valid** | 0.9103 | 0.9442 | 0.9269 | 215 |
| | **accuracy** | | | 0.9059 | |
| | **macro avg** | 0.9039 | 0.8921 | 0.8974 | 340 |
| | **weighted avg** | 0.9056 | 0.9059 | 0.9052 | |

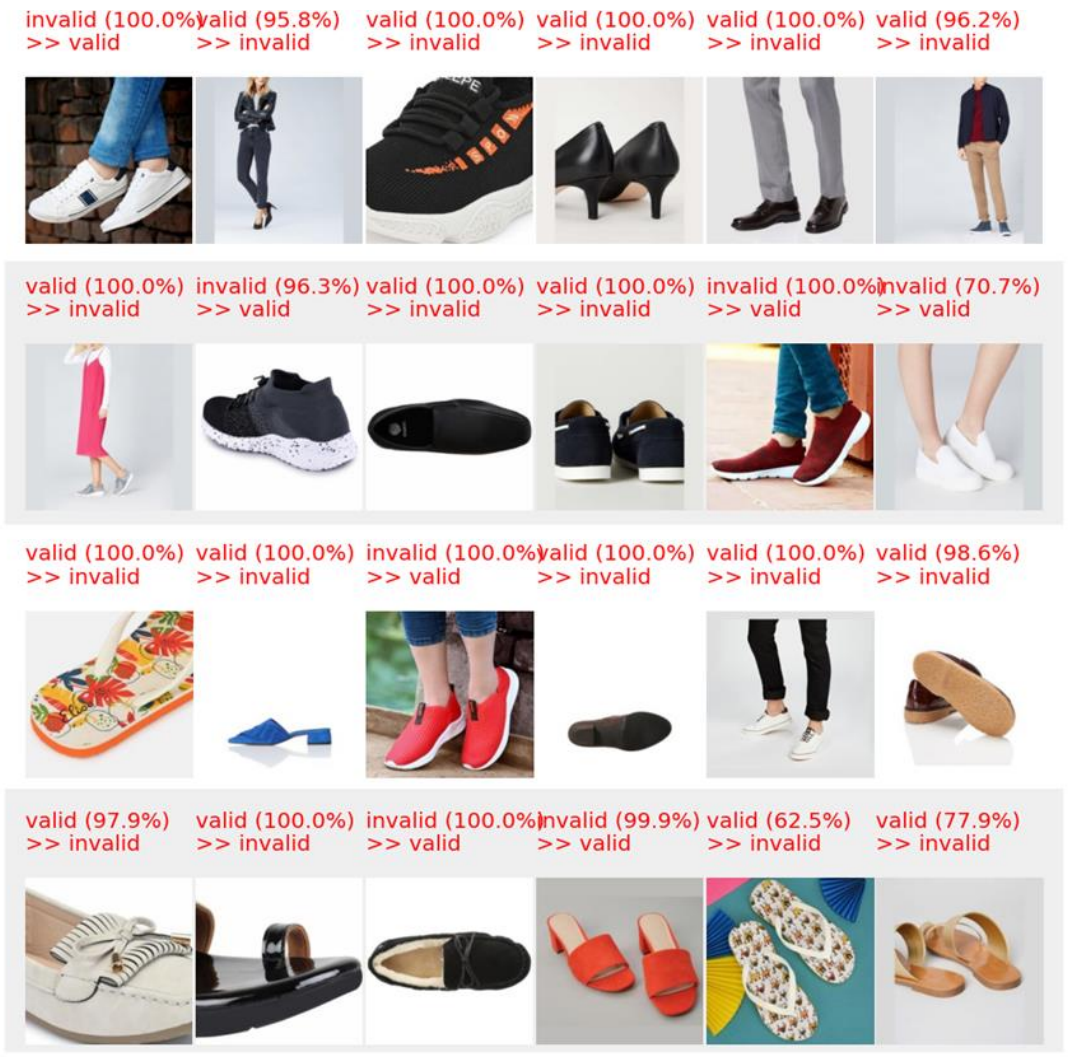

invalid (100.0%) >> valid
valid (95.8%) >> invalid
valid (100.0%) >> invalid
valid (100.0%) >> invalid
valid (100.0%) >> invalid
valid (96.2%) >> invalid
valid (100.0%) >> invalid
invalid (96.3%) >> valid
valid (100.0%) >> invalid
valid (100.0%) >> invalid
invalid (100.0%) >> valid
invalid (70.7%) >> valid
valid (100.0%) >> invalid
valid (100.0%) >> invalid
invalid (100.0%) >> valid
valid (100.0%) >> invalid
valid (100.0%) >> invalid
valid (98.6%) >> invalid
valid (97.9%) >> invalid
valid (100.0%) >> invalid
invalid (100.0%) >> valid
invalid (99.9%) >> valid
valid (62.5%) >> invalid
valid (77.9%) >> invalid

Figure 17 Sample Images that cannot be accurately classified in the Test Dataset

After the automatic image validation process, as mentioned in section 4.3.2, we conducted metadata improvement runs in three steps over three datasets: the unrefined dataset, the manually validated dataset, and the refined dataset.

### *4.7.3. Unrefined Dataset Improvement Runs*

This run is conducted over the dataset containing the raw images and metadata. For RGB Color Space, the CNN AlexNet model performed a 0.7891 accuracy score in 2865.16 seconds with the single-color mode and a 0.6909 accuracy score in 3194.93 seconds with the multi-color mode. The best baseline method performed a 0.4076 accuracy score in

98.80[10] seconds in single-color mode and a 0.3468 accuracy score in 53.09 seconds in multi-color mode.

