Cairo University
Faculty of Engineering
Credit Hours System
Communication and Computer Engineering
CCEN481: Graduation Project – 2
Spring 2019

Shopping Pal

A SHOPPING ASSISTANT APPLICATION

Team Members

Hussein Mohamed Hesham Hosny 1141308
Mohamed Osama Hussein Eleiwa 1142032
Salma Ibrahim Ahmed Mohamed 1144055
Yousra Hussein Ibrahim El-Sherbiny 1142072

Supervised By

Dr. Samir Shahin

<table>
<tr><td>Project Title</td><td colspan="2">Shopping Pal</td></tr>
<tr><td>Keywords</td><td colspan="2">Fully Convolutional Network (FCN), Texture Cues, Stream Merging, Association Rule Mining, Clothing Parsing, Compositional Network, NoSQL.</td></tr>
<tr><td rowspan="2">Students</td><td>Name: Hussein Mohamed Hesham
E-mail: HussainHHosny@gmail.com
Phone: 01006989941
  </td><td>Name: Mohamed Osama Hussein
E-mail: mossama897@gmail.com
Phone: 01002596640
  </td></tr>
<tr><td>Name: Salma Ibrahim Ahmed
E-mail: Salmaaibrahim96@gmai.com
Phone : 01142774275
  </td><td>Name: Yousra Hussein Ibrahim
E-mail: yousra-hussein@outlook.com
Phone: 01004002528
  </td></tr>
<tr><td>Supervisor</td><td colspan="2">Name: Dr. Samir Shahin
E-mail: sshaheen@eng.cu.edu.eg</td></tr>
<tr><td>Project Summary</td><td colspan="2">An application, supported by a set of artificial intelligence modules, that a user can use; to discover the trends nowadays, to find similar pieces to any piece provided by the user, to get recommendations for outfits suitable for any occasion, and finally, to get recommendations for other pieces that match (go with) another given piece provided by the user. The application can take any image as an input from the user and perform the mentioned functionalities on this photo. For example, the user can take an image of a yellow blouse and choose to get recommendatins on other pieces that can go with this blouse that follows the global trend in fashion.</td></tr>
</table>

Acknowledgement

First and foremost, we would like to thank **Dr. Samir Shahin** for his wonderful regulation, guidance, encouragement, management and definitely patience with us over our progress. His insight has guided us and driven our project to what it is. His encouragement throughout our project's inception, design, and final implementation, and his eagerness to help us deliver a high-quality product, prove we were fortunate to have him as our advisor. He exerted much effort starting from helping us to develop the basic idea of the project up till the final touches of the implementation.

In addition to that, we want to thank the **teaching staff of Faculty of Engineering, Cairo University**, who gave us everything within and outside the classrooms. We are very glad our paths have crossed with some of the greatest minds in the entire country; we learned so much from them on both academic and personal levels.

We would also love to thank **our families** for their never ending support, encouragement and prayers without which we would not have passed many hardships.

We thank **God** above all for putting the people above in our paths and lives, and for empowering us throughout our work throughout the year on this project. We have faced many challenges in this project that required true strength to face. That strength could only have come from God and His plans for us.

Abstract

Shopping and clothing nowadays are considered as main parts of every person's life. A person will always find himself/herself making a choice for an outfit for a certain occasion, and thinking how to match his/her outfit. In addition, a lot of time is spent on searching for a similar piece that a person could have seen before whether on the internet, or in a store but with a (high price/ bad quality). Furthermore, there is a large group of people interested in being up to date with the trending colors, cuts and pieces to cope with the fashion changes. All these activities are time consuming, can be disappointing when certain pieces or styles are not found, and not to mention energy draining at a lot of occasions. That has motivated us to build an application that helps people carry out the mentioned activities, but faster thus saving a lot of time, and more efficiently.

Our proposed solution is an application that a user can use; to discover the trends nowadays, to find similar pieces to any piece provided by the user, to get recommendations for outfits suitable for any occasion, and finally, to get recommendations for other pieces that match (go with) another given piece provided by the user.

It's as if one is shopping and browsing through fashion pieces, that match his/her specific taste, at tens of stores at the same exact moment so the user would be exposed to a larger variety of fashion pieces (selected to be viewed to each customer separately depending on his/her taste and on the trend) to satisfy any need he/she may have.

Table of Contents

List of Figures

List of Tables

Table of Acronyms

ABBREVIATION	DEFINITION
FCN	Fully Convolutional Networks
CNN	Convolutional Neural Networks
AI	Artificial Intelligence
CONVNET	Convolutional network
LBP	Local Binary Pattern
OHE	One Hot Encoding
CONV	Convolutional Layer
UI	User Interface
UX	User Experience
HSL	Hue-Saturation-Lightness
HSV	Hue-Saturation-Value
RGB	Red-Green-Blue
NLEFT	Number of left instances
NRIGHT	Number of right instances
NTOTAL	Number of left and right instances
SQL	Structured Query Language
NOSQL	Not only SQL
CSV	Comma-separated values

1. Introduction

1 Introduction

1.1 Problem Definition

We all know that shopping and clothing is a main part of a person's daily life, but somehow not every person knows how to match their outfits, how to dress for a certain event, and how to buy a certain piece of clothes that match other pieces that they already have. Moreover, nearly every person has seen before either a shirt, a pair of shoes, a bag on the internet, on a model, on a friend or even on a stranger that he/she likes so much but he/she does not know from where he can buy it. He would surf most of the online shopping websites searching for this piece. This process is time consuming and he/she might not necessarily find it after wasting this time. Another problem is that sometimes people find a piece of clothes that they like but they cannot afford its price, so they start searching for a similar piece in another store with a more reasonable price.

Furthermore, a lot of people just don't have the eye to match different fashion pieces together to form an outfit. It may be a person who's not into fashion, indecisive person who can't choose a blouse that goes with those pants, or even a color blind person who can't even see the colors to match a pair of socks together let alone match an outfit.

1.2 Project Idea

Our idea is mainly having a daily- used application that the user can always use whenever he's having any issue related to clothes or fashion. In other words, our idea was to make an application that can be used as a guide for the user in anything related to shopping and fashion. So, we came up with "Shopping Pal", which is an application that has the following main functionalities:

1) **Piece Retrieval**

Description:

The application user can add an image of a piece of clothes, the application retrieves to him/her all the similar pieces in all the shopping stores that are

registered in our application. This process is done by matching the desired piece with the available pieces in the shopping stores using specific features like: color, texture, length, shape and others.

Input:

Image of desired piece.

Output:

All the pieces in the shopping stores that is similar to the desired piece.

2) Outfit Recommendations

Description:

The application user can request recommendations/suggestions for outfits that is suitable for a certain event like: birthday, breakfast outing, formal meetings, university outfits, …etc.

Input:

Chosen Event

Output:

Recommended outfits

3) Piece Matching

Description:

The application user can give the application a piece of clothes of his own as an input and ask the application to recommend other pieces that match with the given piece.

Input:

An image of a shirt.

Output:

Jackets, Pants, Shoes, Bags, Scarfs, etc. that match with this shirt. Every recommended piece appears with the shopping store from which you can buy it.

1.3 Motivation and Justification

1.3.1 Motivation

The variety of fields of computer engineering used in the design and implementation of the application will not only be useful to our intended customers but can also be considered as a luxury and a source of joy (That may occur when the application recommends an outfit that the customer never would have looked for but immediately likes it). Examples of such computer engineering fields: Artificial Intelligence, Image Processing, Recommendation Systems, Mobile Development, etc.

Moreover, this project is that it is not directed to a certain sector of people, but solves an issue faced by nearly every person. Our solution to the problem is unique and it was not done before that is why our main motive is to propose an innovative solution that would have a large volume of audience.

Another important point is that shopping is moving nowadays towards online shopping instead of the traditional shopping; therefore, this application copes with the changing demographics. In addition, this application is extendible which means we can always add new features to it and extend existing ones.

Last but not least, from the previous points discussed in the motivation section, we would like to conclude the section with that our main motivation is to make people's lives easier, simpler and happier, while making a profit.

1.3.2 Educational Value

1) **Technical**

We implemented many of what we've learnt in different courses in the project like:

- Image processing and computer vision
- Big data analysis and data mining (association rules mining)
- Machine learning
- NOSQL using MONGODB
- Web crawling

We also learned about many different topics that we needed to use them in the project:

- Deep learning used in FCN and MNIST models
- CNN used in Similarity Retrieval model
- Mobile application development (full stack development) using React-Native and Express.js

2) **Non-Technical**

- Time management of large-scale projects
- Team work
- Task splitting
- Taking decisions
- Searching and information extraction
- Patience over technical and non-technical challenges
- Writing business plans and reviews

1.4 Domain of Applications

Our application now is mainly a mobile application. But it is made in a way that can be easily modified to be a website. We began to implement the frontend of it to be working as a website and it succeeded, but it's not ready yet.

It can be used by any person who is holding a smart phone. What is needed from the user is only to download the application and to have internet on his mobile.

As our application gives the user the option to buy clothes through it, so we also need to make deals with stores and online pages and websites.

Our application can be used by people who are interested in fashion and shopping stuff, fashion bloggers and influencers and anyone who would buy clothes.

1.5 Summary of Approach

After deciding on the idea, we made a lot of research to see how to achieve our goal in the best way. We found many research papers and implementations, and divided them among each other to read. After a while, we decided on some papers to take as a reference in our later work. In order to approach the results that we planned for, we thought of the different parts in the project and divided them in modules. These divided parts are:

1) **Datasets and their preprocessing**

   We searched for a dataset that can match with our requirements and we chose two datasets, each one contains images of people wearing some outfits and having a pixel level annotation with each image.

2) **Crawler/Scraper**

   We approached the best way of crawling the websites to extract the needed information about the products and we made this by scrapping the needed website by a “Scrapy” framework used in Python.

3) **Pieces Identification/Extraction**

   We identified the different pieces worn by a person from an image using a Fully Convolutional Neural Network that works by training it with the 2 datasets we’ve got.

4) **Features Extraction and Similarity Retrieval**

This is approached by a CNN VGNET model to take a picture of an outfit and returning some pictures that have similar overall features.

5) **Single Piece Type Identification**

This module is made to know the type of a single piece captured by a person (not worn) by using an MNIST model and training it by MNIST database.

6) **Matching Rules Generation**

We generated the matching rules to be used in the recommendation system by using the images crawled from the fashion bloggers profiles and performing an association rules mining operation to get matching pieces and colors from them.

7) **Interface and Integration**

We worked on the front-end and back-end in parallel. Which means that when we were implementing any page in the front end, we were implementing its functionalities at the backend, instead of implementing all of the front-end then implementing all the back-end.

1.6 Document Overview

In this book, we're presenting our work from all perspectives. The book consists of 7 chapters.

Chapter 1 is the introduction showing our main project idea, motivations that lead to it, problem definition and what we've learnt from the project technically and non-technically.

Chapter 2 is for the Market Survey, where it is used to investigate the market for our project. We used the market survey initially to ensure that people need this application and that it will help them. And we also used it to ask people for suggestions and recommendations for the application which helped us a lot. Chapter 2 is also for feasibility study for assessment of the practicality of our proposed project.

Chapter 3 is for stating the necessary background where it shows the general science behind the project and all the technical details related to it.

Chapter 4 is for explaining the design specification, assumptions made and requirements, and the block diagram that shows the flow and connections of the whole project.

Chapter 5 is for giving an insight about the languages, tools and libraries, and brief explanation about them.

Chapter 6 is for providing our test plan and results.

Chapter 7 is for the overview, conclusion and future work that we're planning to implement in the near future.

Finally, the references is included in the last chapter, chapter 8.

2. Market Survey & Feasibility Study

2 Market Survey & Feasibility Study

2.1 Introduction

This chapter is made to show the application end users, investigate the market and assess our position should we enter it. We have analyzed potential buyers, how the market is expanding, what are the key challenges, and similar products that already exist in the market and how we are going to gain a competitive advantage over them.

2.2 Customers

2.2.1 Intended Customers

This project is made for all the people who are interested in fashion and trends, and that's what make our project have a lot of audience. These audience in details are:

- **Application End Users**

  The application end users are the people who are interested to be aware with the trending colors, cuts, …etc. per season, interested to find pieces that match with a given piece, and interested to find similar pieces to a given piece.

- **Fashion Bloggers**

  Known fashion bloggers that are always interested to know the trends.

- **Color Blind People**

  Color blind people who cannot trust their eyes in matching colors with each other so they can depend on the application to match colors for them instead.

2.2.2 Potential Customers

These are the customers that our application still does not cover their needs. However, with a few modifications to the application, the application can easily cover their needs or even create a sub-product specific to those customers.

- **Fashion Designers**

Fashion designers can use the trend and matching detection algorithms we use to aid them in designing fashion pieces following the trend to attract more customers to those fashion designers.

2.3 Current Market

This section shows the nature of the current market that our project will be in, showing the similar products, competitors and substitutes. It presents functional, non-functional specification and limitations of each product.

Since shopping and clothing nowadays are considered as main parts of every person's life. And people are not only interested in wearing "good" clothes but they are trying to reach the "best" in terms of colors and trending outfits. All the stores, online shopping websites and any stakeholder included in the clothing and fashion cycle are doing their best to offer the end user what they want.

As a result of seeing all the people seeking for the best in fashion, all the stores, online shopping websites and any stakeholder included in the clothing and fashion cycle are doing their best to offer the end user what they want. And people thought of building services that help the customers in such things. We've searched for the products that are similar or close to our application and here are the ones we found:

1) **Deja Fashion**

Specifications:

- Show the fashion icons/fashion bloggers

- Show looks a person may like
- Gives you the option to shop from certain stores
- Makes you save favorite looks

Limitations:

- Fashion icons available are limited and are popular in certain countries only (Not popular in Egypt)
- Stores available are limited and not available in Egypt

2) liketoknowit

Specifications:

- Gives you the option to search by product name
- Gives you the option to search by influencer name
- Gives you the option to shop from certain stores
- Makes you save favorite looks
- Gives you the option to search for similar pieces

Limitations:

- Fashion icons available are limited and are popular in certain countries only (Not popular in Egypt)
- Stores available are limited and not available in Egypt
- Results of similar pieces are not accurate. They only get similar colors.

2.4 Market Survey Statistics

After deciding on the idea, we made a market survey in order to take people's opinion and know their suggestions and recommendations for the project. Here are some important questions from the survey:

- **Question 1:**

Have you ever experienced buying a piece of clothes and then did not find any pieces matching with it?

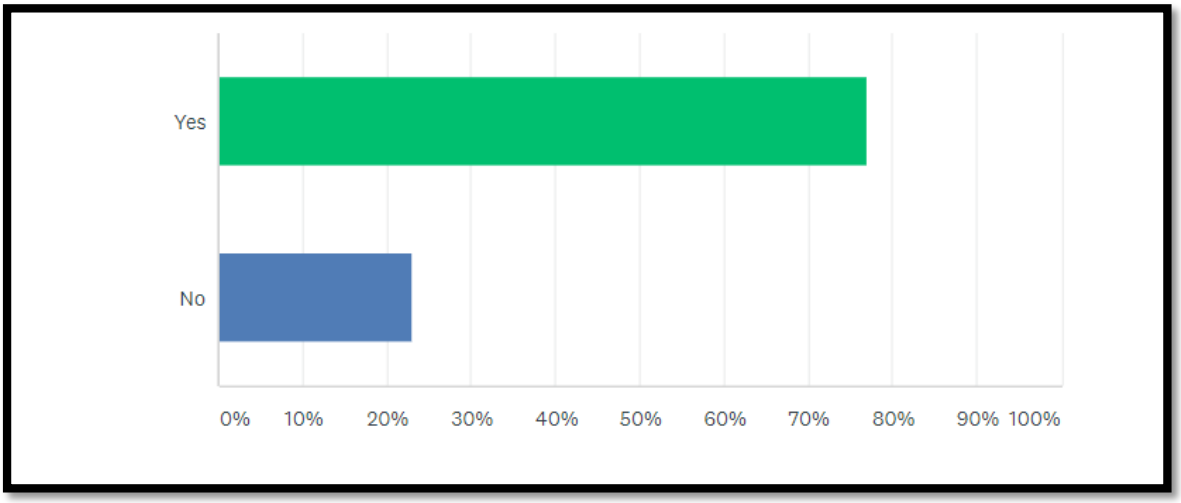


Figure 2-1 around 80% of the respondents face this shopping problem.

- **Question 2**

Have you ever experienced seeing a piece of clothes, a pair of shoes, a bag or a scarf that you like but did not know where to find it?

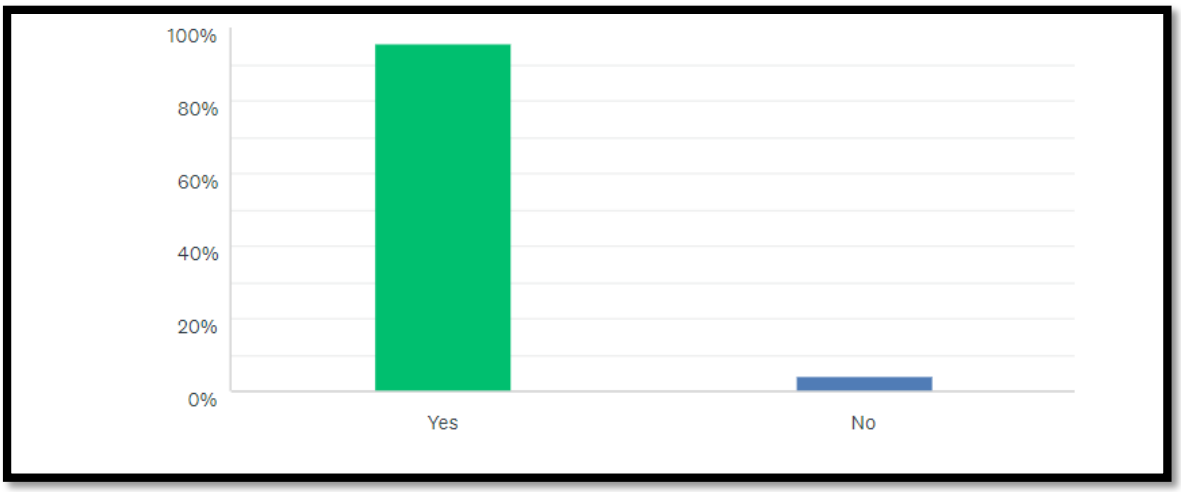


Figure 2-2 96% of the respondents face this shopping problem.

- **Question 3**

Would you like to have an application that aids you in the shopping problems described in the three previous questions?

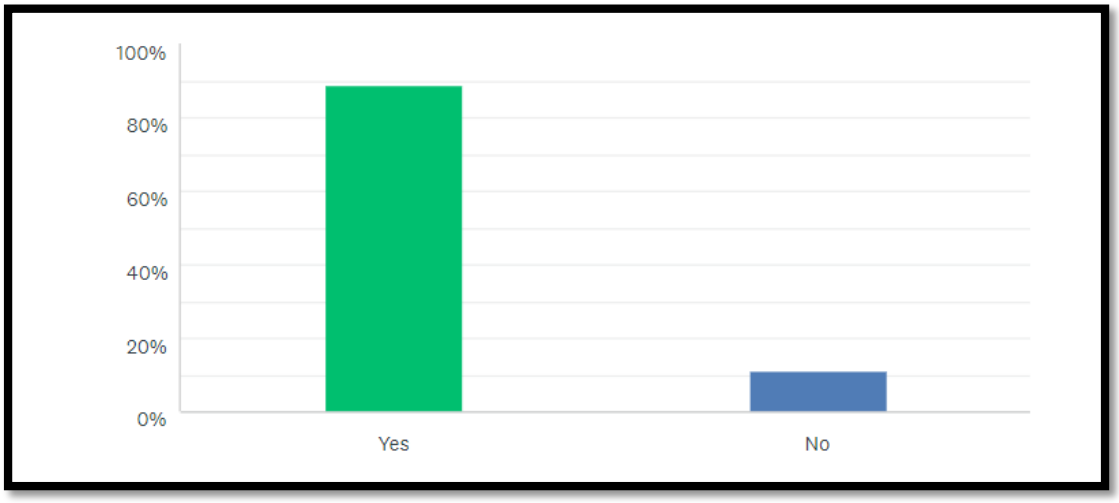


Figure 2-3 90% of the respondents will use the application to help them in the shopping problems.

2.5 Feasibility Study

In this section, we'll discuss the feasibility study needed in this project. Taking into account the product functionalities and provided services, we shall discuss the overall feasibility of the

project, the hardware and software requirements, economic feasibility from the cost and benefit side, schedule, marketing strategy and legal feasibility.

2.5.1 Description of Products & Services

The main services that our product offers are:

- Pieces matching from products found in stores and online websites available in Egypt
- Similarity retrieval from products found in stores and online websites available in Egypt
- Showing seasonal trends based on known fashion icons

2.5.2 Technological Feasibility

In order to implement each of the services mentioned in the previous section, we have some software requirements.

2.5.2.1 Software Requirements

1) **Datasets:**
    - Downloading CFPD and CNN datasets
2) **Frontend Requirements:**
    - React Native
    - Expo toolchain
    - VS Code
3) **Backend Requirements:**
    - Node.js
    - Express.js
    - JavaScript
    - Postman
    - MongoDB
    - VS Code
4) **Crawler Requirements:**
    - Scrapy Framework
    - Python

- MongoDB

5) **Neural networks and CNNs requirements**
   - Python 3.6 or above
   - Tensorflow
   - Browser (To use Google Colab)
   - OpenCV (CV2)
   - Scikit-image
   - Scikit-learn

2.5.2.2 Hardware Requirements

We're using google colab for training our models to be faster than the CPU in training, and to be able to handle large number of parameters without crashing

GPU Used: Tesla-k80 GPU

Figure 2-4 Tesla-k80

GPU Specifications:

- 4992 NVIDIA CUDA cores with a dual-GPU design
- Up to 2.91 teraflops double-precision performance with NVIDIA GPU Boost
- Up to 8.73 teraflops single-precision performance with NVIDIA GPU Boost
- 24 GB of GDDR5 memory

- 480 GB/s aggregate memory bandwidth
- ECC protection for increased reliability
- Server-optimised to deliver the best throughput in the data center

2.5.3 Economic Feasibility

2.5.3.1 Cost

1) **Hosting Cost**
   - Hosting on Google Play Store: A one-time fee of $25
   - Hosting on Apple App Store: $99/year
   - Which means that the hosting will cost $124 for the first year, then $99 for each year beginning from the second year.
2) **Development Cost:**

   We take $1 per hour. We work 4 months, 12 hour per day. So, the development time of each member is (30*12*4*1) = $1,440 per each developer

   Therefore, total development price for the first year is ($1,440 *4) = $5,760

   After the first year, the development work decreases and is substituted by maintenance and modifications.

3) **Servers Cost:**

   $1-3/month using AWS Free Tier limits.

4) **Total Cost:**

   $5,902 for the first year on average

2.5.3.2 Benefit

Our project is considered a base project for fashion, which has the main features we need for later modifications. We'll then be updating our project and adding some features to include all

the Egyptian stores. Our project is the base for a startup application that we'll e uploaded on both app store and play store.

2.5.4 Schedule Feasibility

In this section, we shall discuss the feasibility (time wise) of deploying our application before competitors emerge.

2.5.4.1 Past Time Plan

The below table shows how we have spent our time during the previous year. The output of that work (our current application) is not the version to be deployed. However, the work done in that time is the base for the application that is to be deployed.