For HSL Color Space, the CNN AlexNet model performed a 0.7540 accuracy score in 5519.77 seconds with the single-color mode and a 0.6506 accuracy score in 6201.11 seconds with the multi-color mode. The best baseline method performed a 0.4277 accuracy score in 44.44 seconds in single-color mode and a 0.3535 accuracy score in 97.38 seconds in multi-color mode. The summary is shown in Table 18. (see Appendix A for the details)

[10] The duration for the baseline methods doesn't include the pairwise distance calculation duration. The average durations are 190 image per second for RGB color space and 40 image per second for HSL color space.

Table 18 Unrefined dataset improvement run result summary

| | | | | **Precision** | **Recall** | **F1 Score** | **Accuracy** | **Support** |
|---|---|---|---|---|---|---|---|---|
| **Single Color** | CNN AlexNet Model | RGB | macro avg | **0.6564** | **0.5882** | **0.6095** | **0.7891** | 3703 |
| | | | weighted avg | **0.7896** | **0.7891** | **0.7862** | | |
| | | HSL | macro avg | 0.5590 | 0.5236 | 0.5326 | 0.7540 | |
| | | | weighted avg | 0.7531 | 0.7540 | 0.7515 | | |
| | Best Baseline Method | RGB | macro avg | 0.4034 | 0.2305 | 0.1831 | 0.4076 | 18592 |
| | | | weighted avg | 0.5564 | 0.4076 | 0.3336 | | |
| | | HSL | macro avg | 0.4387 | 0.2563 | 0.2135 | 0.4277 | |
| | | | weighted avg | 0.5576 | 0.4277 | 0.3515 | | |

| | | | | **Precision** | **Recall** | **F1 Score** | **Accuracy** | **Support** |
|---|---|---|---|---|---|---|---|---|
| **Multi-Color** | CNN AlexNet Model | RGB | macro avg | **0.6069** | **0.5467** | **0.5624** | **0.6909** | 4290 |
| | | | weighted avg | **0.6735** | **0.6909** | **0.6750** | | |
| | | HSL | macro avg | 0.5239 | 0.4876 | 0.4935 | 0.6506 | |
| | | | weighted avg | 0.6333 | 0.6506 | 0.6361 | | |
| | Best Baseline Method | RGB | macro avg | 0.3402 | 0.2101 | 0.1578 | 0.3468 | 21454 |
| | | | weighted avg | 0.4061 | 0.3468 | 0.2698 | | |
| | | HSL | macro avg | 0.2736 | 0.2382 | 0.1902 | 0.3535 | |
| | | | weighted avg | 0.3614 | 0.3535 | 0.2867 | | |

### *4.7.4. Manually Validated Dataset Improvement Runs*

This run is conducted over the dataset containing the selected images and their metadata. For RGB Color Space, the CNN AlexNet model performed a 0.7806 accuracy score in

520.04 seconds with the single-color mode and a 0.6651 accuracy score in 547.82 seconds with the multi-color mode. The best baseline method performed a 0.3347 accuracy score in 1.91 seconds in single-color mode and a 0.2873 accuracy score in 2.12 seconds in multi-color mode.

For HSL Color Space, the CNN AlexNet model performed a 0.7041 accuracy score in 936.43 seconds with the single-color mode and a 0.6193 accuracy score in 1034.46 seconds with the multi-color mode. The best baseline method performed a 0.3609 accuracy score in 0.64 seconds in single-color mode and a 0.3068 accuracy score in 2.25 seconds in multi-color mode. The summary is shown in Table 19. (see Appendix A for the details)

Table 19 Manually validated dataset improvement run result summary

| | | | | Precision | Recall | F1 Score | Accuracy | Support |
|---|---|---|---|---|---|---|---|---|
| **Single Color** | CNN AlexNet Model | RGB | macro avg | **0.7064** | **0.6933** | **0.6923** | **0.7806** | 196 |
| | | | weighted avg | **0.7982** | **0.7806** | **0.7816** | | |
| | | HSL | macro avg | 0.6702 | 0.6388 | 0.6423 | 0.7041 | |
| | | | weighted avg | 0.7308 | 0.7041 | 0.7067 | | |
| | Best Baseline Method | RGB | macro avg | 0.3669 | 0.2856 | 0.2167 | 0.3347 | 992 |
| | | | weighted avg | 0.3785 | 0.3347 | 0.2436 | | |
| | | HSL | macro avg | 0.2547 | 0.3016 | 0.2231 | 0.3609 | |
| | | | weighted avg | 0.2718 | 0.3609 | 0.2627 | | |

| | | | | Precision | Recall | F1 Score | Accuracy | Support |
|---|---|---|---|---|---|---|---|---|
| **Multi-Color** | CNN AlexNet Model | RGB | macro avg | **0.6621** | **0.6728** | **0.6445** | **0.6651** | 218 |
| | | | weighted avg | **0.6939** | **0.6651** | **0.6580** | | |
| | | HSL | macro avg | 0.6322 | 0.6153 | 0.6010 | 0.6193 | |
| | | | weighted avg | 0.6242 | 0.6193 | 0.6010 | | |
| | Best Baseline Method | RGB | macro avg | 0.2535 | 0.2398 | 0.1855 | 0.2873 | 1092 |
| | | | weighted avg | 0.2704 | 0.2873 | 0.2192 | | |
| | | HSL | macro avg | 0.2626 | 0.2806 | 0.2143 | 0.3068 | |
| | | | weighted avg | 0.2556 | 0.3068 | 0.2326 | | |

### *4.7.5. Refined Dataset Improvement Runs*

This run is conducted over the dataset containing the automatically validated images and their metadata. For RGB Color Space, the CNN AlexNet model performed a 0.8383 accuracy score in 2368.36 seconds with the single-color mode and a 0.7279 accuracy score in 2594.60 seconds with the multi-color mode. The best baseline method performed a 0.4331 accuracy score in 37.56 seconds in single-color mode and a 0.3664 accuracy score in 44.44 seconds in multi-color mode.

For HSL Color Space, the CNN AlexNet model performed a 0.7986 accuracy score in 4751.72 seconds with the single-color mode and a 0.6820 accuracy score in 4739.08 seconds with the multi-color mode. The best baseline method performed a 0.4532 accuracy score in 48.50 seconds in single-color mode and a 0.3733 accuracy score in 42.09 seconds in multi-color mode. The summary is shown in Table 20. (see Appendix A for the details)

Table 20 Refined dataset improvement run result summary

| | | | | Precision | Recall | F1 Score | Accuracy | Support |
|---|---|---|---|---|---|---|---|---|
| **Single Color** | CNN AlexNet Model | RGB | macro avg | **0.7226** | **0.5986** | **0.6217** | **0.8383** | 2721 |
| | | | weighted avg | **0.8351** | **0.8383** | **0.8335** | | |
| | | HSL | macro avg | 0.5997 | 0.5624 | 0.5721 | 0.7986 | |
| | | | weighted avg | 0.7989 | 0.7986 | 0.7966 | | |
| | Best Baseline Method | RGB | macro avg | 0.4321 | 0.2432 | 0.1965 | 0.4331 | 13596 |
| | | | weighted avg | 0.5888 | 0.4331 | 0.3537 | | |
| | | HSL | macro avg | 0.5333 | 0.2753 | 0.2369 | 0.4532 | |
| | | | weighted avg | 0.5989 | 0.4532 | 0.3727 | | |

| | | | | Precision | Recall | F1 Score | Accuracy | Support |
|---|---|---|---|---|---|---|---|---|
| **Multi-Color** | CNN AlexNet Model | RGB | macro avg | **0.6330** | **0.5940** | **0.5975** | **0.7279** | 3142 |
| | | | weighted avg | **0.7151** | **0.7279** | **0.7157** | | |
| | | HSL | macro avg | 0.5848 | 0.5306 | 0.5521 | 0.6820 | |
| | | | weighted avg | 0.6818 | 0.6820 | 0.6793 | | |
| | Best Baseline Method | RGB | macro avg | 0.3868 | 0.2197 | 0.1670 | 0.3664 | 15710 |
| | | | weighted avg | 0.4576 | 0.3664 | 0.2827 | | |
| | | HSL | macro avg | 0.2747 | 0.2530 | 0.2056 | 0.3733 | |
| | | | weighted avg | 0.3279 | 0.3733 | 0.3027 | | |

## 4.8. Discussion

The best performance for the automatic validation model is conducted in the RGB color space. The model is ~0.03 more accurate in RGB color space. The recall score for invalid image class is the worse in the experiment, which is 0.8640 and 0.8400 for RGB and HSL color spaces, respectively. This shows us the model has worse performance while predicting invalid images. In contrast, the model has 0.9674 and 0.9442 recall scores for valid images, showing that the model performs better while predicting valid images.

The accuracy scores of all the runs of different settings are summarized in Table 21.

Table 21 Accuracy results summary

| | | | **Automatic Validation Model** | **Unverified Dataset** | **Manually Verified Dataset** | **Refined Dataset** |
|---|---|---|---|---|---|---|
| **Single Color** | CNN AlexNet Model | RGB | **0.9294** | **0.7891** | **0.7806** | **0.8383** |
| | | HSL | | 0.7540 | 0.7041 | 0.7986 |
| | Best Baseline Method | RGB | X | 0.4076 | 0.3347 | 0.4331 |
| | | HSL | | 0.4277 | 0.3609 | 0.4532 |
| **Multi-Color** | CNN AlexNet Model | RGB | 0.9059 | **0.6909** | **0.6651** | **0.7279** |
| | | HSL | | 0.6506 | 0.6193 | 0.6820 |
| | Best Baseline Method | RGB | X | 0.3468 | 0.2873 | 0.3664 |
| | | HSL | | 0.3535 | 0.3068 | 0.3733 |

In general terms, CNN AlexNet models performed better in RGB color space, while baseline methods scored better in HSL color space. The accuracy difference between color spaces in both settings is around ~0.03 except for the manually verified dataset run in the CNN AlexNet model, which is ~0.08 for single color and ~0.05 for multi-color. When we compare confusion matrices of manually validated datasets run with single color settings for RGB and HSL color space (see Figure 18), the classification accuracy dropped significantly for white and black.

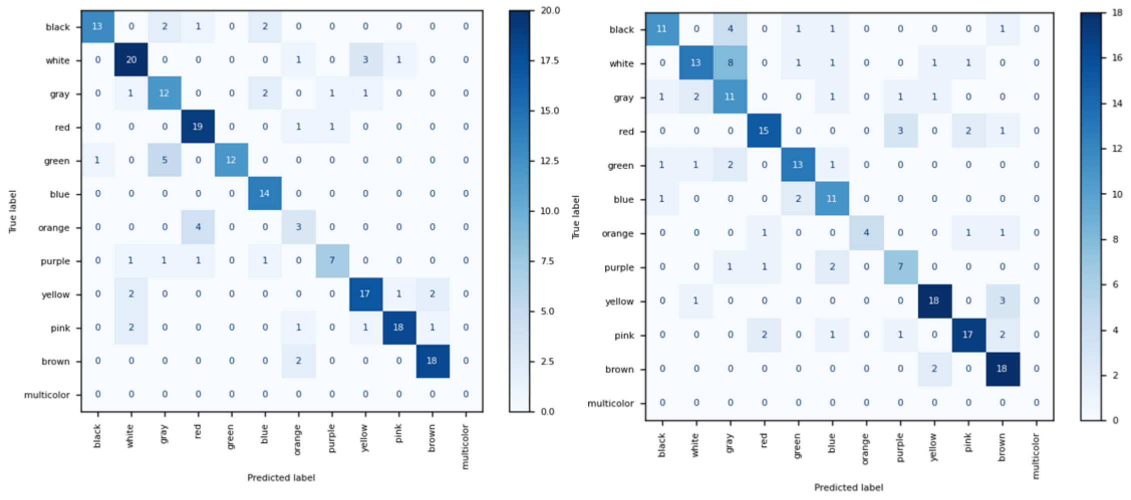


Figure 18 Comparing RGB (Right) HSL (Left) color space performance for the manually validated dataset in a single color setting

The most significant drop is in the gray class. The precision score for gray is dropped to 0.4231. When we compare two confusion matrices for the gray class, we see the HSL model misclassified 12 black and white shoes as gray. Four of them are actually black, and 8 of them are white. When comparing the failed classification samples of two white shoes (Figure 19), we see that these misclassified images are highly white intense shoes. The model predicts these images as gray in HSL color space.