	Week 1	**Week 2**	**Week 3**	**Week 4**
September	Searching for ideas and Online Courses	Searching for ideas and Online Courses	Searching for ideas and Online Courses	Searching for ideas and Online Courses
October	Filtering Ideas	Filtering Ideas	Filtering Ideas	Filtering Ideas
November	Research	Research	Defining project components and features	Defining project components and features
December	Defining project components and features	Graduation Project 1 Report	Graduation Project 1 Report	Final Exams
January	Final Exams	Final Exams	Research and Online courses	Research and Online courses

February	Phase 1	Phase 1	Phase 2	Phase 2
March	Phase 3	Phase 3	Phase 4	Phase 4
April	Phase 4	Phase 5	Phase 5	Phase 6
May	Phase 6	Phase 7	Final Exams	Final Exams

Phase 1

a) Search for important tags/categories in fashion. Example: materials, cuts, colors, print, type.
b) Search the websites of the suitable and selected brands (collected from the market survey) and their Instagram pages, and create custom crawlers to collect images.
c) Design GUI frontend.

Phase 2

a) Search for features to identify each fashion feature from an image. Example: texture to identify material, histogram of colors to identify color type.
b) Create database schema for images and their description, and build it.
c) Design database schema for matching fashion features.

Phase 3

a) Manually place tags on collected images from (1b) and build a database with data using schema from (2b) to obtain database "Manual".
b) Verify GUI design and add any improvements and build.
c) Design GUI backend.
d) Search for ready databases for training (data sets).

Phase 4

a) Build AI model to identify different fashion features from an image.
b) Build crawler for fashion bloggers to extract pieces that go together (using fashion features).
c) Test GUI frontend and backend.

Phase 5

a) Build AI model to recommend matching pieces.
b) Test AI model from (4a).

Phase 6

a) Test AI model from (5a)
b) Integration testing after integrating tested AI model from (4a)

Phase 7

a) Integration testing after integrating tested AI model from (5a)

As shown above, the first did not contribute to the actual implementation of our project and that all of the work done on the implementation has been carried out in the last 4 months.

2.5.4.2 Future Time Plan

	Activity
July	Refining the modules and adding functionalities while negotiating deals with online stores and other fashion experts to make us represent them.
August	Improving UI and UX.

September	Alpha testing followed by beta testing.
October	Deployment

In creating our future plan, we take into consideration the speed of emergence of competitors in the market. Fortunately, there are not many competitors as shown in the market survey above and the competitors do not offer nearly as much functionalities, which would appeal to the public as our application does. The only competitors that could match or surpass our application's functionalities and specs are not based in Egypt. Furthermore, if a competitor from those who are based abroad decides to target Egypt, our time plan allows us to deploy before any competitor could gain momentum in the Egyptian market.

Moreover, the above schedule adheres to our team mates' time plan, before any of us decides to take another job.

2.5.5 Availability

This project is available from now and can be used by any person as long as he's having a smart phone. All what he needs is an internet connection and to have the application downloaded on his/her mobile. It's available at anytime of the day and he can use it at any time he wants.

All the mentioned options are now available in the application. What we'll do is to add new stores and websites that are found in Egypt, and add new fashion bloggers to our application.

2.5.5.1 Marketing Strategy

According to Porter's Competitive Advantage model, we're following a "Differentiation" marketing strategy. Differentiation involves making products or services different from and more attractive than those of our competitors. How we do this depends on the exact nature of our industry and of the products and services themselves, but will typically involve features, functionality, durability, support, and also brand image that our customers value.

To make a success of a Differentiation strategy on our project, we focused on:

- Good research, development and innovation.
- The ability to deliver high-quality products or services.
- Effective sales and marketing, so that the market understands the benefits offered by the differentiated offerings.

We are focusing on delivering the best quality with the best accuracy for our customers, and having unique features that are not found in other applications.

2.5.6 Legal Feasibility

- Our system doesn't conflict with legal requirements in any of the project stages.
- We've sent emails to Universities for permission for using some of the datasets used in our project.
- When we needed a GPU, we used "Google Colab" which is a free cloud service that supports free GPU to be used by any person, with a condition to use only for 12 hours per day.

3. Necessary Background

3 Necessary Background

3.1 Introduction

In this chapter, we shall go over any background information, research and technical details necessary for reading this report.

Literature review shall go over basic concepts and research progressions over the years, in each of those concepts, to show the inspiration for our specific design of the application and for the methods used to implement them. Afterwards we will cover technical details about topics specifically used in our project.

3.2 Literature Review

3.2.1 Introduction

In this section, we will introduce a few topics without going into technical details. This is to provide background about the topic such as previous work or research done on the topic, and any interesting history about it.

3.2.2 Fashion Specific Work

Much research has been done concerning the subject of fashion. This subject covers a large variety of research topics including clothing parsing [1. 2, 3] which focuses on segmenting clothing items in images. Much of the work done in clothing parsing is constructed upon pose estimation while some focus on joint segmentation and labeling.

Clothing retrieval [4, 5] is another approach to the fashion research topic, which aims to retrieve the same fashion piece in different images under different conditions. Each of the mentioned research methods are the base of many models where the data used for training are either collected from human annotated images, or images from crawled websites utilizing the meta-data tags used in the websites.

We use a combination of these methods in our project to achieve all the required functionalities; these methods and approach of implementation are further discussed in the remaining sections of this chapter.

3.2.3 Image Processing

It is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which the input is image, and the output may be image or characteristics associated with that image. Image processing includes the following three steps.

1. Importing the image with optical scanner or by digital photography.
2. Analyzing and manipulating the image, which includes data compression and image enhancement, and spotting patterns that are not to human eyes.
3. The output is the last stage in which the result can be an altered image or some characteristics or reports that is based on image analysis.

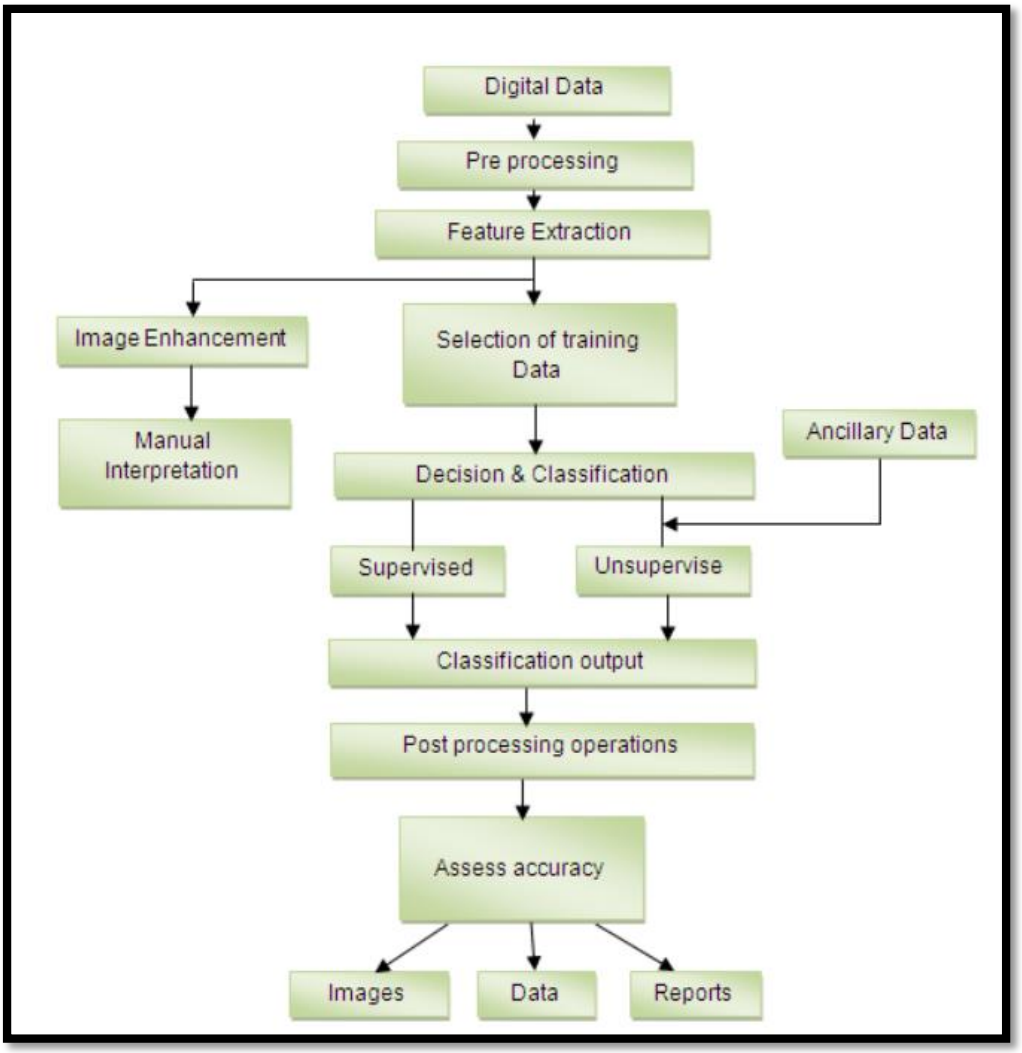


Figure 3-1 Image Processing Phases

Figure (3-1) shows the different phases of image processing.

3.2.4 Big Data Mining and Analytics

"Big Data" is data whose scale, distribution, diversity and timeliness require the use of new technical architectures and analytics to enable insights that unlock new sources of business value.

"Big Data Mining" is a subfield of computer science and statistics with an overall goal to extract information from a data set and transform the information into a comprehensible structure for further use.

“Big Data Analytics” is a form of advanced analytics, which involves complex applications with elements such as predictive models, statistical algorithms and what-if analysis powered by high-performance analytics systems

“Big Data Mining and Analytics” is the complex process of examining large and varied data sets -- or big data -- to uncover information including hidden patterns, unknown correlations, market trends and customer preferences that can help organizations make informed business decisions.

3.2.5 Web Crawling

A Web crawler, sometimes called a ‘Spider’ or ‘Spiderbot’ and often shortened to crawler, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing (web spidering). Crawling can be used in:

1. Data Extraction
2. A general-purpose web crawler (Used by Search Engines)

A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the pages and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites, it copies and saves the information as it goes. If the crawler is used for extracting certain data that the user needs, it must be customized by coding to extract only the needed information. Moreover, if its’s used as a general-purpose web crawler, it saves all the information. The archives are usually stored in such a way they can be viewed, read and navigated as they were on the live web, but are preserved as ‘snapshots'. Figure (3-2) shows the complete cycle of web crawling.

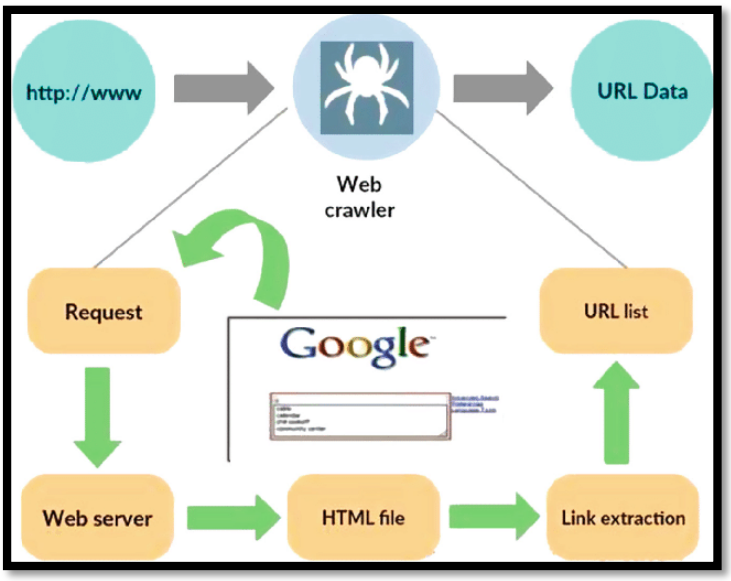


Figure 3-2 Web Crawling Cycle

3.2.6 NoSQL

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in documents instead of the tabular relations used in relational databases. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages, or sit alongside SQL database in a polyglot persistence architecture.

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), finer control over availability and limiting the Object-relational impedance mismatch. The data structures used by NoSQL databases are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability, partition tolerance, and speed. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages (instead of SQL, for instance the lack of ability to perform ad-hoc joins across tables), lack of standardized interfaces, and huge previous

investments in existing relational databases.]Most NoSQL stores lack true ACID transactions, although a few databases have made them central to their designs.

Instead, most NoSQL databases offer a concept of "eventual consistency" in which database changes are propagated to all nodes "eventually" (typically within milliseconds) so queries for data might not return updated data immediately or might result in reading data that is not accurate, a problem known as stale reads. Additionally, some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.

3.2.7 Full Stack Development

The area of knowledge in which a developer know how to integrate the front-end of the application with its back-end and its database. He knows how to handle different type of requests, how to extract the information from these requests and how to transform data to different forms in order to be transferred between applications that are not on the same machine and are not written in the same programming language.

There is a portion of the application the user sees which is the “front-end”, and the largest part of the application remains unseen, this part is the “back-end.”

Back-end development is concerned with deciding how to organize the logic of the whole system so that it can be maintained and run properly. It refers to the server side of development where you are primarily focused on how the system works On the other hand, front-end development manages everything that users visually see first in their application; it is the gate of interaction between the user and the system. Front-end developers are responsible for the look and feel of a site.

3.3 Technical Details related to the project

In this section, we shall review all the basic building blocks required for the main functionalities, covering their technical details.

3.3.1 Fully Convolutional Networks

Fully Convolutional Networks [6] has become a widely used method for segmentation for its high performance and accuracy. It is based on the concept of convolutional networks but in traditional convolutional networks for semantic segmentation, there has always been tension between semantics (the what) and location (the where). This inherent tension has been addressed in FCNs with its local-to-global pyramid and skip architecture by taking advantage of the feature spectrum by combining deep, coarse, semantic information and shallow, fine, and appearance information in the skip architecture.

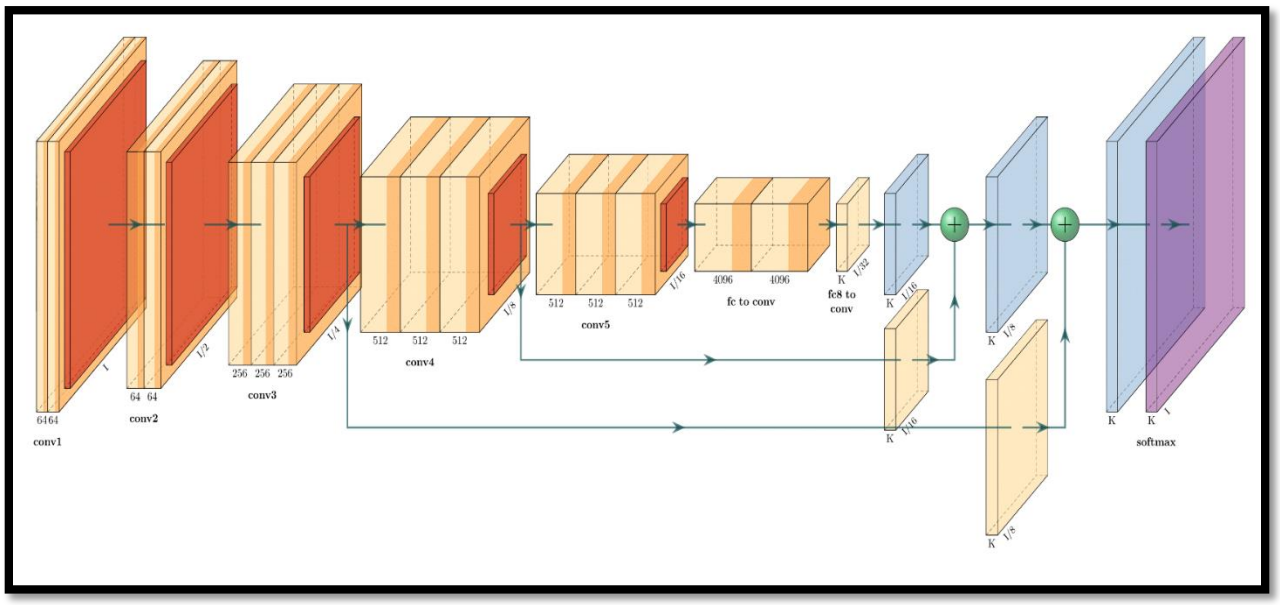


Figure 3-3 FCN pyramid architecture

Down-sampling Path:

The FCN adapts a pyramid architecture as shown in figure (3-3). The first half of the architecture is composed of 5 blocks each of which is composed of a number of convolutional layers followed by a max pool layer. All the convolutional layers in a single block share the same depth. As we go deeper in the first half of the network, the depth of the convolutional layers in the blocks increases and the size of the feature maps decreases due to the max pool layer. Each max pool layer down samples the feature maps by half, so both length and width of

image/feature map, is halved. After the 5 blocks, the original image's width and height would be divided by 32 (halved 5 times), therefore the image needs to have a width and a height that are divisible by 32.

Up-sampling Path:

In the second half of the networks architecture we would like to up sample, make it bigger, the feature maps after so as to map the semantic information captured by convolutions back to its spatial location information.

In a sense, up sampling with factor f is convolution with a fractional input stride of 1/f. So long as f is integral, a natural way to up sample is therefore backwards convolution (sometimes-called deconvolution) with an output stride off. Such an operation is trivial to implement, since it simply reverses the forward and backward passes of convolution. Thus up sampling is performed in-network for end-to-end learning by backpropagation from the pixel-wise loss.

Note that the deconvolution filter in such a layer need not be fixed (e.g., to bilinear up sampling), but can be learned. A stack of deconvolution layers and activation functions can even learn a nonlinear up sampling.

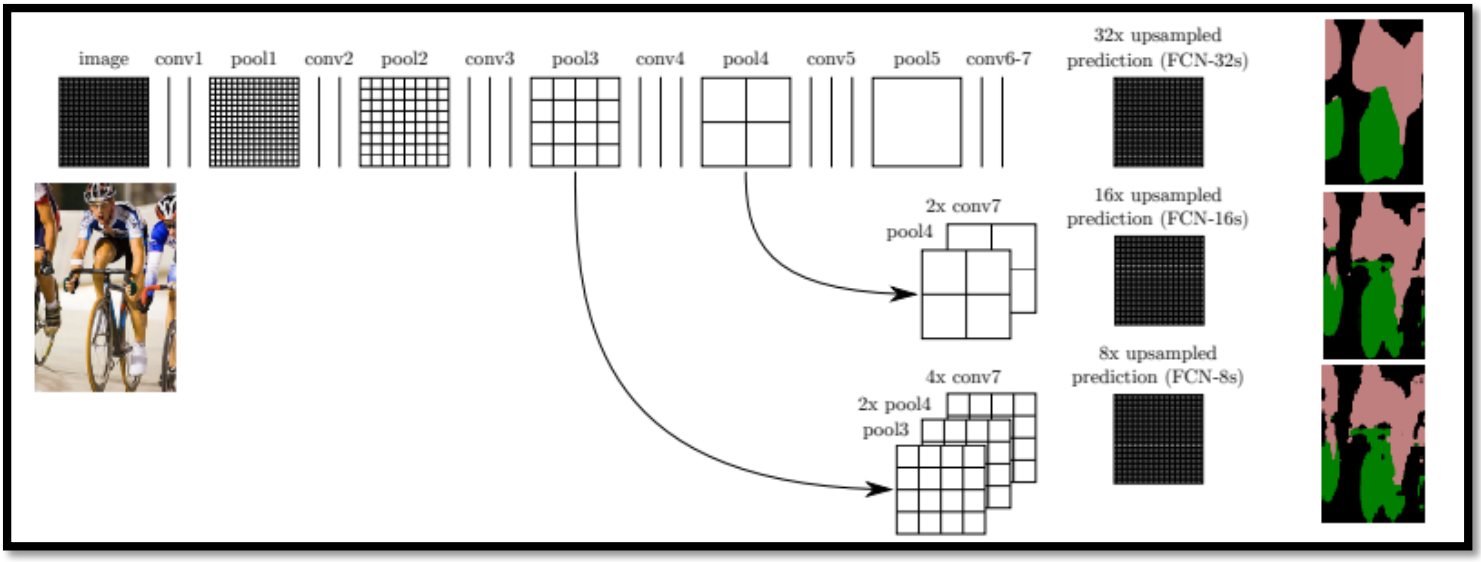


Figure 3-4 FCN skip connection illustration

Skip Connections:

As we go deeper in the network, deep features and semantic information are captured while spatial location information is lost.

This is addressed by adding skips that combine the final prediction layer with lower layers with finer strides. This turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones (Figure 3-3). Combining fine layers and coarse layers lets the model make local predictions that respect global structure.

There are 3 variants of the FCN as shown in figure (3-4). The fcn-32 variant does not implement the skip connections which makes the prediction coarse and without many details about spatial information of the predicted label. The fcn-16 uses skip connection on one location only from the 5 blocks (from pool layer of block 4 specifically), while fcn-8 uses skip connection on 2 points so as to take advantage of every part of the feature spectrum. The network as a whole can be seen better in figure (3-3) while the difference between the 3 variants regarding the results can be comprehended better from figure (3-4).

3.3.2 VGG Net with K Categories

As we wanted features to be robust to background changes and to focus entirely on the outfit. Further, features should be meaningful to fashion attributes such as styles. Thus, we exploit the composition of outfits, so that outfits are represented by observing how likely the possible constituents are.

Convolutional neural networks (CNNs) are renowned for their high recognition performance. In particular, we used 3×3 kernels for the convolutional filters to keep the number of weights down for the network and allow increasing the number of layers. A preliminary analysis showed that dropout in the convolutional layers was not beneficial, and thus dropout is used only in the fully-connected layer to prevent overfitting throughout the architecture. The network output is given as a vector of probabilities associated with k clothing items (i.e., blazer, shirt, skirt, dress etc.), that is a compositional feature vector.

We used the same architecture as in VGG16 Network but we added a fully connected layer of K classes. Moreover, we can't use Softmax layer as the problem is multi label and we want to give each clothing item the probability that it exists, which is independent on other clothing items.

As known that Softmax layer normalizes the values in a vector from 0 to 1 and the sum of all elements of the vector is 1. This is used in single label problems as we need to vote for all classes. Then, we choose the highest probable class.

Softmax function is given by:

$$\sigma(zi) = \frac{e^{zi}}{\sum_{j=1}^{k} e^{zj}} \text{ , for i =1,….,k}$$

In Multi-Label problems, we need the probabilities to be independent so, we use **Sigmoid** layer as it works independently and achieves the desired model.

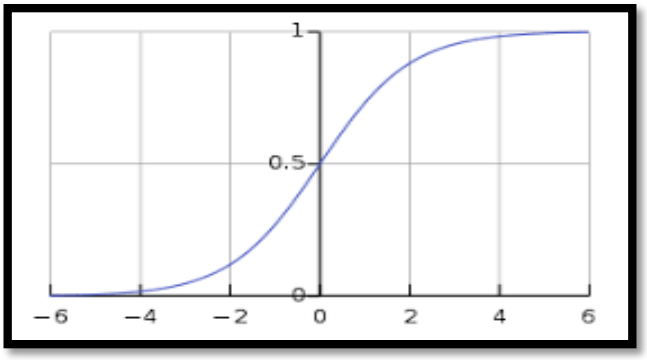


Figure 3-5 Sigmoid

Table 3-1 Table 1 Detailed Architecture of VGG Multi-label Problem

Type	Kernel size	Output Size
Convolution	3 x 3	224 x 224 x 64
Convolution	3 x 3	224 x 224 x 64
Max Pooling	3 x 3	112 x 112 x 64
Convolution	3 x 3	112 x 112 x 128
Convolution	3 x 3	112 x 112 x 128
Max Pooling	3 x 3	56 x 56 x 128
Convolution	3 x 3	56 x 56 x 256
Convolution	3 x 3	56 x 56 x 256
Convolution	3 x 3	56 x 56 x 256
Max Pooling	3 x 3	28 x 28 x 256
Convolution	3 x 3	28 x 28 x 512
Convolution	3 x 3	28 x 28 x 512
Convolution	3 x 3	28 x 28 x 512
Max Pooling	3 x 3	14 x 14 x 512
Convolution	3 x 3	14 x 14 x 512
Convolution	3 x 3	14 x 14 x 512
Convolution	3 x 3	14 x 14 x 512
Max Pooling	3 x 3	7 x 7 x 512
Full-Connected Dropout (50%)		4096
Full-Connected Dropout (50%)		4096
Sigmoid		K

3.3.3 Composite Network for Semantic Composition

We follow the approach of using 3x3 kernels for the convolutional filters to keep the number of weights down for the network and allow increasing the number of layers. One-pixel padding is used to keep the input size and output size of the convolutional layers constant. In order to allow efficient learning the entire network from scratch, we rely on Batch Normalization layers and Dropout is used to prevent overfitting where it drops weights of the neurons to zero during training to make the training loss decrease with the validation loss.