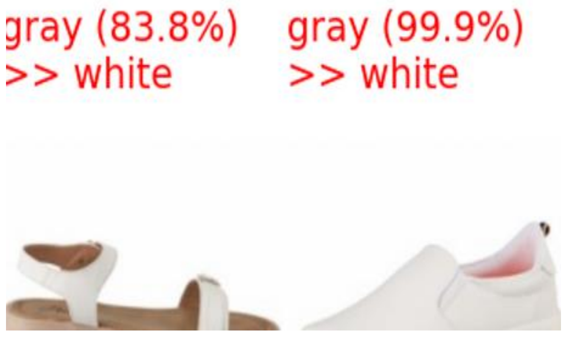


Figure 19 Misclassified Image Sample (Gray)

The automatic validation model removed 26.77% of images (5744 invalid images out of 21454 total images) as invalid in the unrefined dataset. This ratio was similarly 25.29% in the expert decision sessions. After this removal, the accuracy rate of the CNN AlexNet Model and baseline methods are increased by ~0.04 after automatic image validation is applied to the unrefined dataset.

The best accuracy score is 0.8383 with the refined dataset, RGB color space on a single color setting. All of these accuracy scores are calculated towards the defined classes in the dataset. But, we recorded a significant count of images misclassified in the raw dataset according to the outputs of the manual validation process. We examined 2416 images in total; 609 are invalid, 715 are misclassified, and 1092 are valid. The misclassified image ratio is 29.59% (715 images out of 2416) on a multi-color included setting. When we ignore the multi-color class, the ratio becomes 29.71% (654 images out of 2201 images total). These ratios include the invalid images in the total image numbers when we remove the invalid images from the total image counts: the ratio increases to 39.56% for multi-color (715 over 1807), 39.73% (654 over 1646) for single color setting.

When we look at the randomly chosen failed 36 sample images from the run (see Figure 20), 17 out of 36 sample images are misclassified, and one is invalid. The misclassified sample ratio is 47.22%, and the invalid sample ratio is 3.12%.

According to these numbers, we can calculate a rough prediction about the real accuracy: The count of total images used in the test database for the run is 2721. The model predicted 2281 of them successfully. The remaining image count is 440, which is not classified correctly by the model. According to the sample statistics, 208 of these images are misclassified (47.22%). On the other hand, the ratio calculated over the manual validation process is that 39.73% of images are misclassified. This means 1081 images out of 2721 are misclassified. So, 811 correctly classified images (1079 – 268) are also misclassified. In the end, we can predict the actual accuracy:

$$accuracy' = \frac{2281 - (1081 - 208)}{2721 - 1081} = \frac{1408}{1640}$$

$$accuracy' \cong 0.8585$$

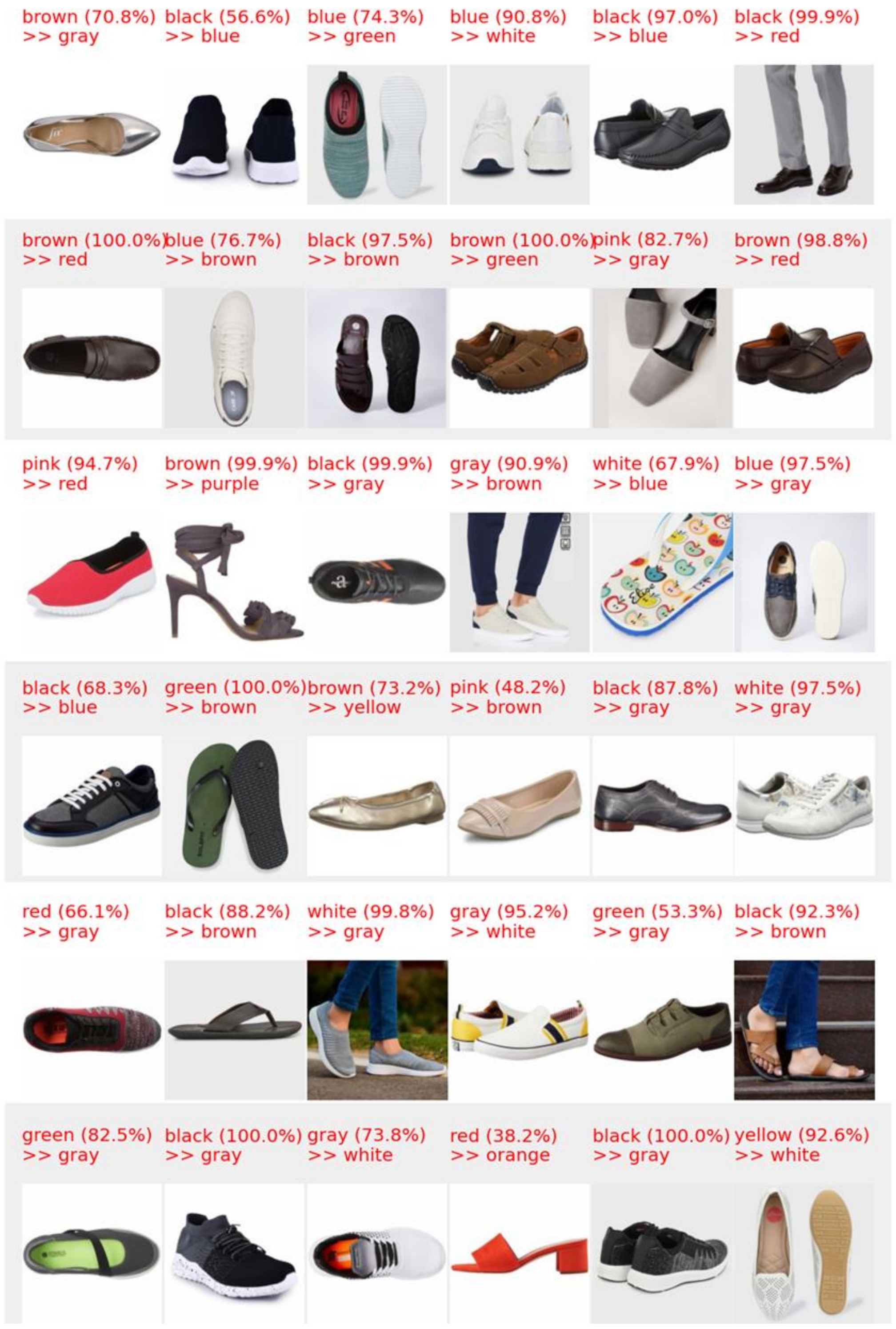


Figure 20 Sample Images that cannot be accurately predicted in the test dataset of the best model

According to this rough calculation, the prediction accuracy is increased by ~0.02. But it is harder to calculate the actual accuracy precisely without validating every image. Also, human subjectivity affects the outputs if we have enough capacity to validate all the output images manually.

On the other hand, if we consider whether the misclassification exists in the raw dataset or not, the CNN AlexNet model performs almost two times more accurately than the baseline models. When we examine the confusion matrix of the best CNN AlexNet model in Figure 21, we can verbalize some key points according to the model's outputs.

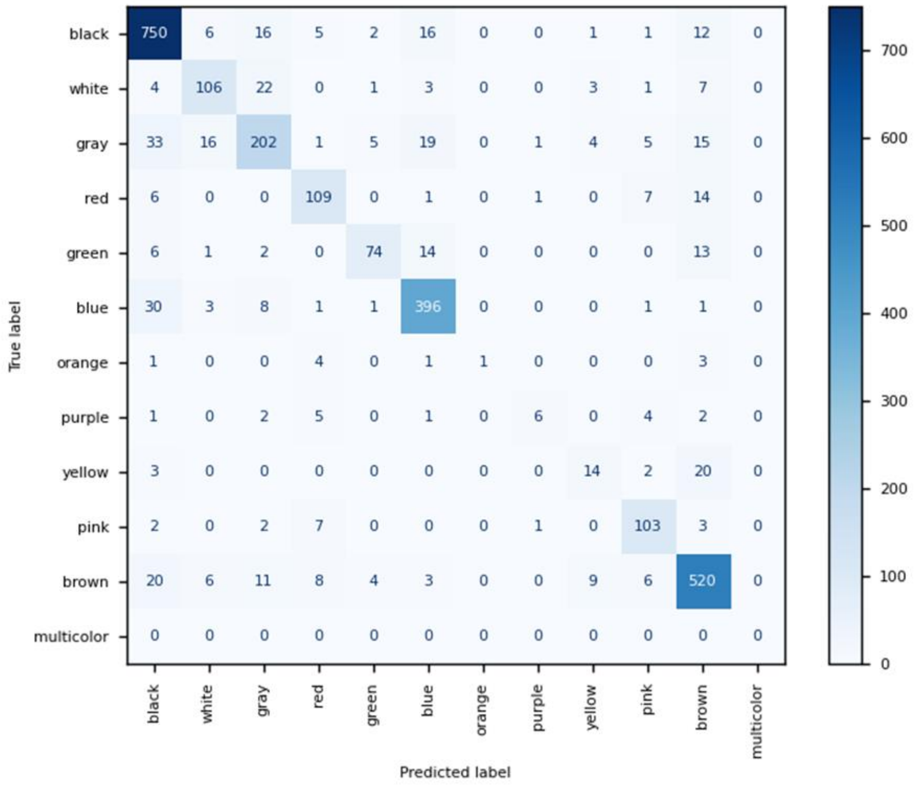


Figure 21 Confusion matrix of the best model

The classification report of the best model is shown in Table 22.

Table 22 Classification Report of the best model

| | Precision | Recall | F1 Score | Support |
|---|---|---|---|---|
| **black** | 0.8762 | 0.9271 | 0.9009 | 809 |
| **white** | 0.7681 | 0.7211 | 0.7439 | 147 |
| **gray** | 0.7623 | 0.6711 | 0.7138 | 301 |
| **red** | 0.7786 | 0.7899 | 0.7842 | 138 |
| **green** | 0.8506 | 0.6727 | 0.7513 | 110 |
| **blue** | 0.8722 | 0.8980 | 0.8849 | 441 |
| **orange** | 1.0000 | 0.1000 | 0.1818 | 10 |

Table 23 (cont.) Classification Report of the best model

| | Precision | Recall | F1 Score | Support |
|---|---|---|---|---|
| **purple** | 0.6667 | 0.2857 | 0.4000 | 21 |
| **yellow** | 0.4516 | 0.3590 | 0.4000 | 39 |
| **pink** | 0.7923 | 0.8729 | 0.8306 | 118 |
| **brown** | 0.8525 | 0.8859 | 0.8688 | 587 |
| **multicolor** | 0.0000 | 0.0000 | 0.0000 | 0 |
| | | | | |
| **micro avg** | 0.8383 | 0.8383 | 0.8383 | 2721 |
| **macro avg** | 0.7226 | 0.5986 | 0.6217 | 2721 |
| **weighted avg** | 0.8351 | 0.8383 | 0.8335 | 2721 |

The model performs well for the black, white, gray, red, green, blue, pink, and brown classes. A few gray class predictions mixed with black, white, and blue class, and yellow class predictions mixed with brown classes. The model could not perform accurate predictions in the orange and purple classes. Almost all the predictions for these two classes are not correct.