A full overview of the architecture can be seen in figure(3-6) below. The architecture is simpler in training as it has less parameters than the VGG. Moreover, we used VGG as it gives more accurate results.

The output of the network after convolution layers is K probabilities, each representing the probability of the presence of the clothing item. We used also **Sigmoid** as the activation function in the last layer because the problem is Multi-Label Classification problem. More than one clothing item may exist in one image.

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 384, 256, 64)	1792
block1_conv2 (Conv2D)	(None, 384, 256, 64)	36928
block1_dropout (Dropout)	(None, 384, 256, 64)	0
block1_pool (MaxPooling2D)	(None, 96, 64, 64)	0
block1_batch_normalization (	(None, 96, 64, 64)	256
block2_conv1 (Conv2D)	(None, 96, 64, 128)	73856
block2_conv2 (Conv2D)	(None, 96, 64, 128)	147584
block2_dropout (Dropout)	(None, 96, 64, 128)	0
block2_pool (MaxPooling2D)	(None, 24, 16, 128)	0
block2_batch_normalization (	(None, 24, 16, 128)	512
block3_conv1 (Conv2D)	(None, 24, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 24, 16, 256)	590080
block3_dropout (Dropout)	(None, 24, 16, 256)	0
block3_pool (MaxPooling2D)	(None, 6, 4, 256)	0
block3_batch_normalization (	(None, 6, 4, 256)	1024
convlayer (Conv2D)	(None, 6, 4, 128)	295040
flatten_5 (Flatten)	(None, 3072)	0
fc3 (Dense)	(None, 23)	70679

Total params: 1,512,919
Trainable params: 1,512,023
Non-trainable params: 896

Figure 3-6 Comp Network Architecture

3.3.4 Image Processing

A field of image processing that we used in our project are the color models. Many different color models can be applied to a certain image and extract the color feature from it. These color models include the following:

3.3.4.1 RGB Module

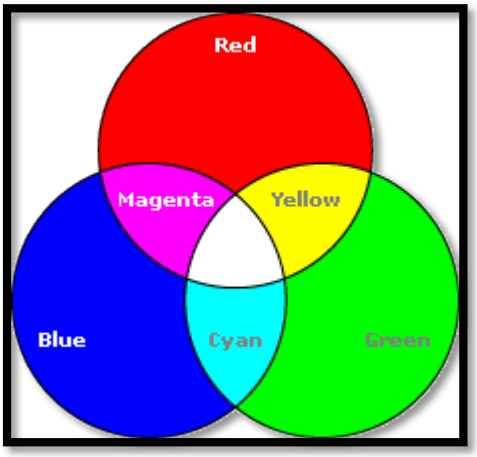


Figure 3-7 - RGB Color Model

The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

The main purpose of the RGB color model is for the sensing, representation and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography.

RGB is a device-dependent color model, where different devices detect or reproduce a given RGB value differently, since the color elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer, or even in the same device over time. Thus, an RGB value does not define the same color across devices without some kind of color management. However, each color has a range of values.

Typical RGB input devices are color TV and video cameras, image scanners, and digital cameras. Typical RGB output devices are TV sets of various technologies (CRT, LCD, plasma, OLED, quantum dots, etc.), computer and mobile phone displays, video projectors, multicolor LED displays and large screens such as JumboTron. Color printers, on the other hand are not RGB devices, but subtractive color devices (typically CMYK color model).

In order to choose a color by using the RGB model, you should change R, G and B values that identifies the color you need. These values, which have integer values from zero to 255.

3.3.4.2 HSV Module

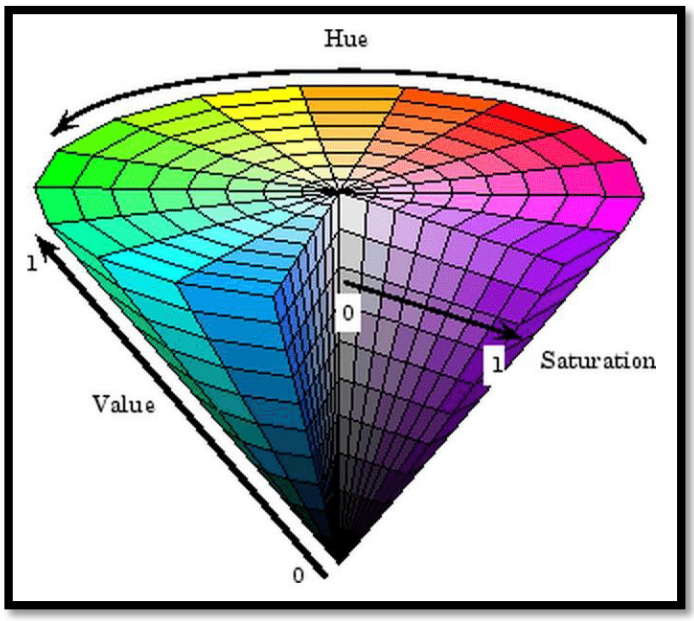


Figure 3-8 - HSV Color Model

HSV is defined in a way that is similar to how humans perceive color. It is based on three values: hue, saturation, and value. This color space describes colors (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness value. Some color pickers (like the one in Adobe Photoshop) use the acronym HSB, which substitutes the term brightness for value, but HSV and HSB refer to the same color model.

The HSV color wheel is sometimes depicted as a cone or cylinder, but always with the three HSV components.

Hue (The color portion of the color model) is expressed as a number from zero to 360 degrees:

Table 3-2 Hue mapping

Color	Angle
Red	0–60
Yellow	60-–20
Green	120–180
Cyan	180–240
Blue	240–300
Magenta	300–360

Saturation (the amount of gray in the color) from 0 to 100 percent. Reducing the saturation toward zero to introduce more gray produces a faded effect. Sometimes, saturation is expressed in a range from just 0-1, where zero is gray and 1 is a primary color.

Value (works in conjunction with saturation and describes the brightness or intensity of the color) from 0-100 percent, where zero is completely black, and 100 is the brightest and reveals the most color.

3.3.4.3 HSL Module

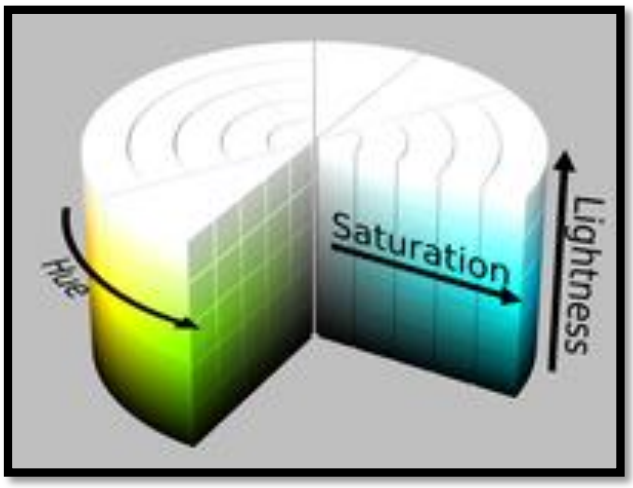


Figure 3-9 - HSL Color Model

HSL is having the same hue and saturation of the HSV but with its different orientation of colors in the model, and differs in the last component, which is the lightness here. The ‘Lightness’ of a color is a gradient between black and white, where the ‘bottom’ of the cylinder is a total black, and the ‘top’ of the cylinder is totally white.

This model is also used the same way as we use the HSV in terms of getting the values of the three components. Where the hue is a degree value from zero to 360, and the saturation and lightness are a percentage value from zero to 100%.

3.3.5 Difference between HSV and HSL

The difference appears mathematically in the calculations of the colors’ values, where each color is found in a different position in the two models. Moreover, the following figure(3-10) will show the difference between the HSV and HSL color models.

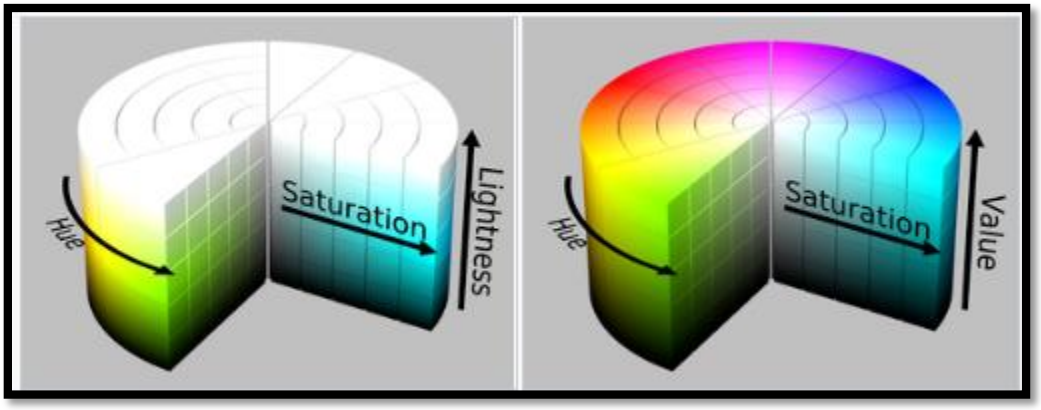


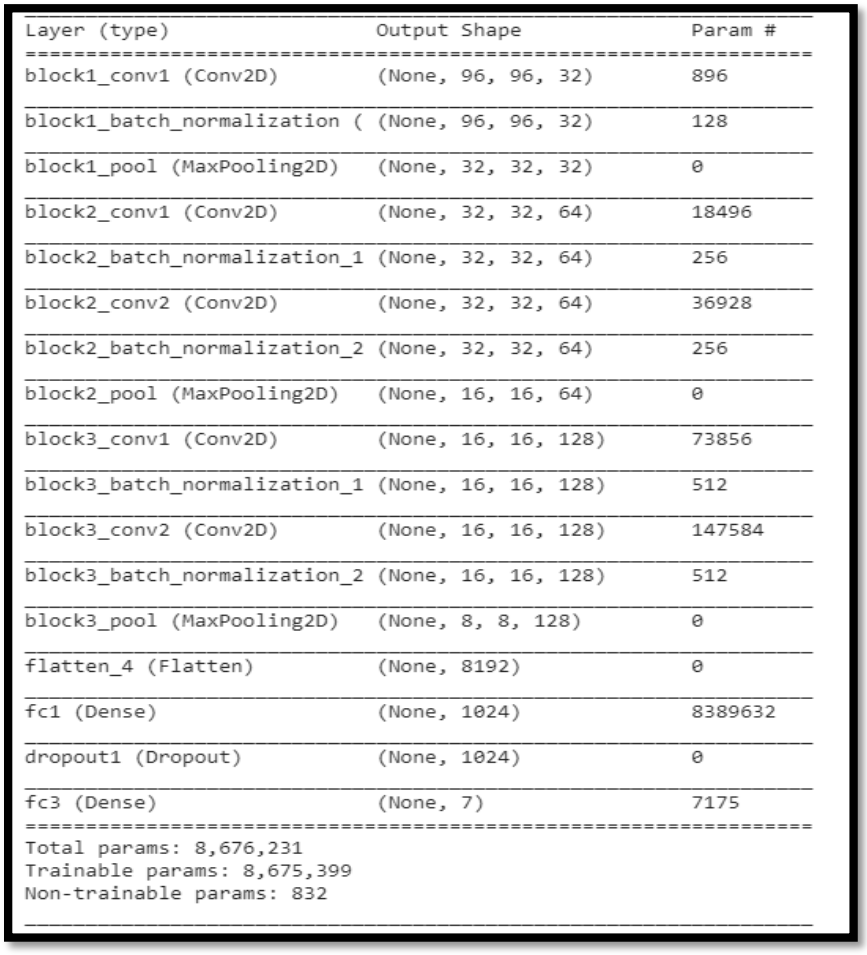
Figure 3-10 - HSL vs HSV

We used image processing in order to build our Color Identification Module. We used different color models in order to extract the color of every clothes’ piece and to reach the results we needed in this module.

3.3.6 Smaller Version of VGG Model

In case of classifying a single clothing item into one category of P categories, the problem will be Multi-Class Single-Label classification problem. Moreover, we need to use a smaller convolutional network because the problem is simpler and we need it as fast as possible.

We used a smaller version of VGG model that works specifically on single label classification. We added Dropout layers that drop the weights during training to zeros to avoid overfitting. Moreover, we used Softmax activation function because we need the highest probability that the piece belongs to a certain class and after adding the P probabilities, the sum is 1 so the probabilities are dependent.



Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 96, 96, 32)	896
block1_batch_normalization (	(None, 96, 96, 32)	128
block1_pool (MaxPooling2D)	(None, 32, 32, 32)	0
block2_conv1 (Conv2D)	(None, 32, 32, 64)	18496
block2_batch_normalization_1	(None, 32, 32, 64)	256
block2_conv2 (Conv2D)	(None, 32, 32, 64)	36928
block2_batch_normalization_2	(None, 32, 32, 64)	256
block2_pool (MaxPooling2D)	(None, 16, 16, 64)	0
block3_conv1 (Conv2D)	(None, 16, 16, 128)	73856
block3_batch_normalization_1	(None, 16, 16, 128)	512
block3_conv2 (Conv2D)	(None, 16, 16, 128)	147584
block3_batch_normalization_2	(None, 16, 16, 128)	512
block3_pool (MaxPooling2D)	(None, 8, 8, 128)	0
flatten_4 (Flatten)	(None, 8192)	0
fc1 (Dense)	(None, 1024)	8389632
dropout1 (Dropout)	(None, 1024)	0
fc3 (Dense)	(None, 7)	7175

Total params: 8,676,231
Trainable params: 8,675,399
Non-trainable params: 832

Figure 3-11 - Detailed Architecture of Pieces Identification Model

3.3.7 Big Data Mining and Analytics

3.3.7.1 Data Structures

The data that is to be analyzed and processed by big data mining and analytics tools can be in many forms and structures.

- **Structured data:**

  Data that is containing a defined data type, format, structure.

  E*xample:* transactions data

- **Semi-Structured data:**

  Textual data files with a discernable pattern, enabling parsing.

  Example: XML data files

- **Unstructured data:**

  Data that has no inherent structure and is usually stored as different types of files.

  Example: text documents, PDFs, images and video

In our project, the data is images crawled from the internet; different modules in the system then process each of these images in order to produce CSV files that correspond to the whole dataset. These CSV files are then passed to the data mining and analytics tool to obtain the module output and provide us with the needed insights.

3.3.7.2 Association Rules Mining

Association rules mining is a procedure, which is meant to find frequent patterns, correlations, associations, or causal structures from data sets found in various kinds of databases such as relational databases, transactional databases, and other forms of data repositories. Given a set of transactions, association rule mining aims to find the rules, which enable us to predict the

occurrence of a specific item based on the occurrences of the other items in the transaction. The rules obtained from this procedure are in the form of (Left Right) which is read as 'Left implies Right'. This is comprehended as if the Left is found, the right will also be found. Two algorithms are used for generating these rules:

1. "Apriori" algorithm
2. "Eclat" algorithm
3. "FP-Growth" algorithm

The three algorithms result in the same matching rules. They only differ in the runtime. Apriori algorithm searches for the rules in a breadth-first algorithm while Eclat and FP-Growth search using depth-first algorithm. For large transaction size or for large data sets it is better to use Eclat or FP-growth instead of Apriori algorithm; this is because the run time of Apriori increases rapidly in these cases. Apriori has serious scalability issues and exhausts available memory much faster than Eclat and FP-Growth algorithms, which have nearly the same runtime. Because of this, Apriori should not be used for large datasets. Most frequent item set applications should consider using either FP-Growth or Eclat. Figure (3-12) shows the runtime of the three algorithms when increasing the transaction size and for increasing the size of the dataset.

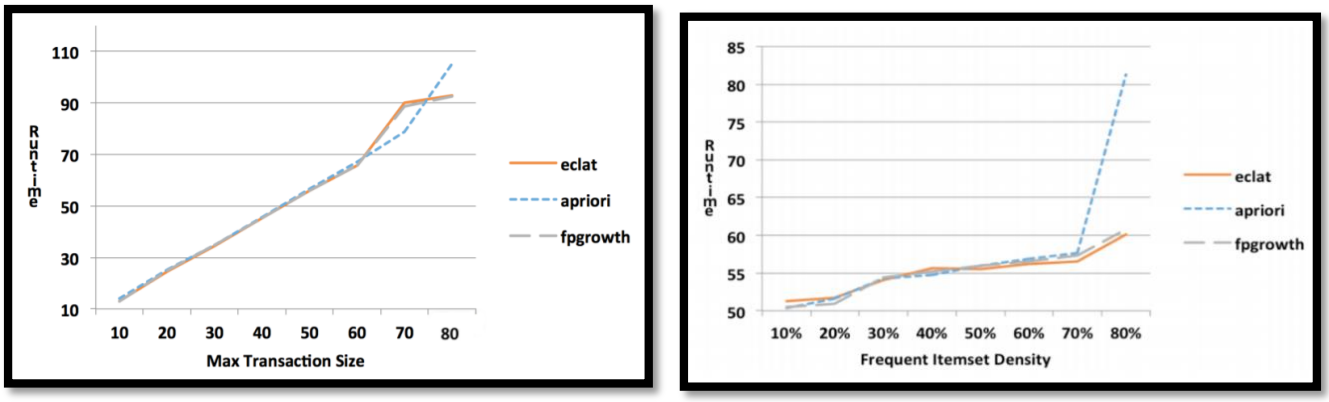


Figure 3-12 - Comparison between Apriori, Eclat and FP-Growth Algorithms

3.3.8 NoSQL and MongoDB

As the data we have in our project is unstructured data, we used NoSQL to build our database and we chose MongoDB as our database program. Where we used it to save our data in different

collections, and each collection contains many JSON documents that holds the information of the item. These data stored in the documents are then used in the other project modules, which are the similar outfits' module, trending and matching module and color identification module. The architecture of MongoDB will be explained in detail in chapter 4.

3.3.9 Full Stack Development

Since our project consists of different modules, where each module is developed based on a certain science and using different languages and tools; there has to be a centralized point of integration between them all to produce a complete system. After integrating, we must decide how to visualize to the user every functionality the system provides and how to let the user interact with the system. These requirements are fulfilled using full stack development where we have the back-end, the middleware and the front-end all together. Figure (3-13) shows the structure of full stack development.

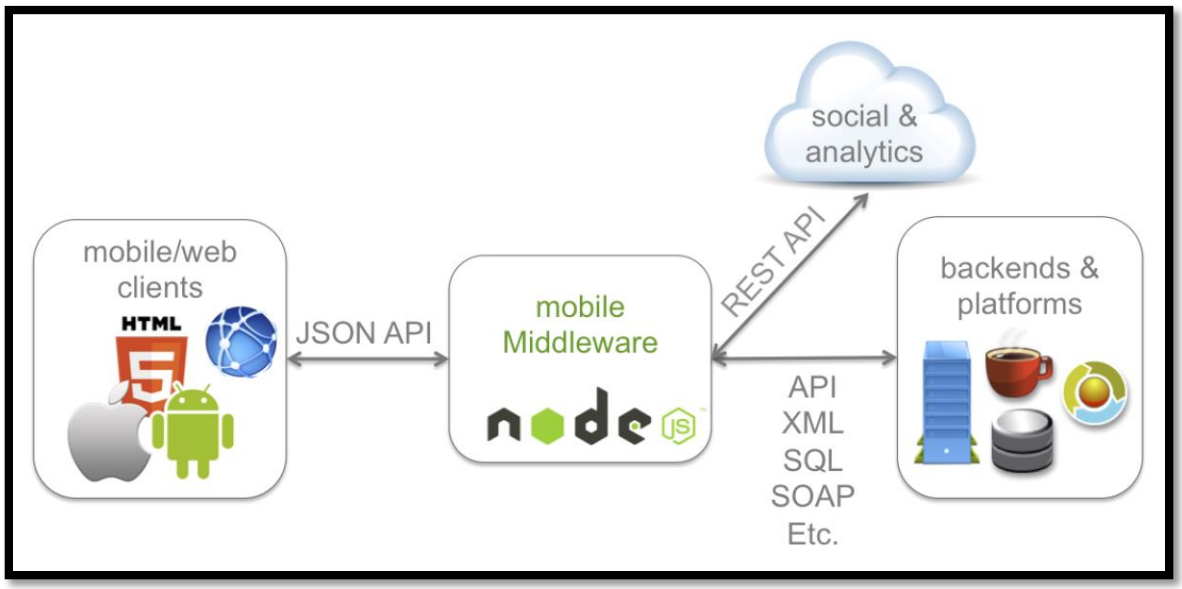


Figure 3-13 - Full Stack Development Structure

1) Front-end development

A front-end is what the users sees on either a web application or a mobile application. For web development, front-end languages include HTML, CSS, and JavaScript. For native iOS applications, front-end languages are Swift and Objective-C while for native Android

applications, XML is used. On the other hand, there are cross-platform applications that work on both Android and iOS; these applications use certain frameworks such as Xamarin, React-Native, and NativeScript. In our project, we decided to make a cross platform mobile application and we used React-Native framework. It is used to develop applications for Android, and iOS by enabling developers to use 'React' along with native platform capabilities

2) **Mobile middleware:**

For mobile applications, there has to be a module that can understand front-end languages and back-end languages so that it can connect them together. This module is developed in Node.js, which has multiple frameworks such as Hapi.js and Express.js. These frameworks provide MVC applications (Model-View-Controller). The middleware accepts HTTP requests from the mobile application, does some processing on these requests and then send the processed data to the corresponding back-end module or API that does the logic and send the output back to the middleware. The middleware then embed the output of the logic in the response and send it back to the mobile application. In our project we used Express.js framework to do this job

3) **Back-end Development:**

The back-end is responsible for the logic of the system. It can be an AI module, a data-mining module, an API or it can be logic that is implemented in the controllers of the MVC application itself. There are many back-end languages depending on the logic that is needed but most importantly, there are Python, Java, C++ and others. In our project we are using Python for AI modules and R for data mining and analytics modules.

4. Design Specifications

4 Design Specifications

4.1 Introduction

This chapter explains the system design of the 'My Shopping Pal' project. . The overall design will be discussed with referral to the block diagram then details of each module including its exact input and output.

4.2 System Design

Our Proposed system is composed of seven main modules as shown in the figure (4-1).

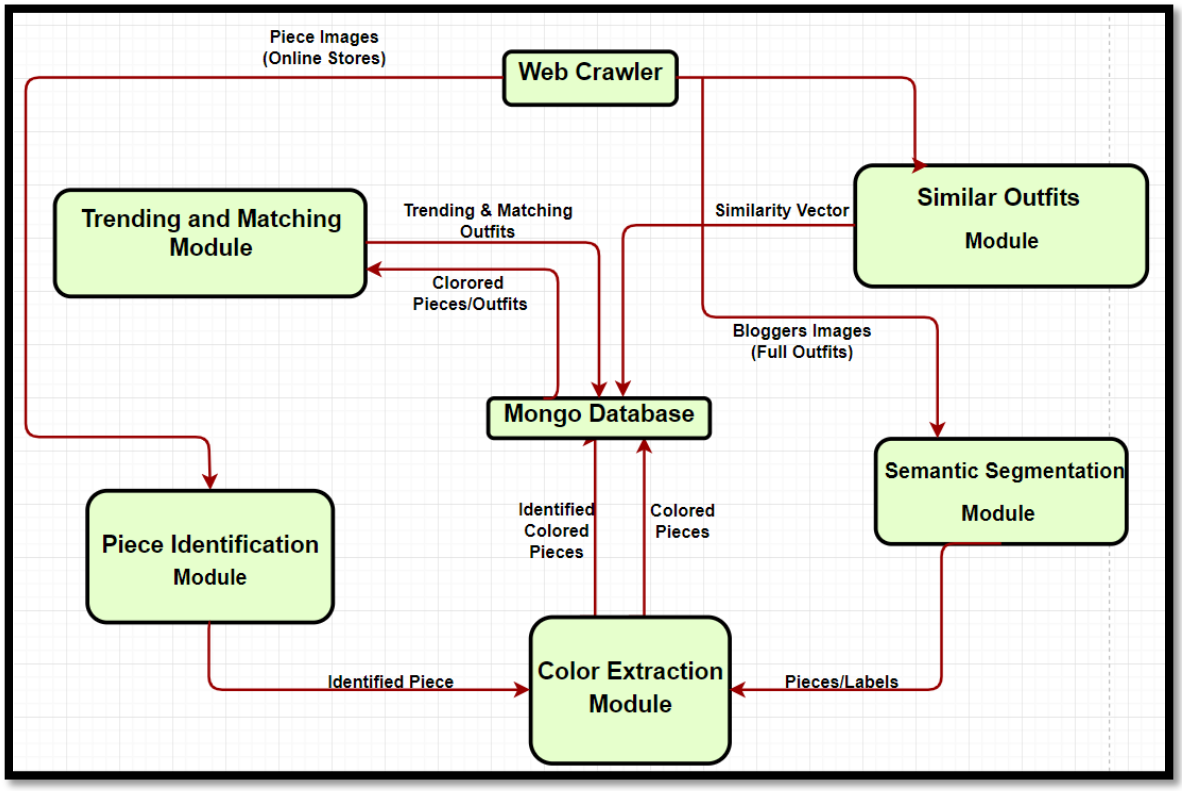


Figure 4-1 System Architecture

4.3 Modules Explanation

4.3.1 Web Crawler

In order to extract the needed information of a product from different websites, we built a crawler for each website as each website is different than the other and each one's front-end is different, so each one is customized in a way that makes us able to extract the information we need from any product on this website. The crawler makes us able to scrap the needed information and save it. Figure (4-2) shows the web crawler cycle

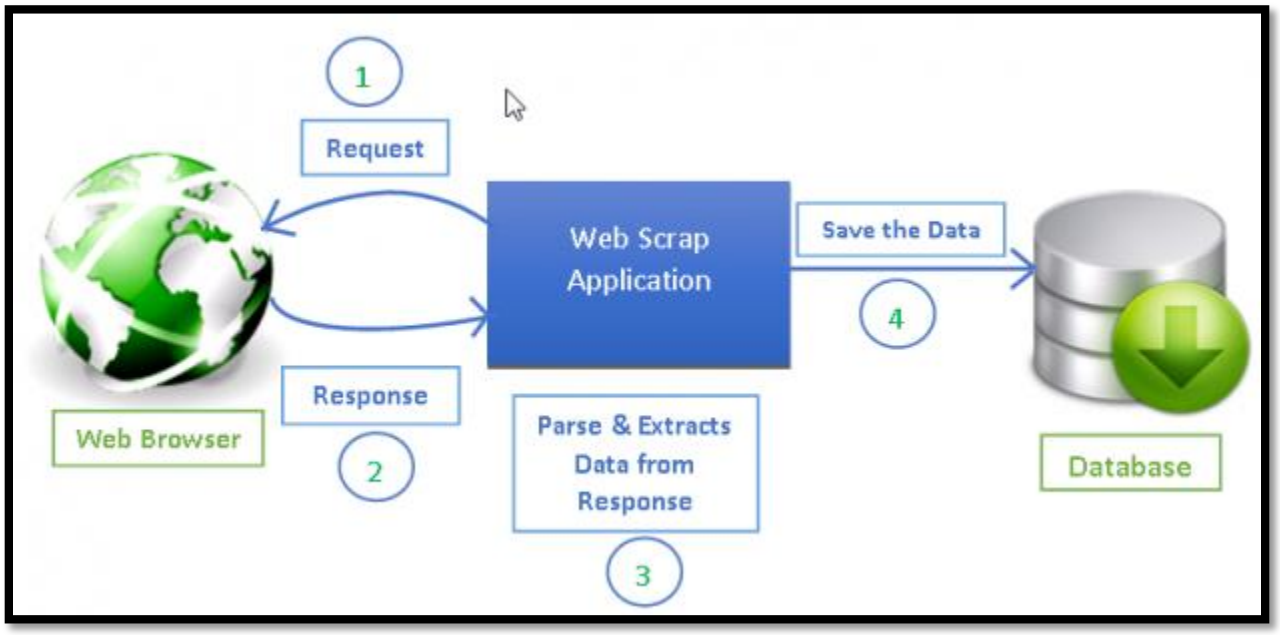


Figure 4-2 - Web Crawler Cycle

Our crawlers take the initial URLs (seeds) to crawl all the links, compare each crawled link with the link pattern of the products in each website by using the regex we entered in the code. If the link is found to have the same link pattern so this means that this link is a product and hence, we start extracting the data from this link by getting the "xpath" of each information we need to extract from the "inspect" option of the link.

Here is an example to clarify the steps of extracting an xpath of a certain information we need, that we then save it in the database:

1) Search for the information you need on the website

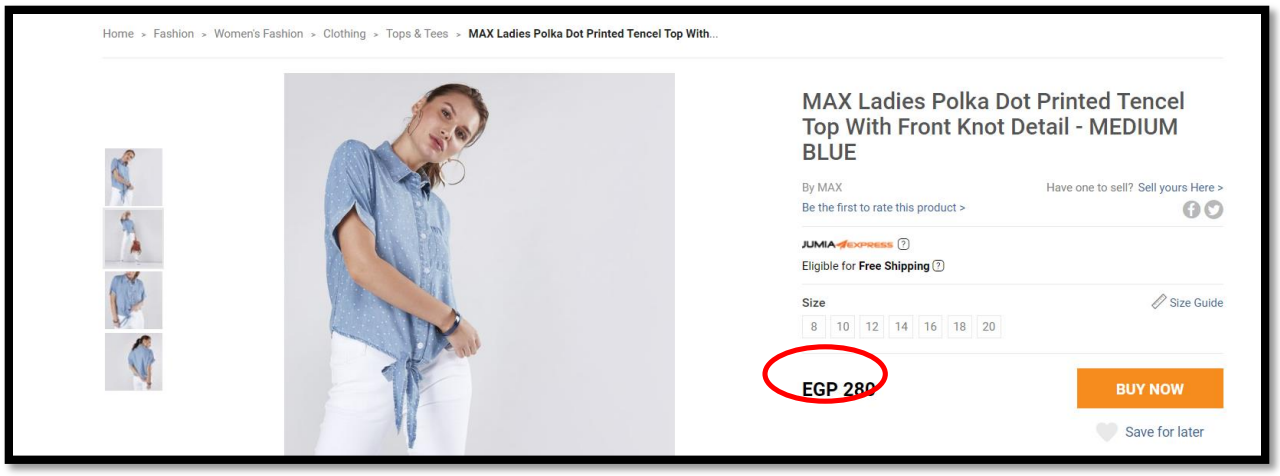


Figure 4-3 - Web Crawler Step 1

2) Right click on the information you found in step 1, then click inspect

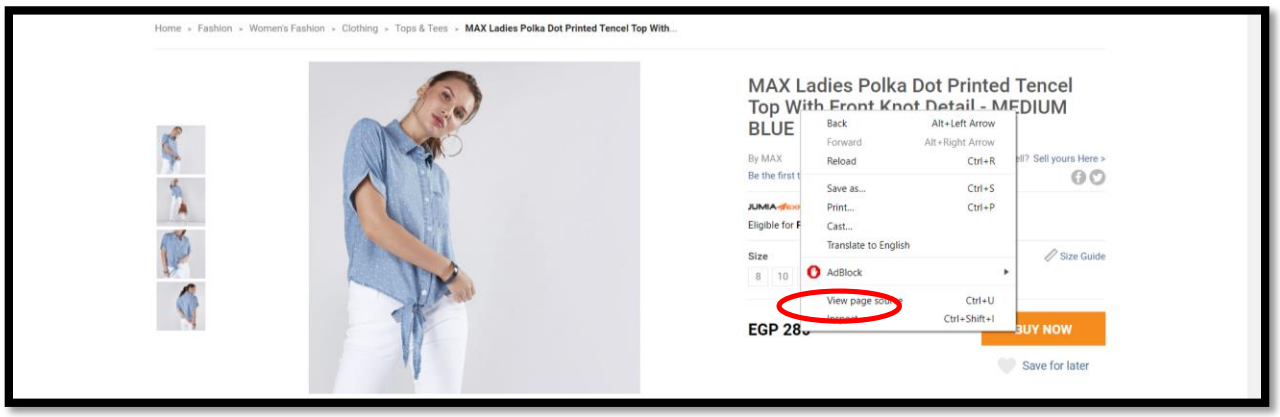


Figure 4-4- Web Crawler Step 2

3) Identify the location of the information in the elements page from the tags wise

Figure 4-5 - Web Crawler Step 3

4) Build the xpath from the information you get from step 3 and try it first in the search box

Figure 4-6 -Web Crawler Step 4

5) Use the xpath built in step 4 in python to extract the information

```
title = response.xpath("/html/body/main/section[1]/div[2]/div[1]/span/h1[contains(@class,'title')]/text()").extract()
price = response.xpath("/html/body/main/section[1]/div[2]/div[1]/div[7]/div[1]/div/span/span[2]/@data-price").extract()
imgSrc = response.xpath("/html/head/meta[contains(@property,'og:image')]/@content").extract()
brandName = response.xpath("/html/body/main/section[1]/div[2]/div[1]/div[contains(@class,'sub-title')]/a/text()").extract()
brandLink = response.xpath("/html/body/main/section[1]/div[2]/div[1]/div[contains(@class,'sub-title')]/a/@href").extract()
mainMaterial = ['None']
color = ['None']
link_extractor = LinkExtractor(allow=JumiaDressesSpider.url_matcher, unique = True)
next_links = [link.url for link in link_extractor.extract_links(response) if not self.is_extracted(link.url)]
```

Figure 4-7 - Web Crawler Step 5

We have built three crawlers for three different websites in order to obtain variety in the results that are returned to the users. These three websites are:

1) Jumia
2) Souq
3) Shein

4.3.2 Mongo Database

In our project, we use NoSQL and MongoDB to save our data in JSON documents. The data in these documents is mainly extracted information about the crawled images from the internet. As it is shown in the system architecture, the crawler passes the images to three different modules in the system where each module processes the image, extracts information, and features from it. These features and information are then saved in the database for every image. Our database architecture is as follows:

The database is divided into different collections where each collection represents a different clothes category (e.g. Dresses, Blouses, Shorts, etc.). In addition, a collection is made that contains the matching rules that are resulted from the big data mining and analytics applied on the fashion bloggers images. Our database collections are shown in figure (4-8)

Figure 4-8 - Database Collections

Each of these collections contains many JSON documents where each document represents an image. A single document in a collection contains all the information extracted for this image, whether crawled or extracted from our modules. Where it contains general information for the piece, which was found on the website (e.g. Image Caption, Price, Season, Pattern, Fabric …etc.). In addition, it contains the Similarity Vector of the image, which is calculated by using the Similar Outfits module. A sample of one of our documents is shown in figure (4-9).

[1] (id="5cdc6b94ab09808bfb899850")	
_id	5cdc6b94ab09808bfb899850
imgSrc	//img.ltwebstatic.com/images2_pi/2019/01/14/15474535383354752679_thumbn..
Category	dress
Color	Multicolor
Composition	100% Polyester
Crawled Link	https://www.shein.com/Surplice-Neck-Self-Belted-Floral-Dress-p-662390-cat-17..
Fabric	Fabric has no stretch
Image Source	//img.ltwebstatic.com/images2_pi/2019/01/14/15474535383354752679_thumbn..
Neckline	Shawl Collar
Pattern Type	Floral
Season	Summer
Style	Boho
Title	Surplice Neck Self Belted Floral Dress
Vector	
color	white

Figure 4-9 JSON Document of an Image

4.3.3 Semantic Segmentation Module

This module is responsible for segmenting any image with a person in it to separate fashion items this person is wearing creating a mask of each piece. We have created a package to perform many functionalities related to this task. In this section we shall explain this package and breakdown its building blocks to explain them in details.

Package Design

- Segmenter
  - Dataset
    - Khurana
    - CFPD
  - Model
    - FCN8
    - Texture
    - Fused Model
  - Testing
    - Visualize
    - Confusion Matrix
  - Utilities

4.3.3.1 Dataset:

The dataset class is responsible for any preprocessing done on the datasets used in our segmenting model. We have used two datasets for this package, Khurana and CFPD, each of which is composed of images of people wearing a variety of fashion items and a pixel annotated image which describes which fashion item each pixel belongs to as shown in figure (4-10)

Figure 4-10

These pixel level annotations maybe be noisy around the edges as it is human labeled. However, these noisy edges do not affect the general performance of our proposed model as the model generalizes over the whole dataset so it returns a fairly neat prediction of that pixel level annotation as shown in figure (4-11).

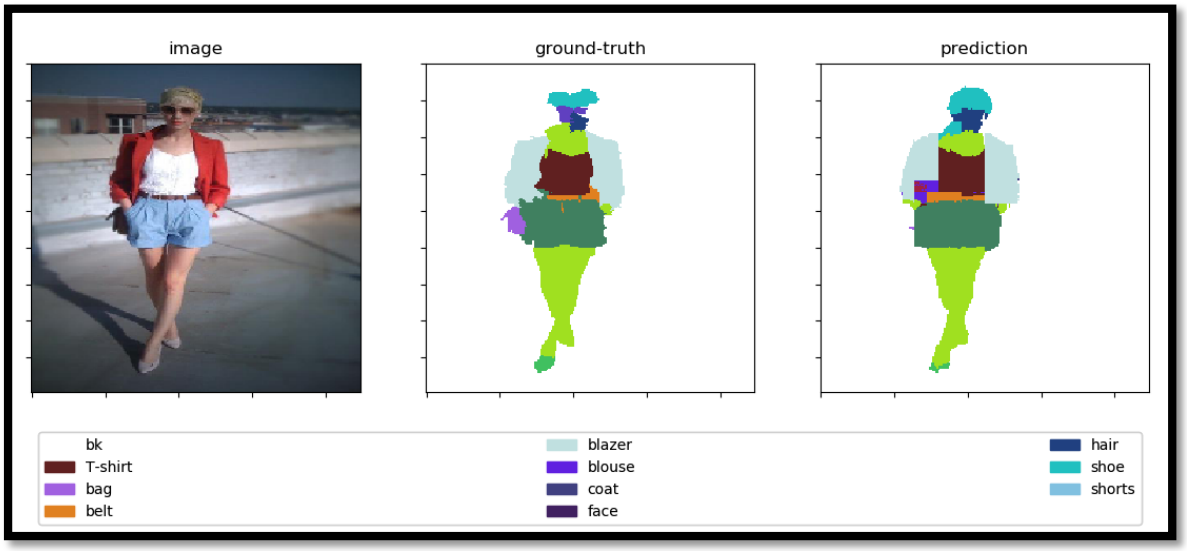


Figure 4-11

4.3.3.2 Model:

This class is responsible for building the segmentation model, training it and using it to make predictions.

The architecture of our proposed model is inspired by that discussed by Khurana et al in their research paper “Exploiting Texture Cues for Clothing Parsing in Fashion Images” [3]. Their proposed design is shown in figure (4-12).

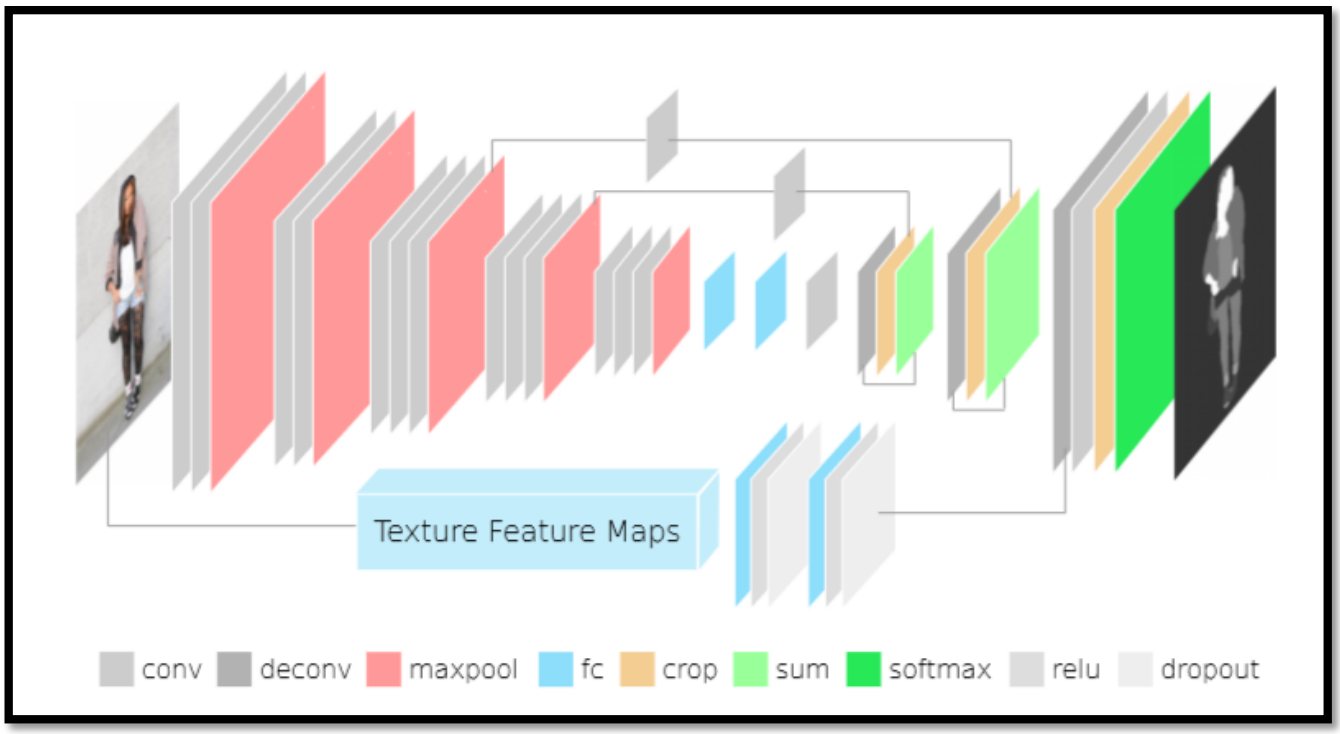


Figure 4-12

As shown in figure (4-12), there are two streams for segmentation. One of which is the FCN8 which we will refer to as the segmentation stream while the other is a texture stream.

Traditionally, texture cues have been used for texture segmentation and texture recognition. However, it has been observed that the texture cues provide fine grained features associated with the material of a particular clothing type, and complement the shape and position information exploited by the contemporary segmentation pipelines. For instance, in the case of sweaters vs tops, it may be hard to distinguish between the two classes on the basis of spatial or shape cues

alone. However, typically the two clothing items not only have different materials, but also contain very different visual patterns on them, which can be exploited for improved segmentation. Thus, in this work, the standard segmentation pipeline has been augmented with a second stream based on the texture-based features.

Our model is inspired by the design above, however, instead of the second stream being composed of the texture cues only, the texture stream is passed as an input to another FCN8 model as shown in figure (4-13).

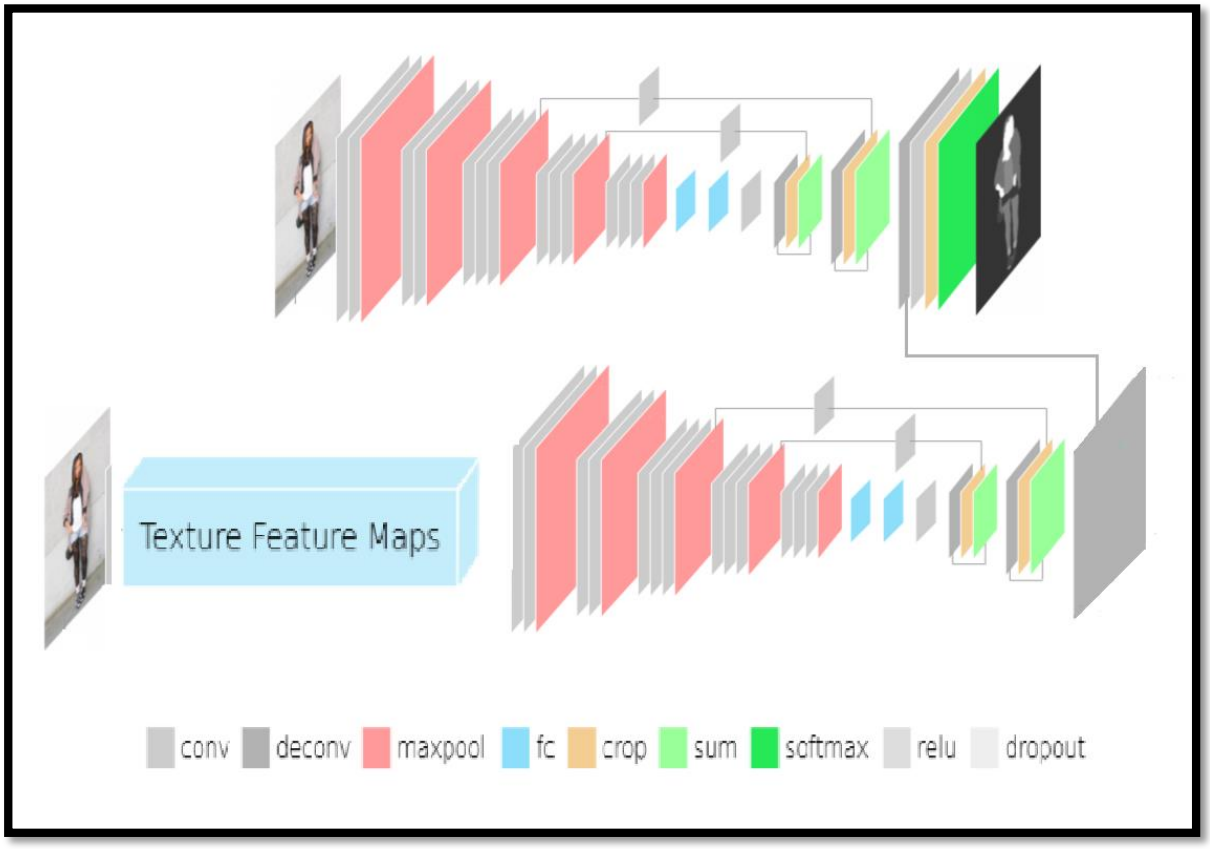


Figure 4-13

Texture Cues:

The two most commonly used texture descriptors are the Gabor feature descriptor [7, 8] and Local Binary Patterns [9]. We found that Gabor feature descriptor performs better than the LBP so we used it as the base for the second stream.