Now, we can compare those outputs with the best baseline method. The confusion matrix of the Refined Dataset Pairwise Argmin method is shown in Figure 22.

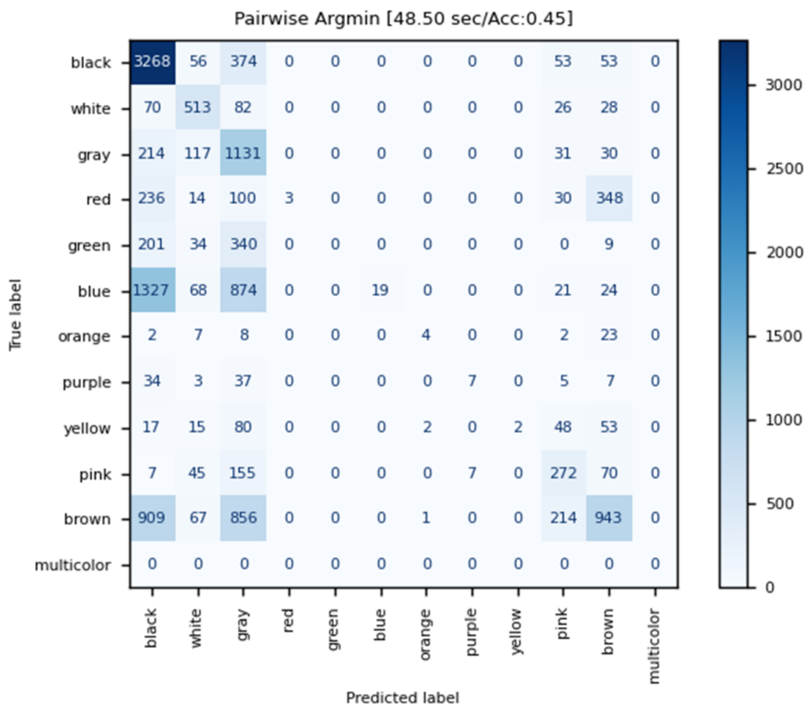


Figure 22 Refined dataset pairwise argmin confusion matrix (single color with HSL color space)

The best baseline method shown in Figure 22 performed with a 0.4532 accuracy score. The model predicts almost all the images as black, white, or gray according to the confusion matrix and the outputs (see Appendix A). The recall score for black is 0.8591, while the precision score is 0.5200. The recall score for white is 0.7135, while the precision score is 0.5463. The recall score for gray is 0.7426, while the precision score is 0.2802.

For the comparison of the accuracy of the baseline method over the color spaces. The methods that use HSL color space performed ~0.025 better than the methods that use RGB color space in all experiments.

According to the accuracy figures in Table 21, single color models performed better than the multi-color models, ~0.10 for the CNN AlexNet models and ~0.07 for the baseline models. The most significant difference between single color and multi-color is on CNN AlexNet model with HSL Color Space is 0.1166. The divergence between these two methods is based on the calculation of the multi-color. In CNN AlexNet models, the multi-color class is calculated with the images defined as multi-color in the dataset. On the other hand, the baseline methods identify an image as multi-color if more than one probability of color classes has a probability higher than 25%. So, the CNN AlexNet model relies on the predefined classes in the dataset, which have ~29% and ~39% misclassification ratios on unrefined and refined datasets, respectively. This misclassification ratio has a negative effect on output accuracy.

# CHAPTER 5

# CONCLUSION

The marketplaces have millions of products listed with tens of millions of metadata to validate and correct. However, they also do not have a validated ground truth that helps them train the models that can be used in the automated validation processes. In this study, we aim to propose a new way to create a ground truth from the unrefined bulk dataset, which is our first objective. Our second objective in this research is to offer a new model to be used in the validation process that gets images of the products and validates the metadata.

We proposed a new methodology, including automatic validation and classification for product metadata using product images with a deep neural network-based model. This approach could be applied to different metadata values in various product categories. The trained models could be used to validate newly uploaded metadata values for a product in an automated fashion. While the accuracy and reliability of product metadata are essential for marketplaces, this process helps them give their customers a better experience over product search and filtering. Inaccurate or unreliable product metadata can lead to issues like poor customer satisfaction. By improving the product metadata, the new methodology has the potential to significantly improve these business processes and ultimately lead to increased efficiency and profitability.

To see the performance of our methodology, we used 21,454 shoe images and respective metadata from the ABO dataset and performed 26 experiments over the dataset with the proposed CNN AlexNet model and statistical baseline methods. We aimed to validate the predefined color metadata of the shoe products in those experiments in two color spaces, RGB and HSL, and include a multi-color class option. In the first step, we ran our model without any refinement on the raw dataset; then, we selected one hundred valid images per color manually and created a validation model based on CNN AlexNet to create a refined dataset by validating the images from the raw dataset. Then we reran our models on this refined dataset and manually validated the dataset for comparison. In the end, our proposed model achieved 83.83% accuracy in the best settings, while the best baseline method achieved 45.32% accuracy.

For future work, the additional color spaces could be used to see their effectiveness in classification. For example, XYZ, YUV, or CIELAB color spaces could be used in future work. These additional color spaces could be helped the performance of the proposed model.