Gabor feature [7, 8] responses are extracted corresponding to different wavelengths, orientations and phases. The following parameters are adopted for feature maps extraction - wavelength: 3-8 pixels, orientation: 0◦, 45◦, 90◦, 135◦ and 5 phase values spaced uniformly from 0 to the wavelength (λ). Only 4 orientation values are chosen since most of the textures are essentially aligned along these angles. We use an 11×11 sliding window for Gabor feature map extraction at each pixel. In Khurana's research paper [3], they have experimented with windows of size 9, 13, 15 but found 11 to be working best so this is the value we have used in our texture stream.

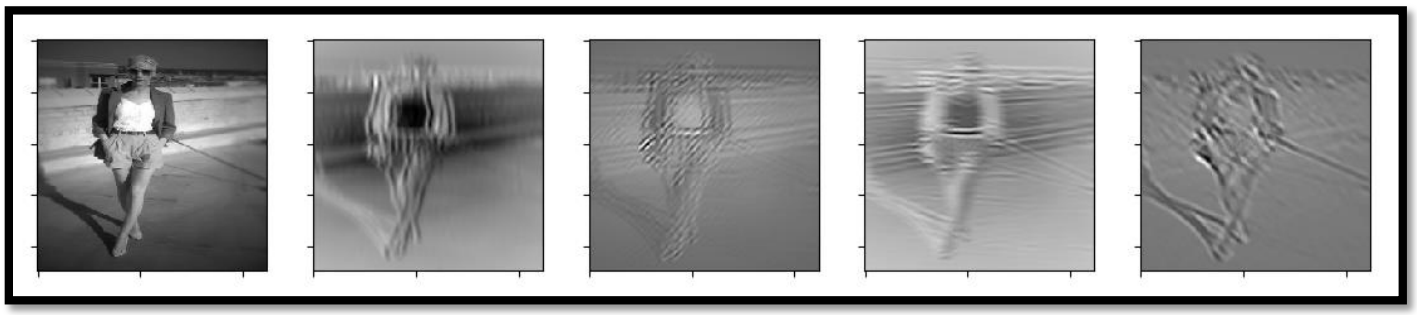


Figure 4-14

Streams Fusion:

The layer before the softmax layer of each stream is concatenated then a convolutional layer of kernel size (1, 1) is applied to fuse the two streams. This method of merging streams has been applied after experimenting with a variety of different methods and kernel windows, which shall be illustrated in chapter 5. After the convolutional layer, a softmax layer is applied to give each pixel a (weight/confidence level) for each (category/ fashion item).

4.3.3.3 Testing:

This class has two main functionalities:

- Visualize the output of any layer in the CNN to help with debugging and understanding the functionality of each layer.
- Compute and plot the confusion matrix for any model implemented in the class Model to help us understand in which categories does the model perform well and

which categories does the model under perform. The output of this functionality is shown in chapter 5 when we discuss the performance of our model.

4.3.3.4 Utilities:

This class performs any and all miscellaneous functionalities needed in our package.

Examples:

- Converting from a normal labeled image, where each pixel takes a value depicting the category it falls under, to OHE and vice versa.
- Showing images and predicted labels with a legend describing under which category does each pixel belong to. This functionality has been used to generate the previous figures.

4.3.4 Similar Outfits Module

A number of fashion retrieval approaches are proposed recently, and they usually operate by retrieving outfits that are similar to a query image in terms of visual elements such as color, shape and texture.

While these approaches are effective in retrieving outfits with similar visual appearance, they do not take into consideration the semantics of the outfits, which is related to abstract concepts such as style, e.g., certain outfits may be appropriate for the same times and places, even if these outfits are not visually similar.

We focus on the clothes items in our model. The model retrieves the Top N similar style to the query image. We consider the problem as Multi-Label Multi Class classification problem. We divided the model into two parts. The first is a Deep Convolutional Neural Network to compute a similarity vector and the second is Retrieving Top N Similar Images. The CNN input is the query image, and the output is a similarity vector giving probabilities to the presence of each clothing item. We used K categories so the vector contains K probabilities; Where K is the number of clothes items in our model (Blouse, Dress, Pants, etc.). The CNN works on image-level, it does not tell the exact pixels of the category. It says only the probability that the item exists in the photo.

After crawling the images of the fashion bloggers, we pass these images to this model and calculate the similarity vector for each image. Then, we save these crawled images with its calculated similarity vector in the database. And whenever a user asks for similar outfits to a certain image, we calculate the similarity vector for the use image and compute the Euclidean distance between the query vector and all the vectors returning minimum Top N photos that are similar in style to the query image.

4.3.5 Single Piece Identification Module

The input to this module is a single piece image like the images shown in figure (4-15). The module classify this single piece into one of P categories then process it to either find pieces similar to it or to match the colored piece with other pieces in our database or in the user's closet. This problem is Single-Label Multi-Class Classification Problem where we want only to classify the input image into one of P clothing items. The photo passes through a CNN to vote for the highest probable class for the entered piece. We used photos crawled from online shops for training the this network so that the query image of the user can either be a piece on its own or a model wearing the piece.

Figure 4-15 Single piece images

4.3.6 The Color Extraction Module

After segmenting the image, we start applying feature extraction on it. Our first feature and the most important one for clothes is the colors features. Either the image is passed to the color model with its categories that are produced from the FCN, or the image is passed from the Single Piece Identification Module with only one label. The color model extracts the color of every piece on its own to produce an array of colored pieces, which represents the image. The color feature extraction is done using Image Processing in Python language using HSV color model, where we defined the values of the Hue, Saturation and Value manually in order to produce 19 main colors: white, beige, silver, gray, black, red, maroon, yellow, gold, orange, olive, lime, green, aqua, teal, blue, navy, pink, purple.

The color model takes the image as an in input. It applies some masks on the different pieces found in the image, or a single mask in case it is a single-piece image. For each pixel in the image, it calculates the Euclidean distance between the pixel color and each color of the 19 pre-defined colors. Then, it takes the minimum distance of the 19 calculated distances to decide on the color of this pixel. We tried using the RGB, HSV and HSL color models to define the 19 different colors and we compared the output of each one of them. Figure (4-16) and figure (4-17) shows the comparison between the different color models applied on a full outfit image and on a single piece image.

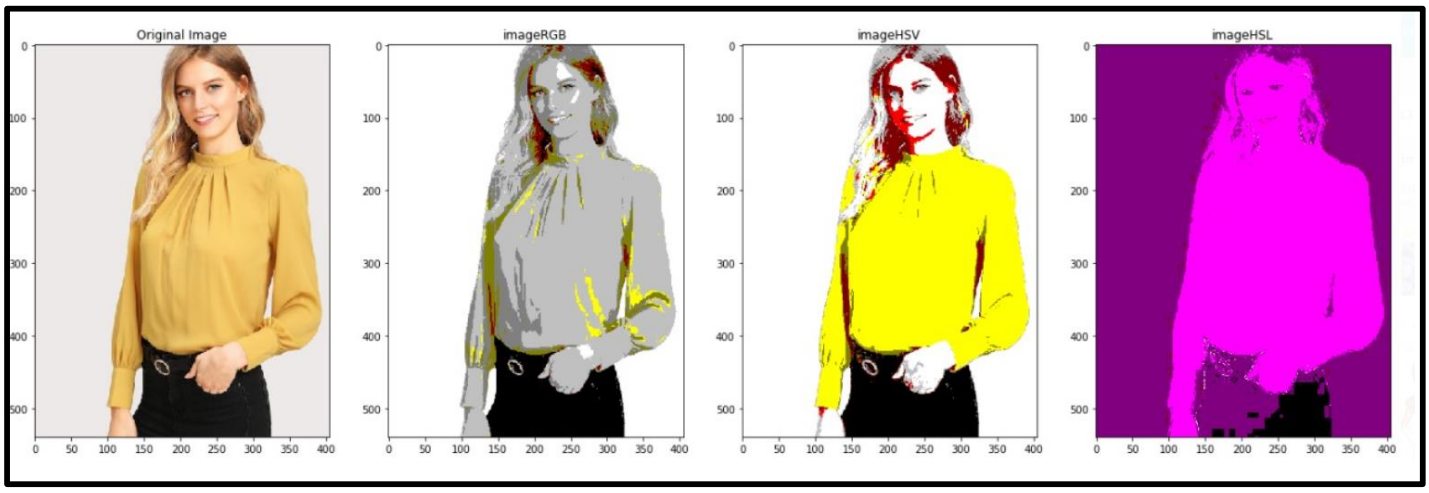


Figure 4-16 - Color Models Applied to Full Outfit Image

In the above figure each pixel in the original image (left most image) is assigned a color label using one of the three methods and this label is depicted in the three other images (pixels assigned a certain color are replaced with that color).

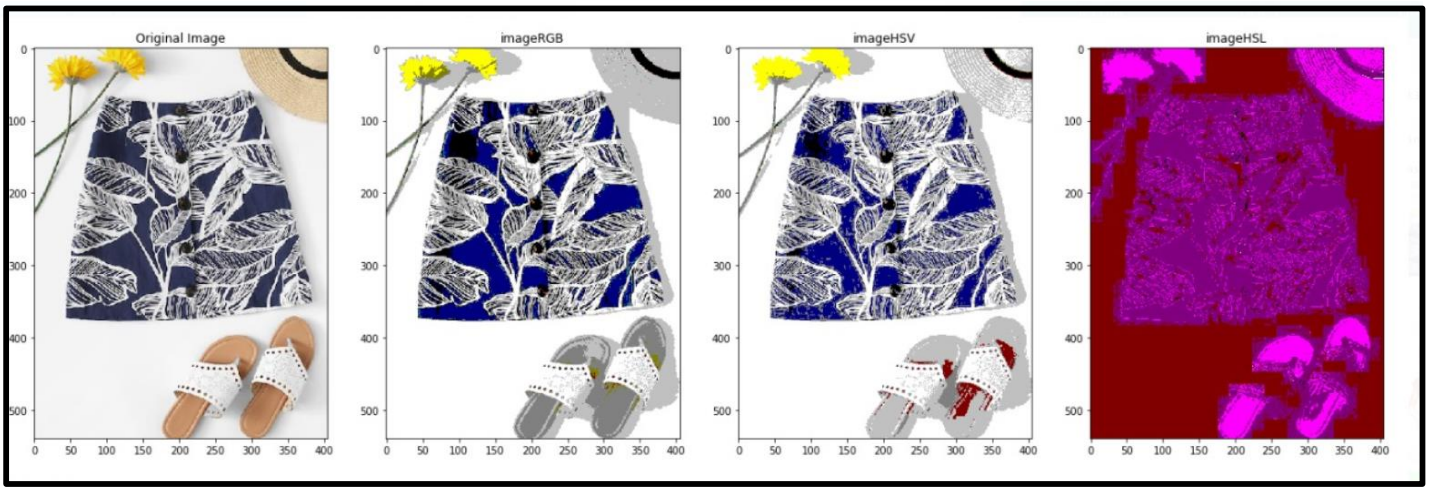


Figure 4-17 - Color Models Applied to Single Piece Image

We found that the best results were computed from the HSV model so we used it with all the images. Figure (4-18) and figure (4-19) show how the module takes a full outfit image, segment it to two separate pieces: the blouse and the pants then extracts the color for every piece

Figure 4-18 - Blouse Color Extraction

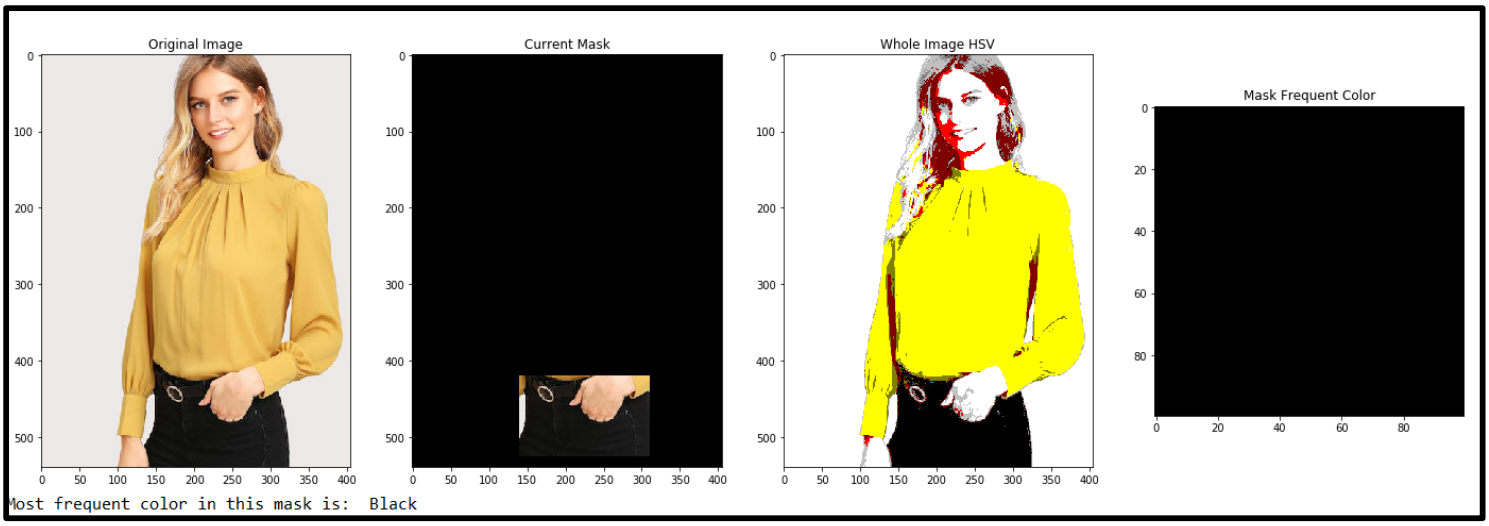


Figure 4-19 - Pants Color Extraction

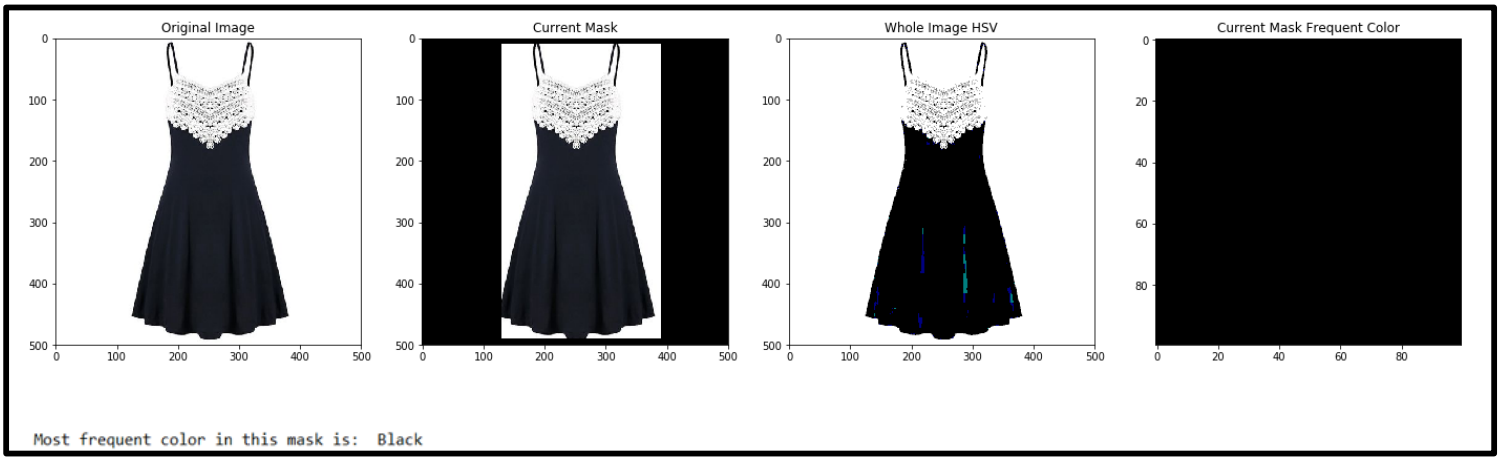


Figure 4-20 - Single Piece Image Color Extraction

Figure (4-20) shows how the module takes in as input a single piece image which is an image of a dress on its own without a model wearing it and find the most dominant color in that dress.

4.3.7 The Matching Module

The input data to this module is CSV files. There are two types of CSV files: the first type are files containing the colored-pieces in each image and the second type are files containing the colors only in each image. Each row in the CSV file correspond to an image in the database. These CSV files are built from information and features extracted from the crawled images of fashion bloggers and trending clothes from several websites. The first type of CSV files is used to generate outfit-matching rules while the second type is used to find colors matching rules.

Figure (4-21) shows an example of colored-pieces CSV files.

	A
1	Cyan-dress
2	Blue-skirt brown-t-shirt
3	Cyan-skirt Pink-t-shirt
4	Red-dress
5	Blue-t-shirt Red-skirt
6	Gray-skirt Red-blouse
7	Blue-t-shirt Gray-scarf Red-skirt
8	white-blouse black-scarf
9	black-t-shirt Yellow-shorts
10	black-pants Pink-t-shirt
11	Blue-blouse Blue-shorts
12	Blue-t-shirt black-blouse Red-skirt
13	Red-t-shirt Gray-leggings
14	Blue-blouse black-pants
15	Yellow-t-shirt Gray-stockings Blue-coat Yellow-skirt

Figure 4-21 - Colored-Pieces CSV File

	A
1	brown Red Cyan
2	brown Gray Blue
3	white Red Cyan Pink
4	black Red
5	brown Red Blue
6	black Red Gray
7	Red Gray Blue
8	black Red white
9	black Red Yellow
10	black white Pink
11	Red Gray Blue
12	black Red Gray Blue
13	black Red Gray

Figure 4-22 - Colors CSV File

Figure (4-22) shows an example of colors CSV files. These CSV files are then passed to the Eclat algorithm, which generates the matching rules based on the current trends. To generate these rules there are some metrics that are used:

1) **Support**:

   Is an indication of how frequently the item set appears in the dataset.

   Rule: NBoth/NTotal

2) **Confidence:**

   Is an indication of how often the rule has been found to be true.

   Rule: NBoth/NLeft

3) **Lift:**

   The ratio of the observed support to that expected if X and Y were independent

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)}$$

4) **Conviction:**

   A high conviction value means that the consequent is highly depending on the antecedent. It can be used to ensure the results of the lift.

$$\text{conv}(X \Rightarrow Y) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \Rightarrow Y)}.$$

5) **Rule Power Factor (RPF):**

   Rule Power Factor is an indication of how intense a rule's items are associated with each other in terms of positive relationship. Rule Power Factor is defined as:

$$\text{rpf}(X \Rightarrow Y) = \text{supp}(X \cup Y) * \text{supp}(X \cup Y)/\text{supp}(X)$$

$$= \quad \text{confidence(A->B)*supp(A} \cup \text{B)}$$

   A case that shows the importance of RPF:

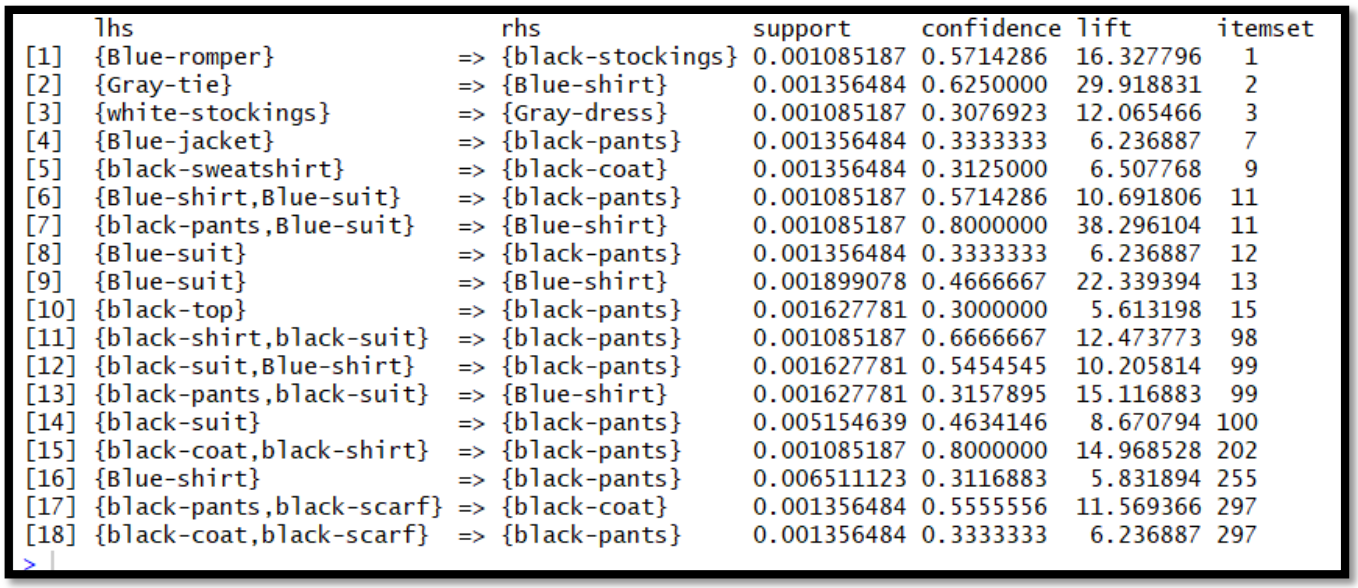
a) If “item A” appeared in 20 transactions and B in 50 out of total 100 transactions and item A and B both together appear 15 transactions. Then conf(A->B) = 0.15/0.2=0.75 = 75%.

b) If “item A” appeared in 30 transactions and B in 60 out of total 100 transactions and item A and B both together appear 20 transactions. Then conf=0.2/0.3=0.66 = 66%.

However, in case (b), both antecedent and consequent item’s occurrences increased individually and also increased in association (both A and B items together purchased) occurrences. While interest measure confidence says surprisingly that case (a) is more important (75%) than case (b) 66%. If we take the help of Rule Power Factor (RPF):
(a) 0.75*0.15= 0.11
(b) 0.66*0.2= 0.13
RPF, correctly judge that case (b) is more important.

To generate the rules of the algorithm, perform the following steps:

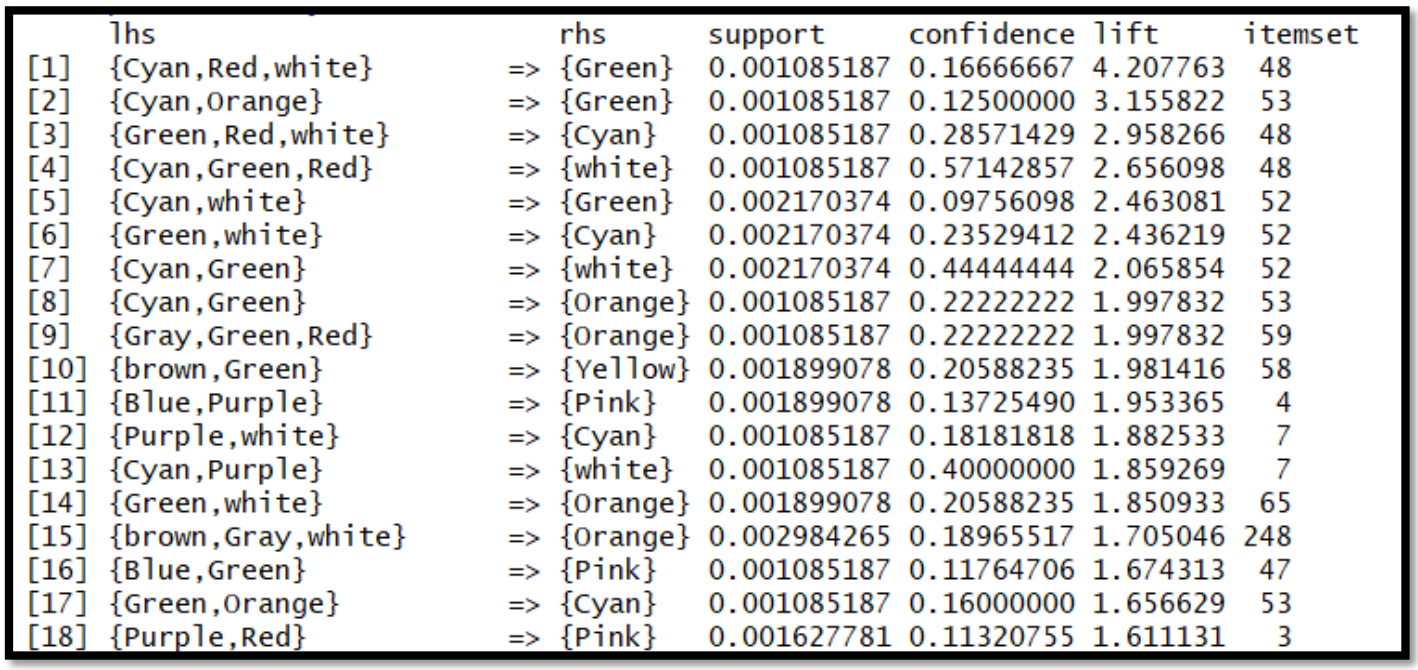
1) It finds the frequent items in the dataset based on a minimum support value.
2) From the frequent item set, it generates all the possible rules using depth-first search.
3) It then calculates the confidence of each rule generated and discard the rules with confidence less than a specified minimum confidence value.
4) It then returns the accepted rules as the outfit matching rules based on the current trends



	lhs		rhs	support	confidence	lift	itemset
[1]	{Blue-romper}	=>	{black-stockings}	0.001085187	0.5714286	16.327796	1
[2]	{Gray-tie}	=>	{Blue-shirt}	0.001356484	0.6250000	29.918831	2
[3]	{white-stockings}	=>	{Gray-dress}	0.001085187	0.3076923	12.065466	3
[4]	{Blue-jacket}	=>	{black-pants}	0.001356484	0.3333333	6.236887	7
[5]	{black-sweatshirt}	=>	{black-coat}	0.001356484	0.3125000	6.507768	9
[6]	{Blue-shirt,Blue-suit}	=>	{black-pants}	0.001085187	0.5714286	10.691806	11
[7]	{black-pants,Blue-suit}	=>	{Blue-shirt}	0.001085187	0.8000000	38.296104	11
[8]	{Blue-suit}	=>	{black-pants}	0.001356484	0.3333333	6.236887	12
[9]	{Blue-suit}	=>	{Blue-shirt}	0.001899078	0.4666667	22.339394	13
[10]	{black-top}	=>	{black-pants}	0.001627781	0.3000000	5.613198	15
[11]	{black-shirt,black-suit}	=>	{black-pants}	0.001085187	0.6666667	12.473773	98
[12]	{black-suit,Blue-shirt}	=>	{black-pants}	0.001627781	0.5454545	10.205814	99
[13]	{black-pants,black-suit}	=>	{Blue-shirt}	0.001627781	0.3157895	15.116883	99
[14]	{black-suit}	=>	{black-pants}	0.005154639	0.4634146	8.670794	100
[15]	{black-coat,black-shirt}	=>	{black-pants}	0.001085187	0.8000000	14.968528	202
[16]	{Blue-shirt}	=>	{black-pants}	0.006511123	0.3116883	5.831894	255
[17]	{black-pants,black-scarf}	=>	{black-coat}	0.001356484	0.5555556	11.569366	297
[18]	{black-coat,black-scarf}	=>	{black-pants}	0.001356484	0.3333333	6.236887	297

Figure 4-23 – Trending Outfit Matching Rules

Figure (4-23) shows some of the trending outfit-matching rules generated from Eclat algorithm.



	lhs		rhs	support	confidence	lift	itemset
[1]	{Cyan,Red,white}	=>	{Green}	0.001085187	0.16666667	4.207763	48
[2]	{Cyan,Orange}	=>	{Green}	0.001085187	0.12500000	3.155822	53
[3]	{Green,Red,white}	=>	{Cyan}	0.001085187	0.28571429	2.958266	48
[4]	{Cyan,Green,Red}	=>	{white}	0.001085187	0.57142857	2.656098	48
[5]	{Cyan,white}	=>	{Green}	0.002170374	0.09756098	2.463081	52
[6]	{Green,white}	=>	{Cyan}	0.002170374	0.23529412	2.436219	52
[7]	{Cyan,Green}	=>	{white}	0.002170374	0.44444444	2.065854	52
[8]	{Cyan,Green}	=>	{Orange}	0.001085187	0.22222222	1.997832	53
[9]	{Gray,Green,Red}	=>	{Orange}	0.001085187	0.22222222	1.997832	59
[10]	{brown,Green}	=>	{Yellow}	0.001899078	0.20588235	1.981416	58
[11]	{Blue,Purple}	=>	{Pink}	0.001899078	0.13725490	1.953365	4
[12]	{Purple,white}	=>	{Cyan}	0.001085187	0.18181818	1.882533	7
[13]	{Cyan,Purple}	=>	{white}	0.001085187	0.40000000	1.859269	7
[14]	{Green,white}	=>	{Orange}	0.001899078	0.20588235	1.850933	65
[15]	{brown,Gray,white}	=>	{Orange}	0.002984265	0.18965517	1.705046	248
[16]	{Blue,Green}	=>	{Pink}	0.001085187	0.11764706	1.674313	47
[17]	{Green,Orange}	=>	{Cyan}	0.001085187	0.16000000	1.656629	53
[18]	{Purple,Red}	=>	{Pink}	0.001627781	0.11320755	1.611131	3

Figure 4-24 - Trending Colors Matching Rule

Figure (4-24) shows some of the trending colors-matching rules generated from Eclat algorithm.

We did not use data mining and analytics only to find the trending colors matching rules and the outfits matching rules, but we also used it to find the most trending colors that the fashion icons wear for each category of clothes' pieces. Figure (4-25) shows the five most trending colors in the Jackets category.

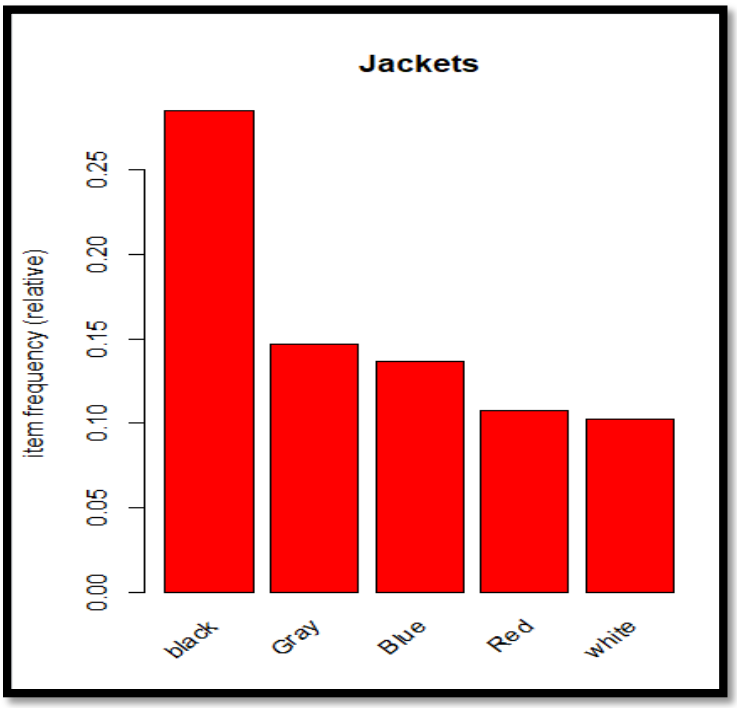


Figure 4-26 - Trending Colors for Jackets

Figure (4-26) shows the five most trending colors in the Tops category

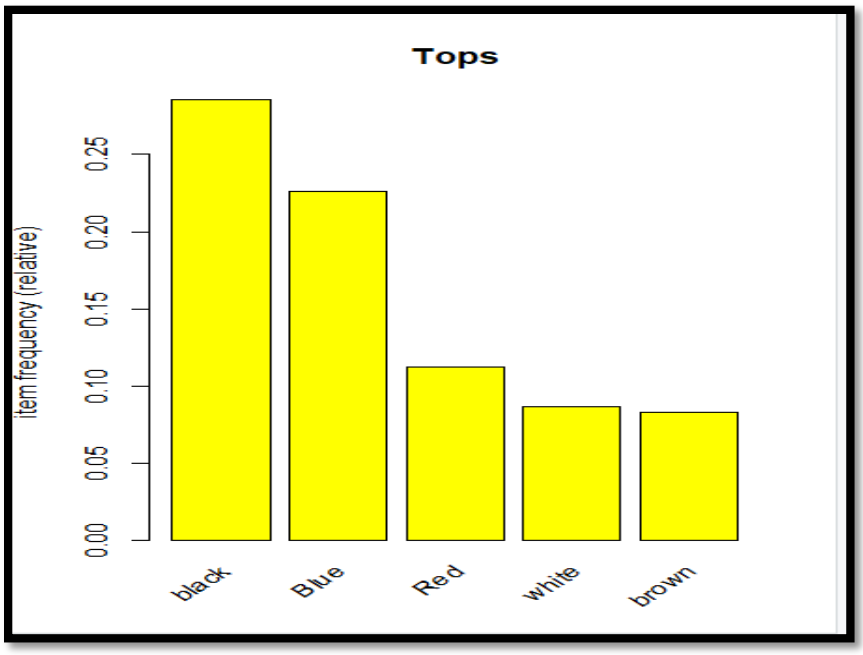


Figure 4-27 - Trending Colors for Tops

4.3.8 The Mobile Application

The Mobile Application: It is the integration module, which connects all the components together to produce a complete final product. We chose to develop a mobile application not a website or a desktop application because it would be the most user-friendly. It is much easier for a user to upload his image, which he can snap from the camera instantaneously or choose from his camera roll to the application and wait for a few seconds to match an outfit with it or find pieces similar to it based on what he wants. It is the fastest way to meet the user's needs and it guarantees the best results since the mobile camera has higher resolution than the laptop camera. A native mobile application is also more responsive than a website when dealing with uploading and retrieving images. Since our users are divided into iOS users and Android users, so we decided to develop a mobile application for both of them. We have four different functionalities offered to the user: 1) "Match an Outfit": the user chooses an image either chosen from his camera roll or snapped with his camera instantaneously and the application match this piece to

him with a whole outfit. It suggests to him how to wear this piece with other pieces that match the current trends. 2) “Now Trending”: which views to the user the up to date fashionable clothes in the stores and they are classified into categories: dresses, skirts, blouses, pants, jackets, etc., so that the user can be updated with the latest styles in each type of clothes. 3) “Find Similar Pieces”: the user chooses an image either chosen from his camera roll or snapped with his camera instantaneously and the application provides to him pieces similar to it found in the stores. 4) “Find Similar Outfit”: the user chooses an image either chosen from his camera roll or snapped with his camera instantaneously and the application suggests to him outfits that look like the one he searched with. It suggests similar overall styles from what is trending now.

5. Languages, Tools& Libraries

5 Languages, Tools& Libraries

5.1 Overview

This chapter talks about the tools we used in the project to implement the different modules of the project, and the tools used in the application. We will also mention the programming languages and libraries used.

5.2. Tools

1) Anaconda Spyder

Description	It is scientific Python Development Environment, and a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging and introspection features.
Usage	We used it in developing the modules, crawler and the preprocessing of the databases.

2) Pycharm

Description	Pycharm is a cross-platform IDE used in computer programming, specifically for the python language. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems.
Usage	We used it in developing the modules, crawler and the preprocessing of the databases.

3) NoSql Manager for MongoDB

Description	Is a tool for Mongo database management, administration and development. It unites user friendly interface and Shell power. Intuitive interface and high performance of the desktop application save time for beginners and professional database developers and administrators.
Usage	We used it to build our main database that is filled with the crawled information and some extra information from the CNN modules, and we deal with it to enter and retrieve images within the application front-end and backend.

4) VS-Code

Description	Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Gitcontrol and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. It allows the user to install extensions that add additional functionality. Visual Studio Code is based on Electron, a framework that is used to deploy Node.js applications.
Usage	we used it in development of the mobile application front-end and back-end. We installed the react-native package and the Node.js package for VS code to help in auto completion and syntax error finding.

5) Postman

Description	It is a tool used to test RESTful APIs. It offers a sleek user interface with which to make HTML requests.
Usage	we used it to test the back-end functionalities by sending requests to the back-end from Postman and receiving the responses for debugging and inspection purposes.

6) Expo Client

Description	It is a tool from Expo where you can preview your application on your mobile device while still developing it. It is a substitute of using iOS or Android simulator on the computer. It has the advantage of hot reloading where the application automatically reloads on your mobile device once you do any changes in the code.
Usage	We used Expo Client in the development of the front-end of the mobile application and for its debugging.

7) RStudio

Description	RStudio is a free development environment for R, a programming language for statistical computing and graphics.
Usage	We used RStudio in the Big Data Analytics and Mining where we applied Association Rules Mining on the colors and colors-pieces combinations resulted from the crawler to get the matching rules.

5.2 Libraries and Frameworks

1) Tensorflow

Description	TensorFlow is an open source software library that was developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.
Usage	It is used in the implementation and training of our CNN and FCN modules.

2) Keras

Description	Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow.
Usage	It is used in the implementation and training of our CNN and FCN modules.

3) React Native

Description	React Native is a framework created by Facebook that allows you to develop native mobile apps for iOS and Android with a single JavaScript codebase**.**
Usage	It is used in the development of the front-end of the mobile application.

4) Express

Description	Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software. It is designed for building web applications and APIs.
Usage	It is used in the development of the back-end of the mobile application.

5.3. Programming Languages

1) Python

Description	Python is an interpreted, dynamic, high-level, general purpose language that is object-oriented and platform-independent. Python offers dynamic data-type and read-made classes and interfaces to many system-calls and libraries.
Usage	Python was the main programming language that we used in our project. We used it to preprocess the datasets, to build our crawler and to build all our modules.

2) Javascript

Description	It is a high-level, interpreted that was initially used in the client-side of web browsers only. JavaScript engines are now used as the server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications.
Usage	The front-end framework React Native is based on React.js library, which is written in JavaScript. In addition, the back-end framework Express.js is written in JavaScript

3) R Language

Description	R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.
Usage	We used it in performing the data analysis to get the matching rules by applying association rules

6. Testing Plan and Results

6 Testing Plan & Results

6.1 Introduction

In this section we shall go over how we planned for and performed the testing process. To begin with, in building each module, we have validated and tested each module separately via a variety of testing techniques which will be discussed under each module. Moreover, after integrating the modules into the application we performed black box testing on the whole application using many different scenarios which covers all the functionalities of the application.

6.2 Semantic Segmentation Module

6.2.1 Data Used

As we discussed in chapter 4, we used 2 datasets, CFPD and Khurana, to train our model against. Each of which, we have split the datasets into 3 sections; training, validation and testing in percentages of 70%, 15% and 15% respectively.

6.2.2 Training and validation:

The training dataset is explicitly used to train the model, while the validation dataset had two main objectives.

The first of the objectives was to tune hyper-parameters using the validation portion of the dataset. The most important hyper-parameter was the learning rate used in training.

Table 6-1 Learning rate validation accross CFPD Validation Dataset

Learning Rate	Validation Loss	Validation Accuracy
0.0001	0.3266	0.8985
0.0005	0.3062	0.9007
0.0008	0.3286	0.8907
0.0015	0.3715	0.8789
0.001	0.3019	0.9008
0.002	0.4003	0.8681

After experiments, as shown in table xxxx, we found that the best learning rate (achieving smallest loss and largest accuracy) to use is 0.001.

Secondly, the validation portion of the dataset was used as a reference to know when to stop training to avoid over fitting and keep the model generalized. This is achieved by checking at which epoch in training does the model achieve the least validation loss and stop training at that epoch. This is illustrated in figure(6-1) below.

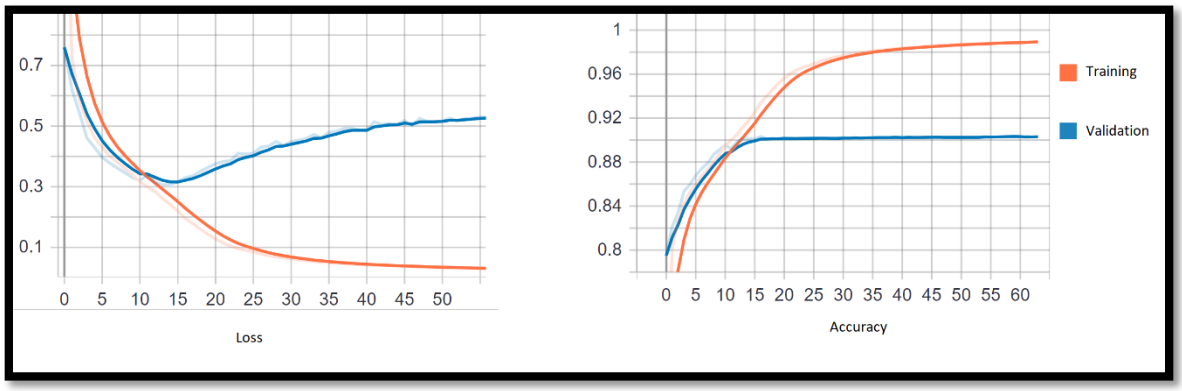


Figure 6-1 Training and validation, loss and accuracy

In the above figure the least validation loss is achieved at the 15th epoch therefore we have stopped training at the 15th epoch before overfitting occurs. If we had continued training for more epoch it would have been shown that the training accuracy would increase while the validation accuracy would decrease due to overfitting.

6.2.3 Choosing Merging Method

In merging the 2 streams for segmentation we experimented with multiple methods. The methods vary based on either the type of merging layer (whether we sum outputs of the 2 streams or concatenate their outputs), the number of convolutional layers following the merging layer and the size of their kernels. The different methods we experimented on and their results are illustrated the table below.

Table 6-2 Different merging methods

Merging Layer	Convolutional Layers Following Merging Layer	Validation Loss	Validation Accuracy
Sum	2 conv (5*5)	0.36	0.8957
Sum	1 conv (1*1) + 1 conv (5*5)	0.42	0.888
Concatenation	1 conv (1*1)	0.3019	0.9008
Concatenation	1 conv (5*5)	0.383	0.897
Concatenation	1 conv (1*1) + 1 conv (5*5)	0.4886	0.8848

6.2.4 Testing the Model

6.2.4.1 Evaluating Accuracy

We used the test portion of the dataset to obtain a final evaluation value for our model. Our model achieved an accuracy of 93% which compares greatly with other models as shown in the below table.

Table 6-3 Accuracies of different implementations of segmentation model for fashion pieces

	Accuracy On CFPD Dataset
OE	92.3
PaperDoll	87.1
SSL	88.5
DLV2 (ResNet)	89.9
DLV2 (VGG)	89.2
FCN-8	91.6
Ours	92.1

An important note to be taken into consideration is that the dataset itself is not very accurate when it comes to pixel positions. On the other hand, the model performs better in regards to pixel positions as it generalizes these variations in relevant pixel positions in the dataset producing more accurate shapes of the fashion items. This means that in actuality the accuracy of the models in the above table maybe be better than they seem.

6.2.4.2 Confusion Matrix

To further understand the models results, we have created a confusion matrix using the test portion of the dataset. This allows us to understand the (categories/ fashion items) that are commonly mistaken as other specific categories as shown in the below figure.

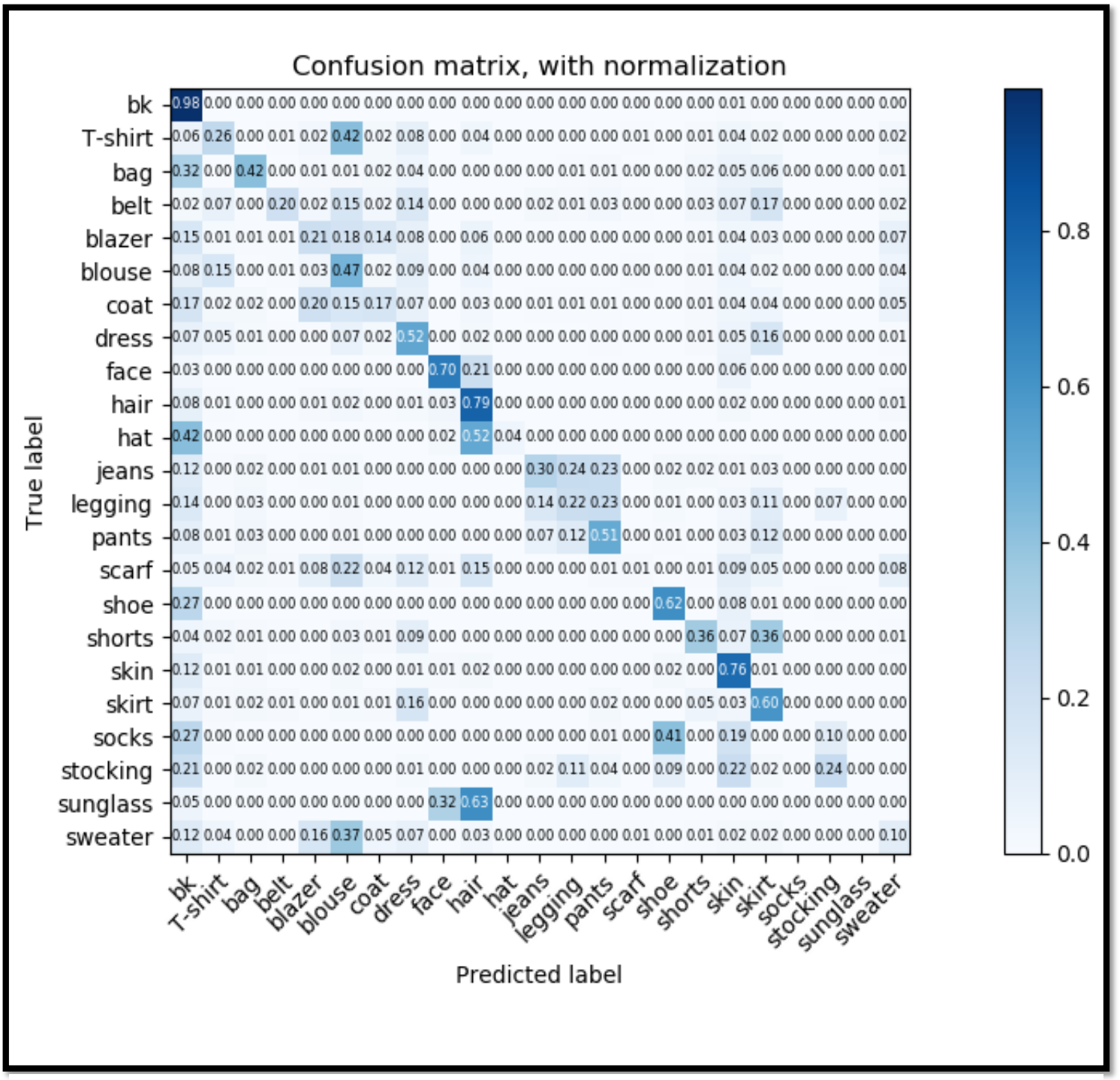


Figure 6-2 Confusion matrix of segmentation model

As an example to help understand the meaning of the confusion matrix shown in the figure above, 52% of pixels in a dress are classified correctly while 16% are classified as a skirt. The previous percentages are important for analyzing the model, as it depicts that sometimes the dress are classified in correctly as a dress which can make sense as they do have similar shapes.