Another point is that we use the multi-color class like another regular color in the CNN AlexNet model. But, in the baseline methods, we take an image as multi-color if it has more than one color with a higher probability of over 25%. These two approaches are not aligned with each other. As another future work, we can ignore the defined multi-color classes in the dataset and create a similar multi-color decision process in the CNN AlexNet model. Also, we can use ROC Curve and F1 score to find the best threshold for the multi-color decision.

Image augmentation is not applied to the input dataset during the metadata improvement CNN AlexNet training. This process could be applied in the future to create additional training images and increase the model accuracy. Similarly, for the automatic image validation model training, additional invalid images could be augmented or generated to increase the input image count and model accuracy.

In conclusion, the proposed new product metadata validation methodology has proven to significantly improve the baseline methods. It has been shown through extensive testing and evaluation. The significant increase in performance demonstrates the effectiveness and efficiency of the new methodology in validating product metadata. The proposed new product metadata validation methodology is a valuable contribution to the field. Its high accuracy rate and potential to improve business processes make it an attractive solution for various industries. Further research and development of the new methodology have the potential to bring even greater accuracy and efficiency to product metadata validation.

# REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bergstra, J., Bardenet, R., … Y. B.-A. in neural, & 2011, undefined. (n.d.). Algorithms for hyper-parameter optimization. *Proceedings.Neurips.Cc*. Retrieved April 13, 2023, from https://proceedings.neurips.cc/paper/4443-algorithms-for-hyper-parameter-optimization

Bergstra, J., Yamins, D., on, D. C.-I. conference, & 2013, undefined. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Proceedings.Mlr.Press*, *28*. https://proceedings.mlr.press/v28/bergstra13.html

Blanco, C. F., Sarasa, R. G., & Sanclemente, C. O. (2010). Effects of visual and textual information in online product presentations: Looking for the best combination in website design. *European Journal of Information Systems*, *19*(6), 668–686. https://doi.org/10.1057/EJIS.2010.42

Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T. F. Y., Dideriksen, T., Arora, H., Guillaumin, M., & Malik, J. (2021). *ABO: Dataset and Benchmarks for Real-World 3D Object Understanding*. 21094–21104. https://doi.org/10.48550/arxiv.2110.06199

Deng, Z., Chen, W. Te, Chen, L., & Yu, P. S. (2022). AE-smnsMLC: Multi-Label Classification with Semantic Matching and Negative Label Sampling for Product Attribute Value Extraction. *Proceedings - 2022 IEEE International Conference on*

*Big Data, Big Data 2022*, 1816–1821. https://doi.org/10.1109/BIGDATA55660.2022.10020304

Gervautz, M., & Purgathofer, W. (1988). A Simple Method for Color Quantization: Octree Quantization. *New Trends in Computer Graphics*, 219–231. https://doi.org/10.1007/978-3-642-83492-9_20

Ghani, R., Probst, K., Liu, Y., Krema, M., & Fano, A. (2006). Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter, 8*(1), 41–48. https://doi.org/10.1145/1147234.1147241

Ghosh, P., Wang, N., & Yenigalla, P. (2023). D-Extract: Extracting Dimensional Attributes From Product Images. *Proceedings - 2023 IEEE Winter Conference on Applications of Computer Vision, WACV 2023*, 3630–3638. https://doi.org/10.1109/WACV56688.2023.00363

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gundimeda, V., Murali, R. S., Joseph, R., & Naresh Babu, N. T. (2019). An automated computer vision system for extraction of retail food product metadata. *Advances in Intelligent Systems and Computing*, *815*, 199–216. https://doi.org/10.1007/978-981-13-1580-0_20/TABLES/8

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December,* 770–778. https://doi.org/10.48550/arxiv.1512.03385

*How can I calculate distance from two pixels HSV?* (2022). Mathematics Stack Exchange. https://math.stackexchange.com/questions/4016084/how-can-i-calculate-distance-from-two-pixels-hsv

*HSL and HSV - Wikipedia*. (2022). https://en.wikipedia.org/wiki/HSL_and_HSV

Hyndman, R. J., & Fan, Y. (1996). Sample Quantiles in Statistical Packages. *American Statistician, 50*(4), 361–365. https://doi.org/10.1080/00031305.1996.10473566

Joblove, G. H., & Greenberg, D. (1978). Color spaces for computer graphics. *Computer Graphics*, *12*(3), 20–25. https://doi.org/10.1145/965139.807362

Kim, M., & Lennon, S. (2008). The effects of visual and verbal information on attitudes and purchase intentions in internet shopping. *Psychology and Marketing*, *25*(2), 146–178. https://doi.org/10.1002/MAR.20204

Krizhevsky, A., & Inc, G. (2014). *One weird trick for parallelizing convolutional neural networks*. https://doi.org/10.48550/arxiv.1404.5997

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Kumar, K., & Saladi, A. (2022a). PAVE: Lazy-MDP based Ensemble to Improve Recall of Product Attribute Extraction Models. *International Conference on Information and Knowledge Management, Proceedings*, 3233–3242. https://doi.org/10.1145/3511808.3557119

Kumar, K., & Saladi, A. (2022b). PAVE: Lazy-MDP based Ensemble to Improve Recall of Product Attribute Extraction Models. *International Conference on Information and Knowledge Management, Proceedings*, 3233–3242. https://doi.org/10.1145/3511808.3557119

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/NECO.1989.1.4.541

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

le Yu Haozheng Tian, Y. Z. S. H., & Velkoski, A. (2022). *E-commerce Product Attribute Value Validation and Correction Based on Transformers*.