However, a percentage such as (7% of a dress is classified incorrectly as background) is not very meaningful in understanding the confusion of the model as that inaccuracy may occur from the edge pixels (which maybe already be incorrect in the dataset, and our model is actually correct about these pixels).

6.3 Similar Outfits Module

6.3.1 Data Used

We used a dataset called CFPD (Color Fashion Parsing Dataset) of K categories. CFPD contains 2682 pixel-annotated images. Where each pixel belongs to one clothing item. Moreover, we need now image level annotation to know the ground truth categories that exist in the image. We divided the data into 3 sets 70% training set of 1877 images, 15% validation set of 402 images and 15% test set of 402 images.

6.3.2 Training and validation:

We extracted for each image an encoded vector of K categories. Where we put 1 if the clothing item exists in the photo and 0 otherwise. This encoded vector is used as label for the training image.

The loss function we used in training the models is Cross-entropy loss, or log loss that measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label, so predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

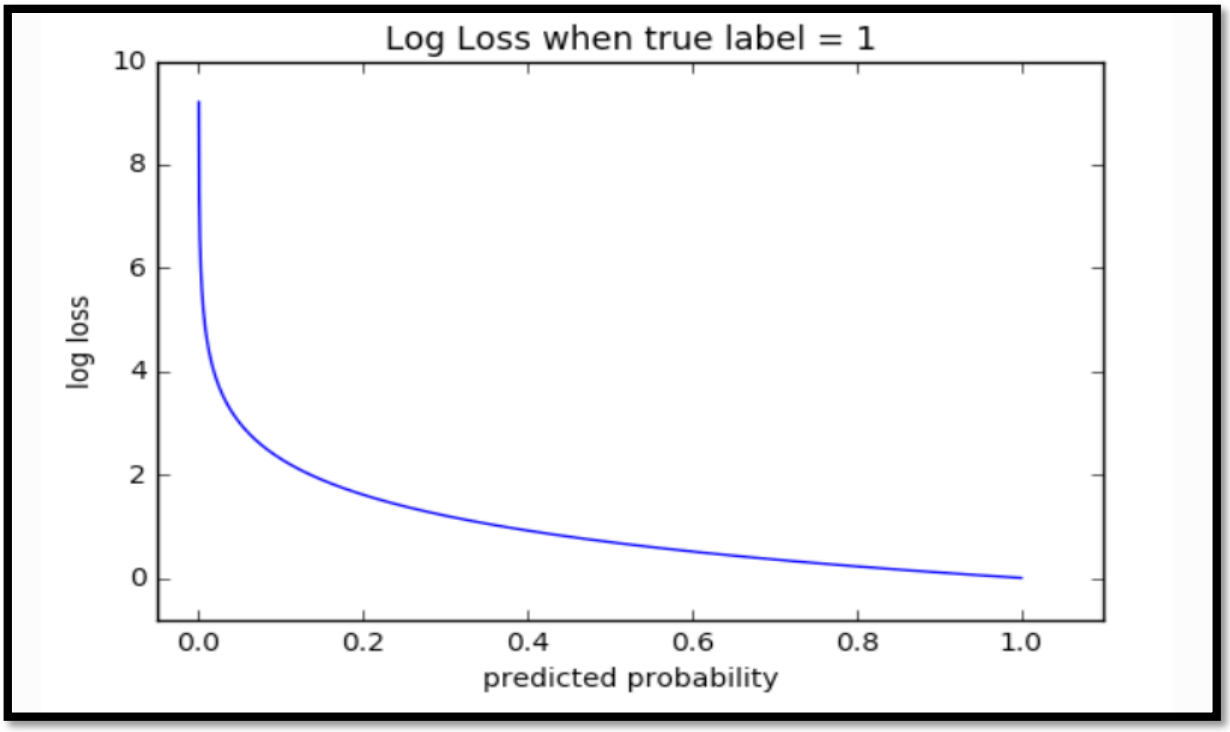


Figure 6-3 Cross Entropy Loss

We got training accuracy of 85% and validation accuracy of 83% before overfitting. Then we evaluated the model using the test set which gave relevant results that gave similar top N images in style to the query image. The following figures represent the training and validation accuracies and some results from the test set.

We tuned the hyper parameters to achieve the best results. We noticed that learning rate affects the model so by tuning we used learning rate of 1e-3. Moreover, we decided where to stop after plotting training vs validation accuracies as shown in the figure below to avoid overfitting.

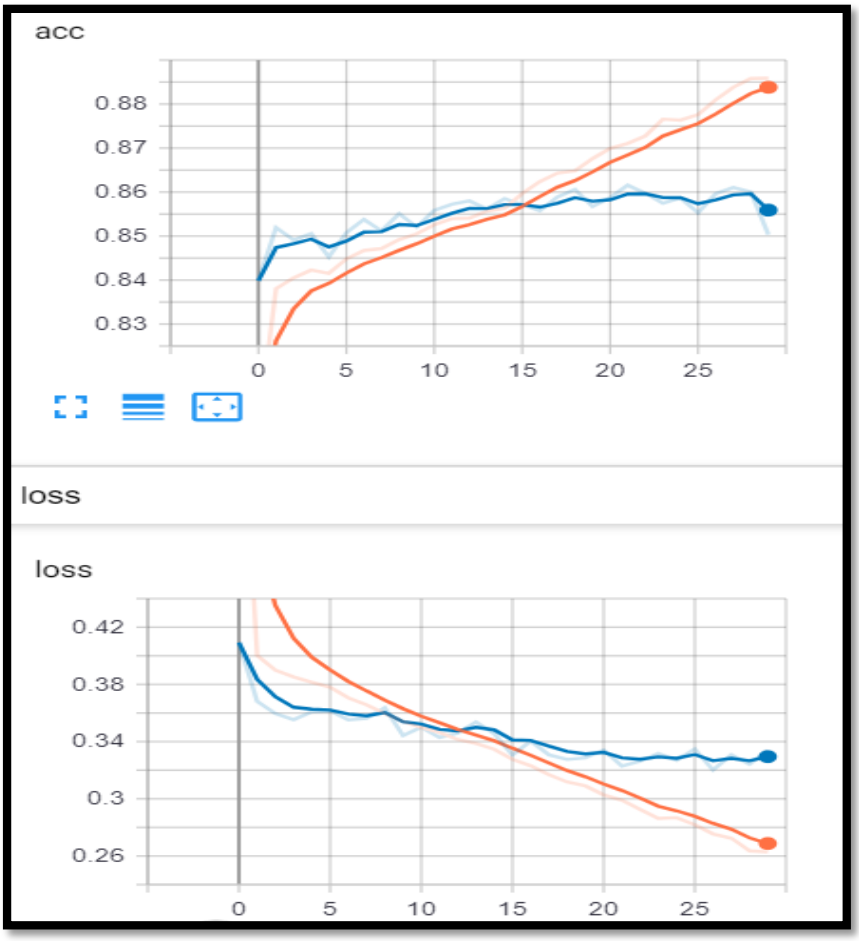


Figure 6-4 Training Similarity Model

6.3.3 Results and Discussion

The figures below show two different examples for the Similarity Retrieval model.

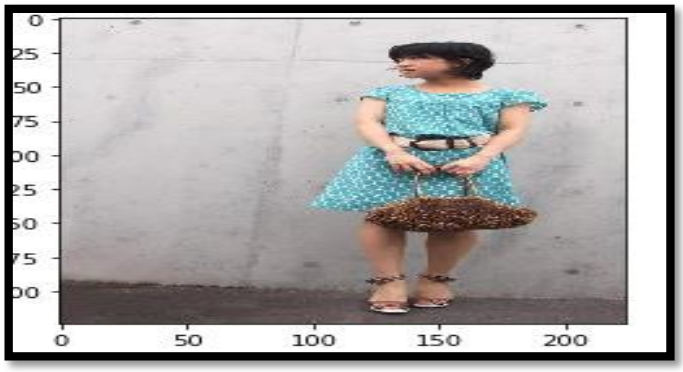


Figure 6-5 Query Image

Figure 6-6 Output of The Model: Results of Similar Styles

Figure 6-7 - Query Image

Figure 6-8 Output of The Model: Results of Similar Styles

6.4 Piece Identification Module

We classified between 6 categories which are (Blouses, Skirts, Dresses, Shorts, Pants, and Sweaters). So we chose P to be 6. The CNN was trained on 500 images for each category, the validation set was 100 images for each category and finally the training set was also 100 images for each category. We achieved training accuracy of 94 % on the Training Set, 93% on the Validation Set and 90% on the Test Set.

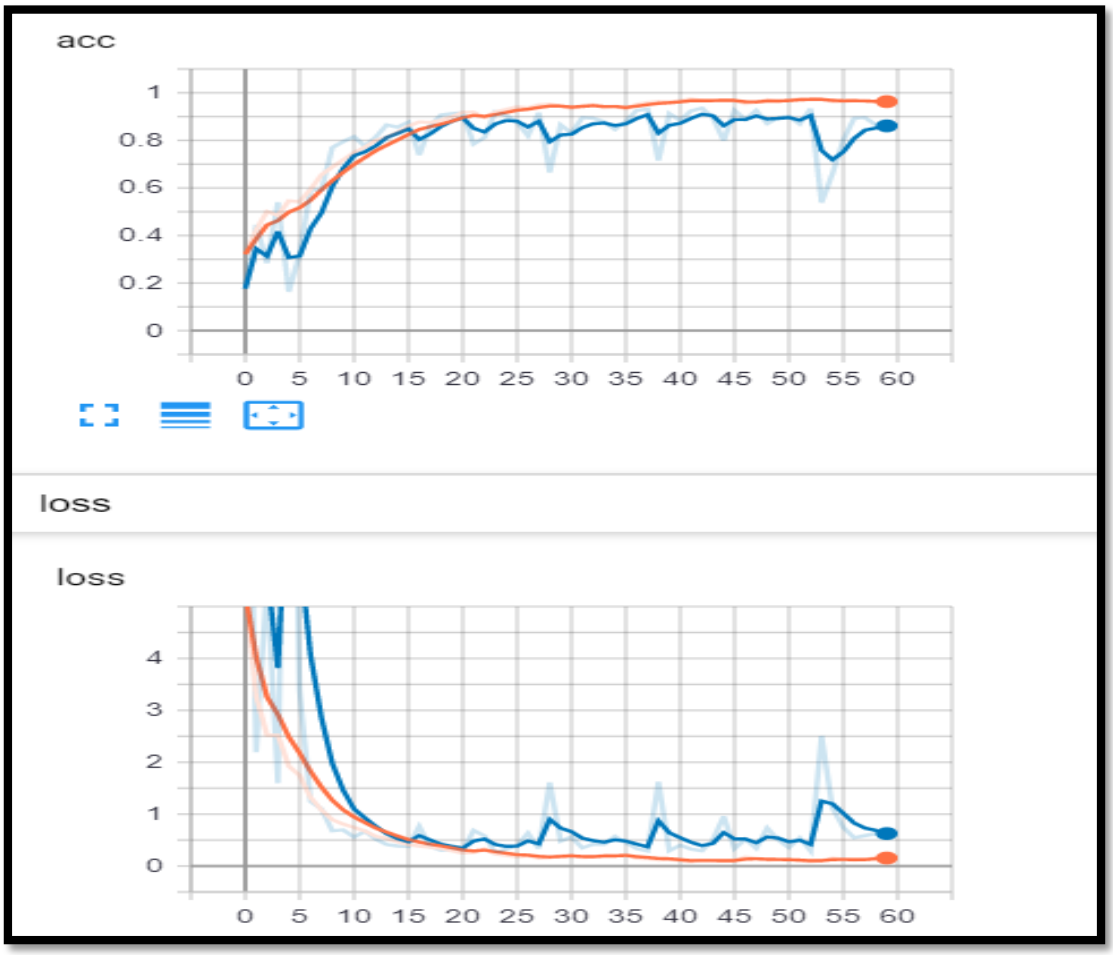


Figure 6-9 - Training Pieces Identification Model

6.5 Trending and Matching Module

6.5.1 Data Used

The data used for this module is CSV files. As explained in chapter4, we have two types of CSV files. The first type contains colored pieces labels for each image and the second type contains colors only for each image.

6.5.2 Results & Discussion

We have some metrics to evaluate the validity and relevance of the matching rules generated for each type of CSV files. These metrics were explained in chapter 4. Here are some criteria for the evaluation of each rule:

1) Lift value greater than one indicates that the rule did not come by coincidence
2) The higher the RPF, the higher the positive relationship between the consequent and the antecedent
3) Conviction values greater than one indicates that the rule did not come by coincidence and that the consequent is highly depending on the antecedent
4) The higher the confidence and support values for a rule, the more valid and relevant the rule is.

6.5.2.1 Analysis of Trending Outfit Matching Results

Figure (6-10) shows the lift and the support values for the first 18 rules generated for outfit matching. Note that the redder the circle is, the higher of the lift and the bigger the circle is, the higher the support. From the graph, we can see that rule 1 has the highest lift and rule 13 has the highest support.

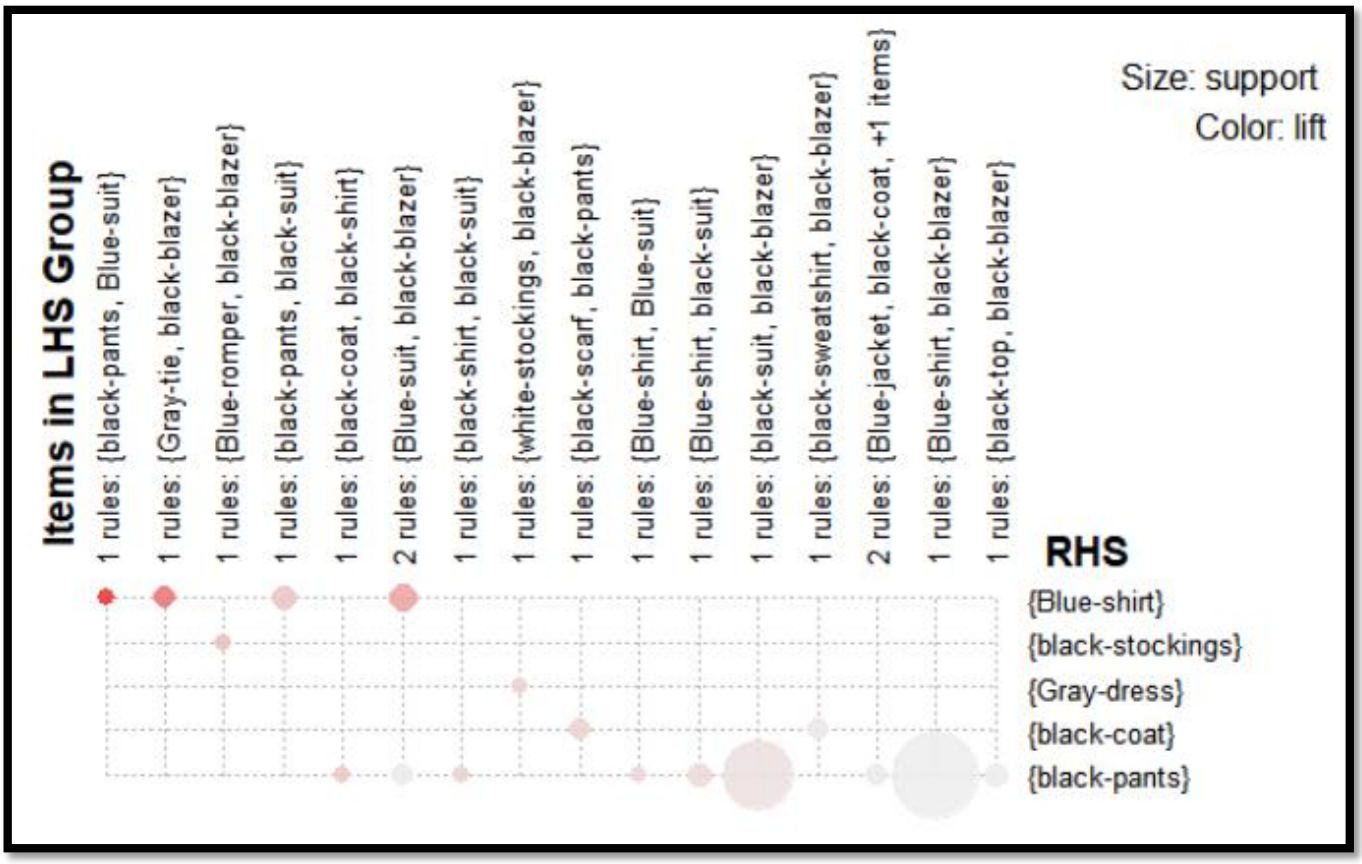


Figure 6-10 – Support and Lift of Outfit Matching Rules

Figure (6-11) shows the conviction values for the first 18 outfit matching rules. We can deduce that rule 1 has the highest conviction value.

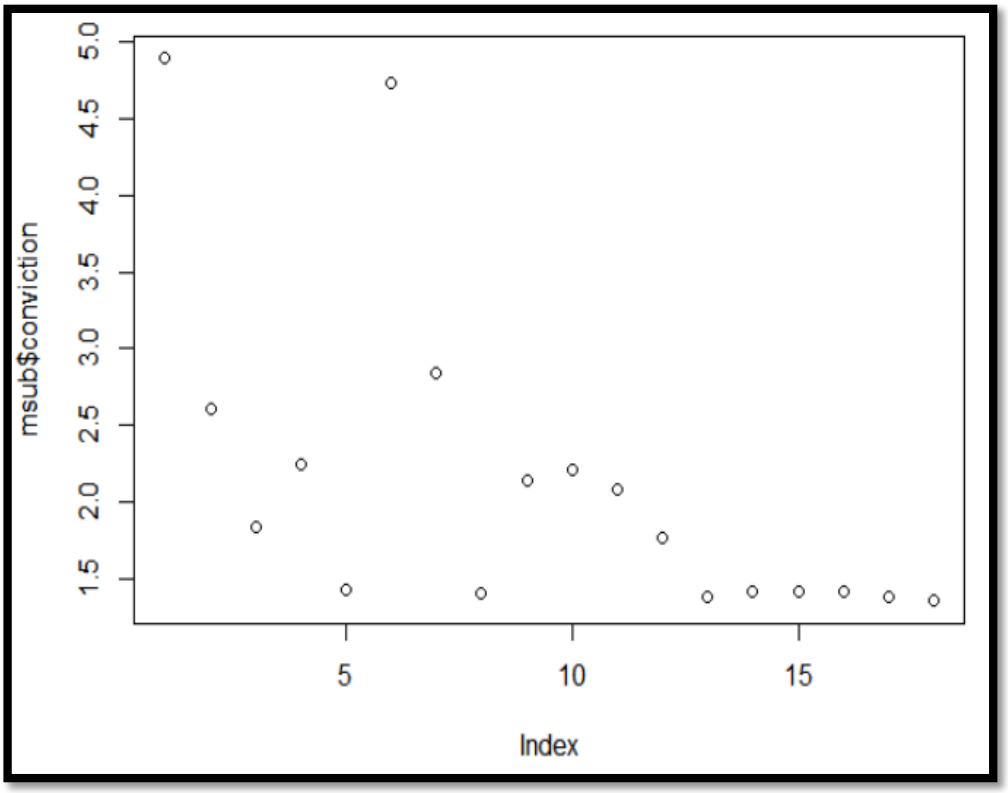


Figure 6-11 - Conviction of Outfit Matching Rules

Figure (6-12) shows the RPF values for the first 18 outfit matching rules. We can deduce that rule 12 has the highest RPF value.

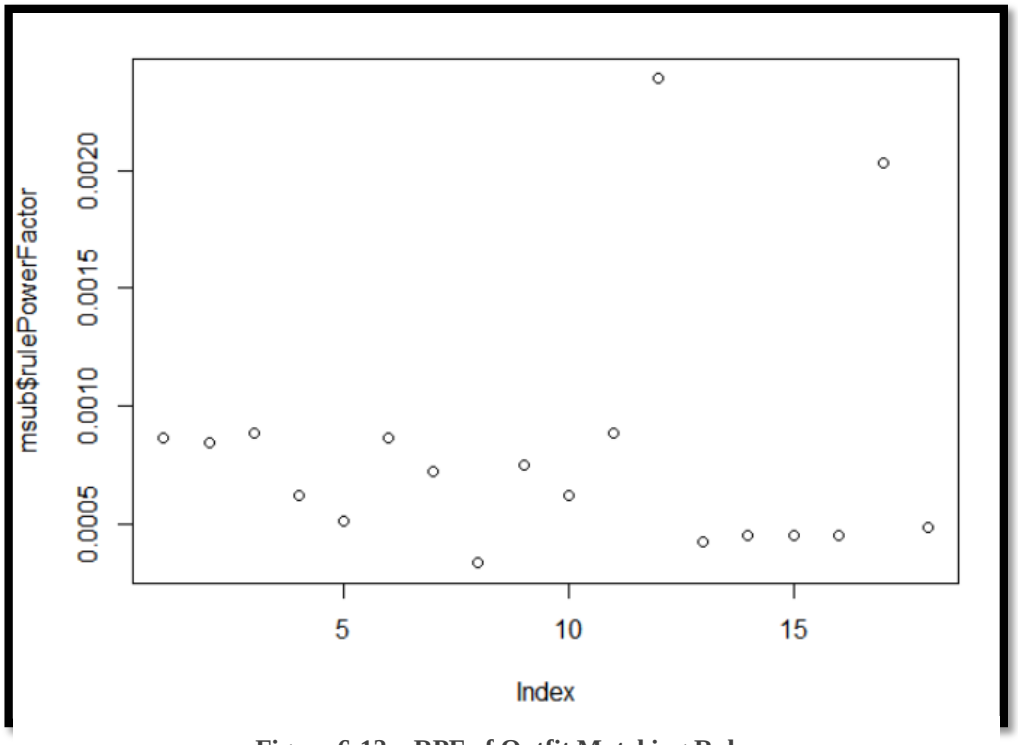


Figure 6-12 – RPF of Outfit Matching Rules

Figure (6-13) shows a scatter plot of the confidence, lift and support values for the first 18 outfit matching rules. Note that there are two rules with confidence values equal approximately 0.8, this is an indication of the goodness of these rules.