Li, M., Wei, K. K., Tayi, G. K., & Tan, C. H. (2016). The moderating role of information load on online product presentation. *Information & Management*, *53*(4), 467–480. https://doi.org/10.1016/J.IM.2015.11.002

Lin, R., He, X., Feng, J., Zalmout, N., Liang, Y., Xiong, L., & Dong, X. L. (2021). PAM: Understanding Product Images in Cross Product Category Attribute Extraction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 3262–3270. https://doi.org/10.1145/3447548.3467164

Lundervold, A. S., & Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift Für Medizinische Physik*, *29*(2), 102–127. https://doi.org/10.1016/J.ZEMEDI.2018.11.002

Marr, D. (1982). Vision: A Computational Investigation of Visual Representation in Man. *Phenomenology and the Cognitive Sciences*, *8*(4), 397. http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&amp;tid=12242&amp;ref=nf

Miami, F., & Zeng, K. (2015). *Next Generation of Product Search and Discovery: Visual Search and Recommendation*.

*optuna.samplers.TPESampler — Optuna 3.1.0 documentation*. (n.d.). Retrieved April 13, 2023, from https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html

Orchard, M. T., Bouman, C. A., & others. (1991). Color quantization of images. *IEEE Transactions on Signal Processing*, *39*(12), 2677–2690.

Ozaki, Y., Tanigaki, Y., Watanabe, S., Nomura, M., & Onishi, M. (2022). Multiobjective Tree-Structured Parzen Estimator. *Journal of Artificial Intelligence Research*, *73*, 1209–1250. https://doi.org/10.1613/JAIR.1.13188

Ozaki, Y., Tanigaki, Y., Watanabe, S., & Onishi, M. (2020). Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 533–541. https://doi.org/10.1145/3377930.3389817

Petrovski, P., & Bizer, C. (2017). Extracting Attribute-Value Pairs from Product Specifications on theWeb. *Proceedings - 2017 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2017*, *8*, 558–565. https://doi.org/10.1145/3106426.3106449

Qiu, D., Barbosa, L., Dong, X. L., Shen, Y., & Srivastava, D. (2015). DEXTER: Large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment*, *8*(13), 2194–2205. https://doi.org/10.14778/2831360.2831372

Rezk, M., Alonso Alemany, L., Nio, L., & Zhang, T. (2019). Accurate product attribute extraction on the field. *Proceedings - International Conference on Data Engineering*, *2019-April*, 1862–1873. https://doi.org/10.1109/ICDE.2019.00202

*RGB color model - Wikipedia*. (2022). https://en.wikipedia.org/wiki/RGB_color_model

Sabeh, K., Kacimi, M., & Gamper, J. (2022). CAVE: Correcting Attribute Values in E-commerce Profiles. *International Conference on Information and Knowledge Management, Proceedings*, 4965–4969. https://doi.org/10.1145/3511808.3557161

Schwarz, M. W., Cowan, W. B., & Beatty, J. C. (1987). An experimental comparison of RGB, YIQ, LAB, HSV, and opponent color models. *ACM Transactions on Graphics*, *6*(2), 123–158. https://doi.org/10.1145/31336.31338

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://doi.org/10.48550/arxiv.1409.1556

Suzuki, S., & be, K. A. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46. https://doi.org/10.1016/0734-189X(85)90016-7

Vandoni, C. E. (1996). *Computer Vision : Evolution And Promise*. CERN. https://doi.org/10.5170/CERN-1996-008.21

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience, 2018*. https://doi.org/10.1155/2018/7068349

Wang, Q., Yang, L., Kanagal, B., Sanghai, S., Sivakumar, D., Shu, B., Yu, Z., & Elsas, J. (2020a). Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 47–55. https://doi.org/10.1145/3394486.3403047

Wang, Q., Yang, L., Kanagal, B., Sanghai, S., Sivakumar, D., Shu, B., Yu, Z., & Elsas, J. (2020b). Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 47–55. https://doi.org/10.1145/3394486.3403047

Wang, Y., Xu, Y. E., Li, X., Dong, X. L., & Gao, J. (2020). Automatic Validation of Textual Attribute Values in E-commerce Catalog by Learning with Limited Labeled Data. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 20,* 2533–2541. https://doi.org/10.48550/arxiv.2006.08779

Wong, T. L., Lam, W., & Wong, T. S. (2008). An unsupervised framework for extracting and normalizing product attributes from multiple web sites. *ACM SIGIR 2008 - 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Proceedings*, 35–42. https://doi.org/10.1145/1390334.1390343

Yang, L., Wang, Q., Yu, Z., Kulkarni, A., Sanghai, S., Shu, B., Elsas, J., & Kanagal, B. (2022a). MAVE: A product dataset for multi-source attribute value extraction. *WSDM 2022 - Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1256–1265. https://doi.org/10.1145/3488560.3498377

Yang, L., Wang, Q., Yu, Z., Kulkarni, A., Sanghai, S., Shu, B., Elsas, J., & Kanagal, B. (2022b). MAVE: A product dataset for multi-source attribute value extraction. *WSDM 2022 - Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1256–1265. https://doi.org/10.1145/3488560.3498377

Zhang, X., Zhang, C., Li, X., Dong, X. L., Shang, J., Faloutsos, C., & Han, J. (2022). OA-Mine: Open-World Attribute Mining for E-Commerce Products with Weak Supervision. *WWW 2022 - Proceedings of the ACM Web Conference 2022*, 3153–3161. https://doi.org/10.1145/3485447.3512035

Zhao, M., Hoeffler, S., & Dahl, D. W. (2009). The role of imagination-focused visualization on new product evaluation. *Journal of Marketing Research*, *46*(1), 46–55. https://doi.org/10.1509/JMKR.46.1.46

Zheng, G., Mukherjee, S., Dong, X. L., & Li, F. (2018). OpenTag: Open aribute value extraction from product profiles. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1049–1058. https://doi.org/10.1145/3219819.3219839

## APPENDICES

## APPENDIX A

* Find the technical report attached, or download it from https://github.com/alatas/MSThesis