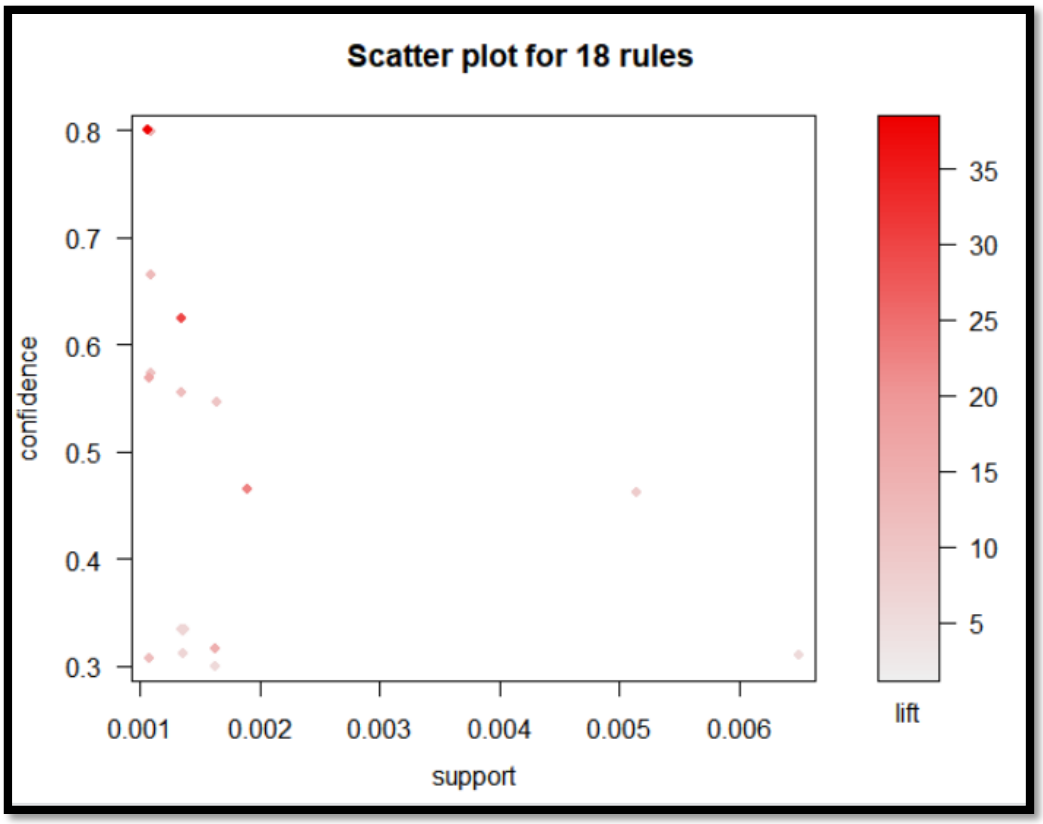


Figure 6-13 - - Scatter Plot of Outfit Matching Rules

6.5.2.2 Analysis of Trending Colors Matching Results

Figure (6-14) shows the lift for the first 18 rules generated for trending colors matching. From the graph, we can see that rule 1 has the highest lift value.

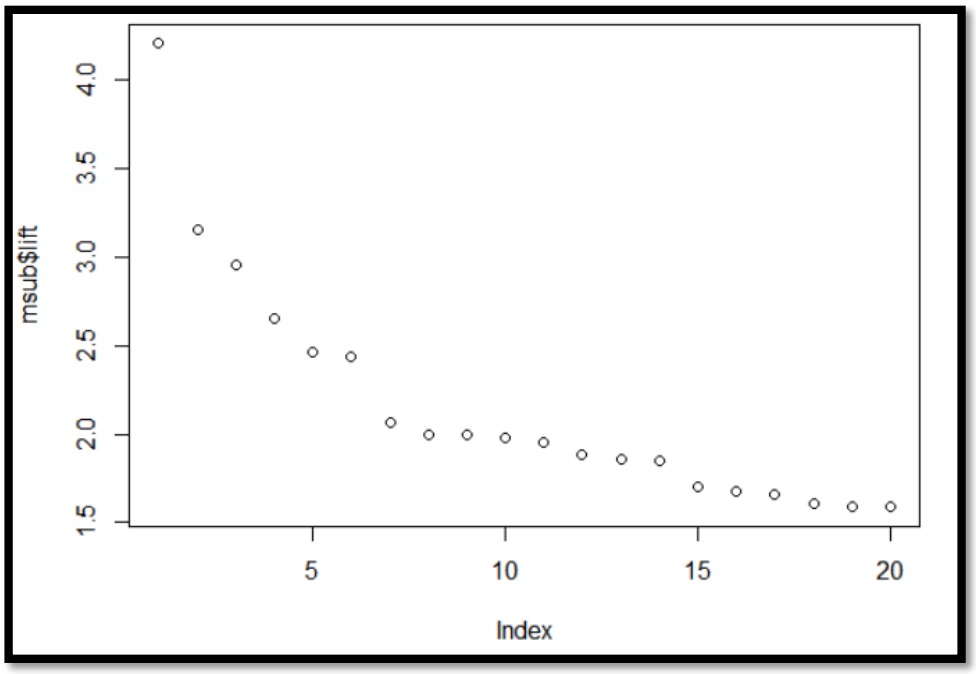


Figure 6-14 - Lift of Colors Matching Rules

Figure (6-15) shows the conviction for the first 18 rules generated for trending colors matching. From the graph, we can see that rule 4 has the highest conviction value.

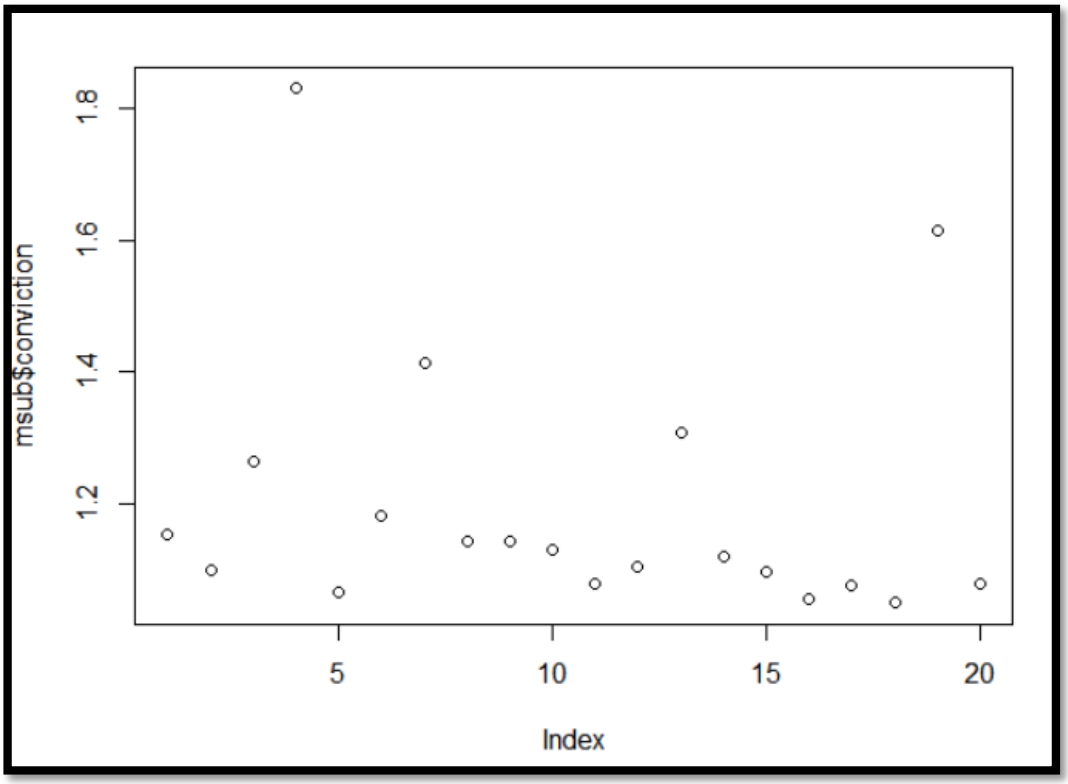


Figure 6-15 - Conviction of Colors Matching Rules

Figure (6-16) shows the RPF for the first 18 rules generated for trending colors matching. From the graph, we can see that rule 7 has the highest RPF value

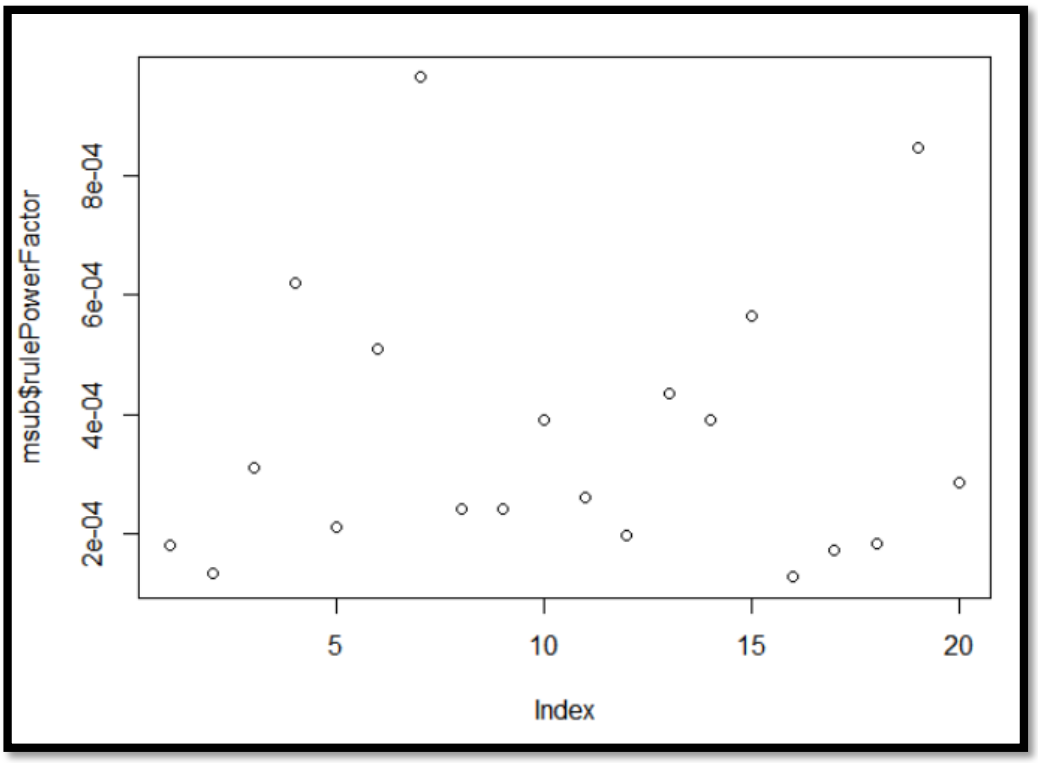


Figure 6-16 - RPF of Colors Matching Rules

Figure (6-17) shows a scatter plot of the confidence, lift and support values for the first 18 trending colors matching rules. Note that there is one rule with confidence values equal approximately 0.8, this is an indication of the goodness of this rule.

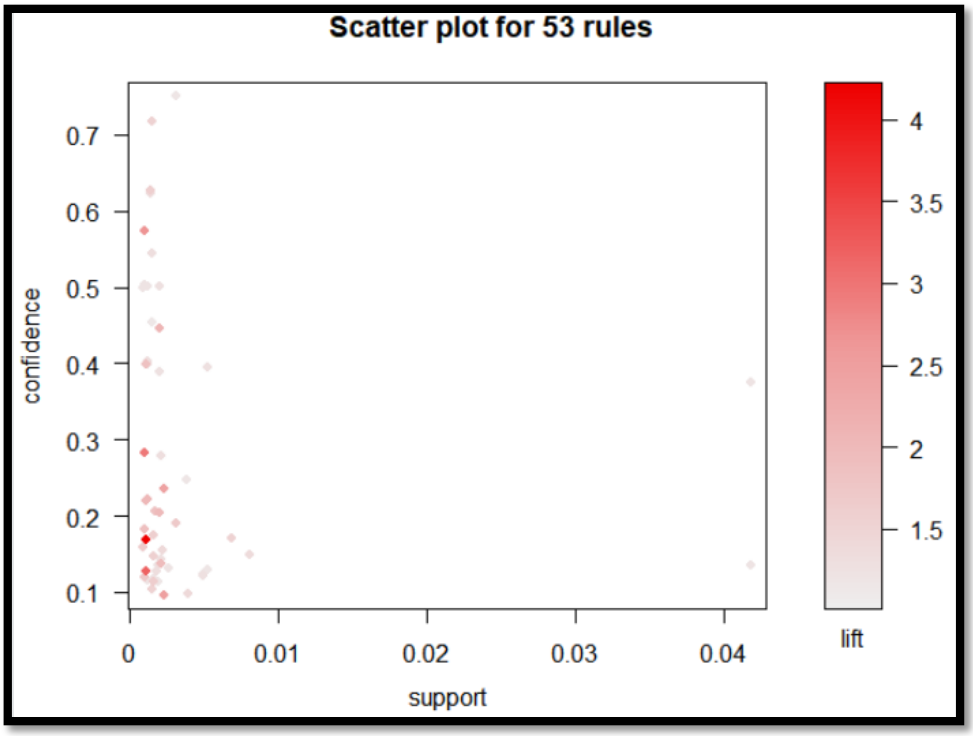


Figure 6-17 - Scatter Plot of Colors Matching Rules

6.6 Scenario Testing

6.6.1 Match an Outfit Scenario

In this scenario, the user chooses an image either snapped with the camera or chosen from the camera roll in order to match an outfit with a clothes' piece from his image. Figure (6-18) and figure (6-19) show the steps of processing this scenario in the system's modules.

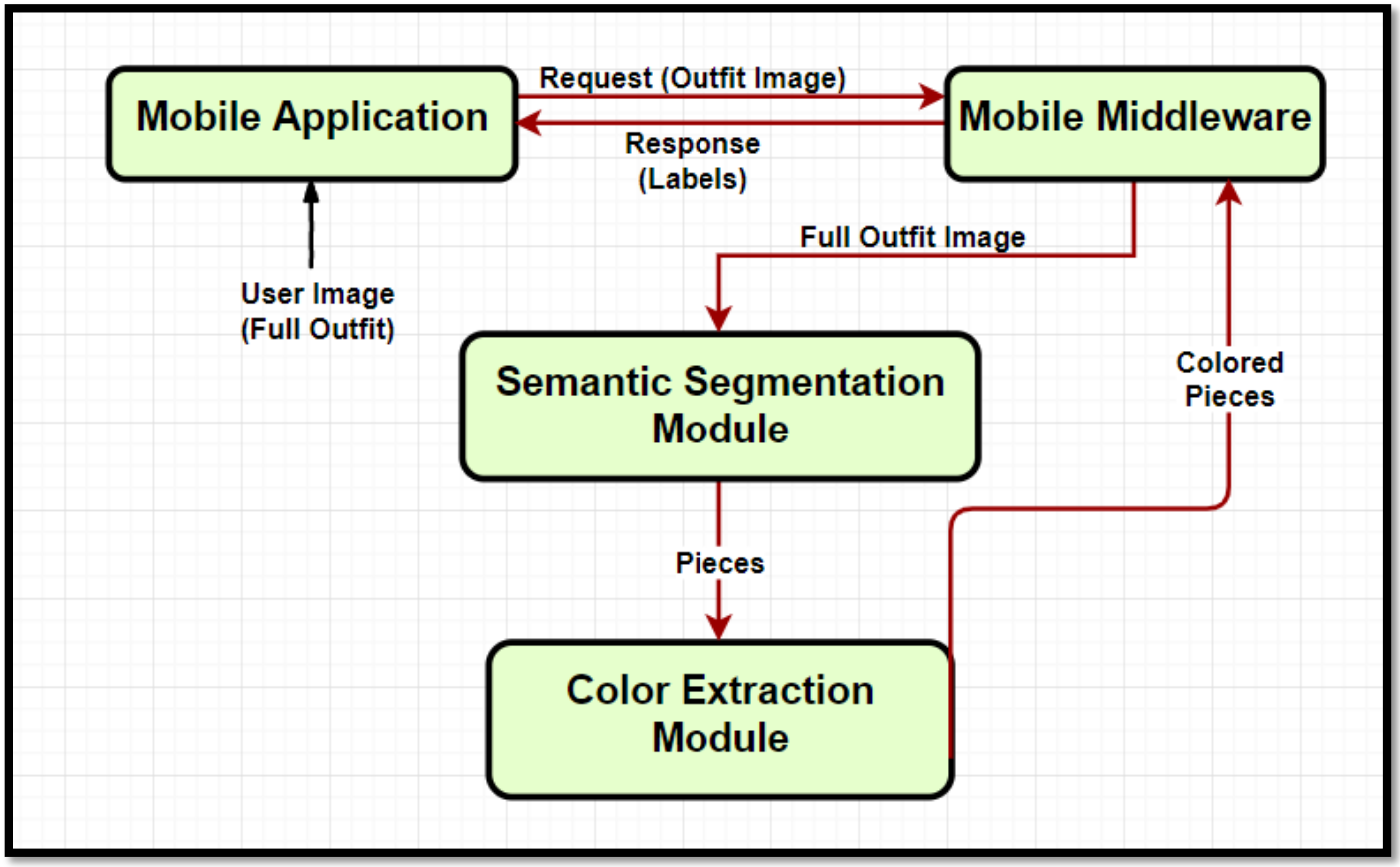


Figure 6-18 - Match an Outfit Scenario - Part1

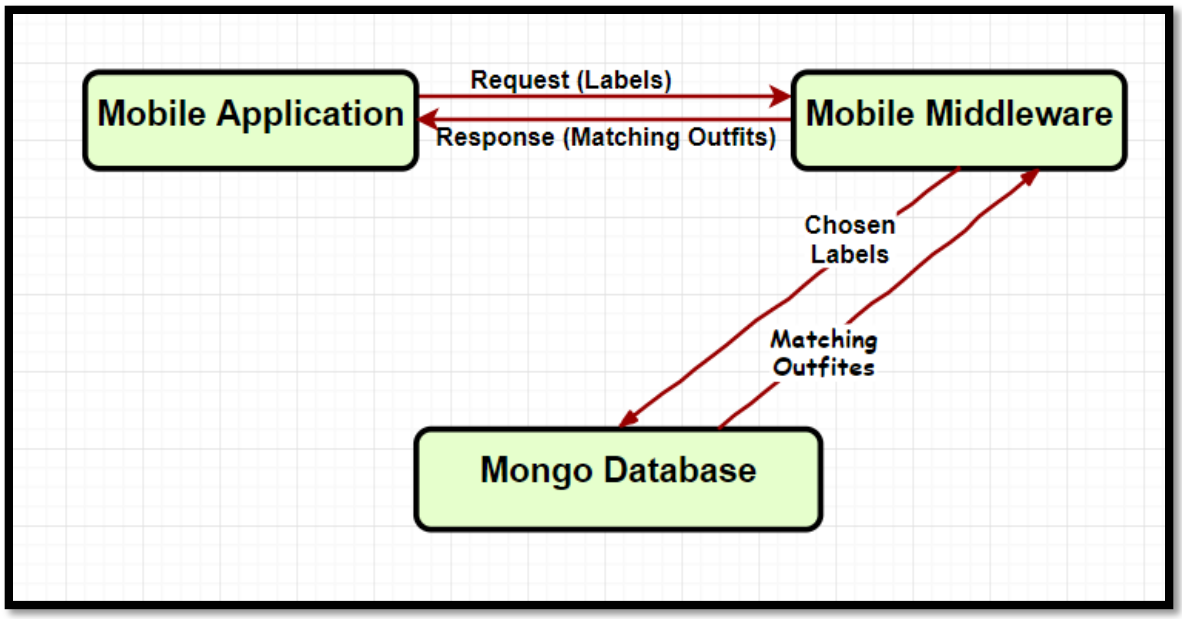


Figure 6-19 - Match an Outfit Scenario - Part1

Scenario steps are as follows:

1) The mobile applications sends a POST request to the mobile middleware in which the query image is embedded into it.
2) The mobile middleware extracts the image from the request and send it to the semantic segmentation module that finds the pieces and their categories from the image.
3) The semantic segmentation module sends the pieces segmented image to the color extraction module where it identify the color feature of the pieces.
4) The color extraction module sends the pieces categories along with the color feature of each of them to the mobile middleware.
5) The mobile middleware sends the colored pieces categories back to the mobile application embedded in the response.
6) The user chooses which colored piece categories he wants to find a matching outfit with it and the chosen label is sent as a request back to the mobile middleware.

7) The mobile middleware extracts the chosen label from the request and find all the matching rules for this chosen label from the database. Using the outfit matching rules, it retrieves all the matching outfits with this piece from the database.
8) The middleware embed the matching outfits in the response back to the mobile application.

Figures (6-20), (6-21), (6-22) show the scenario steps from the mobile application

1) The user picks an image from his camera roll

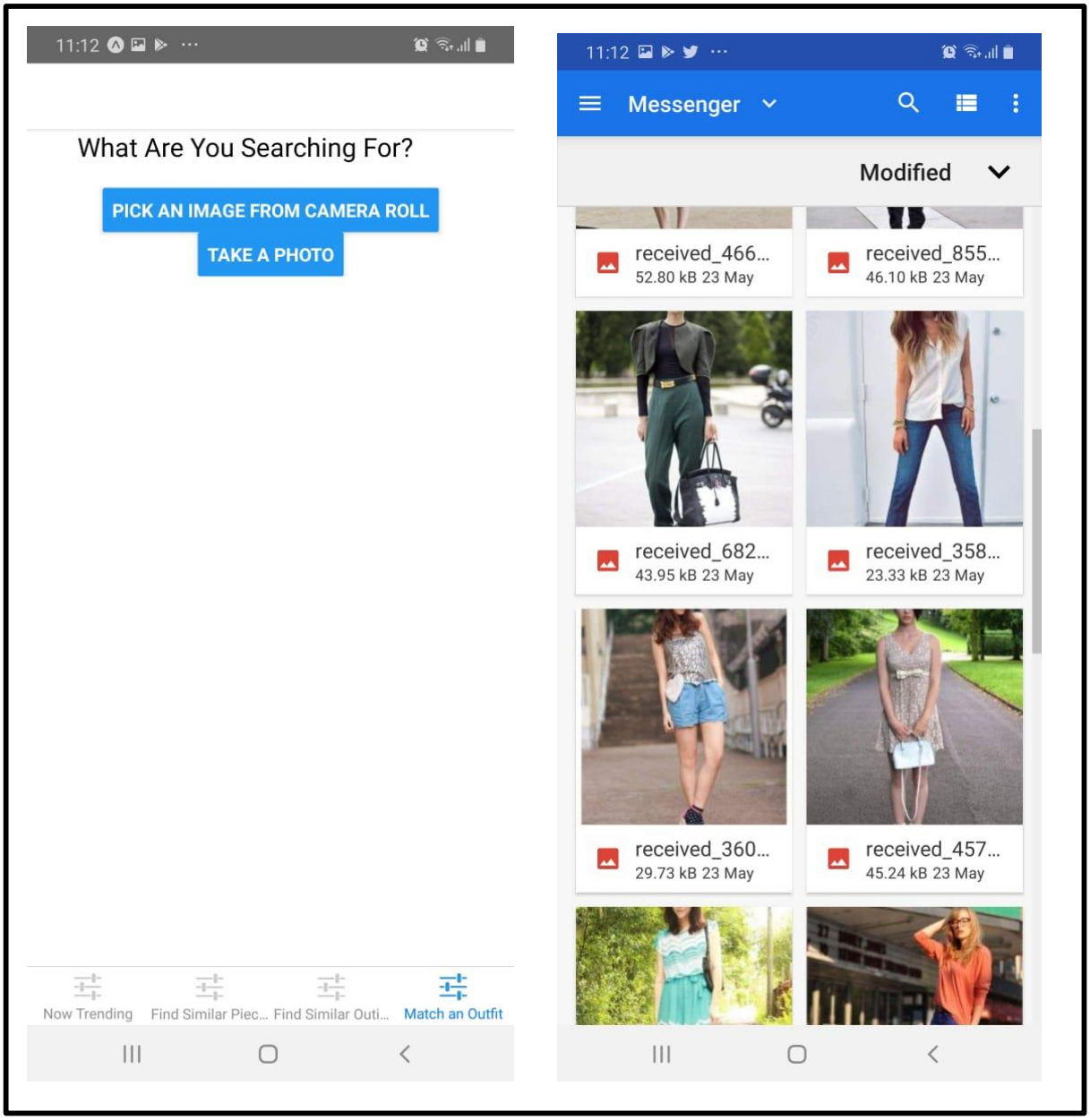


Figure 6-20 - User Choosing an Image

2) The colored pieces categories are returned to the user and he chooses to find a matching outfit with the aqua shorts

Figure 6-21 - Image Features Displayed

3) The matching outfits with aqua shorts are displayed to the user.

Figure 6-22 - Aqua Shorts Matching Outfits

6.6.2 Find Similar Outfits scenario

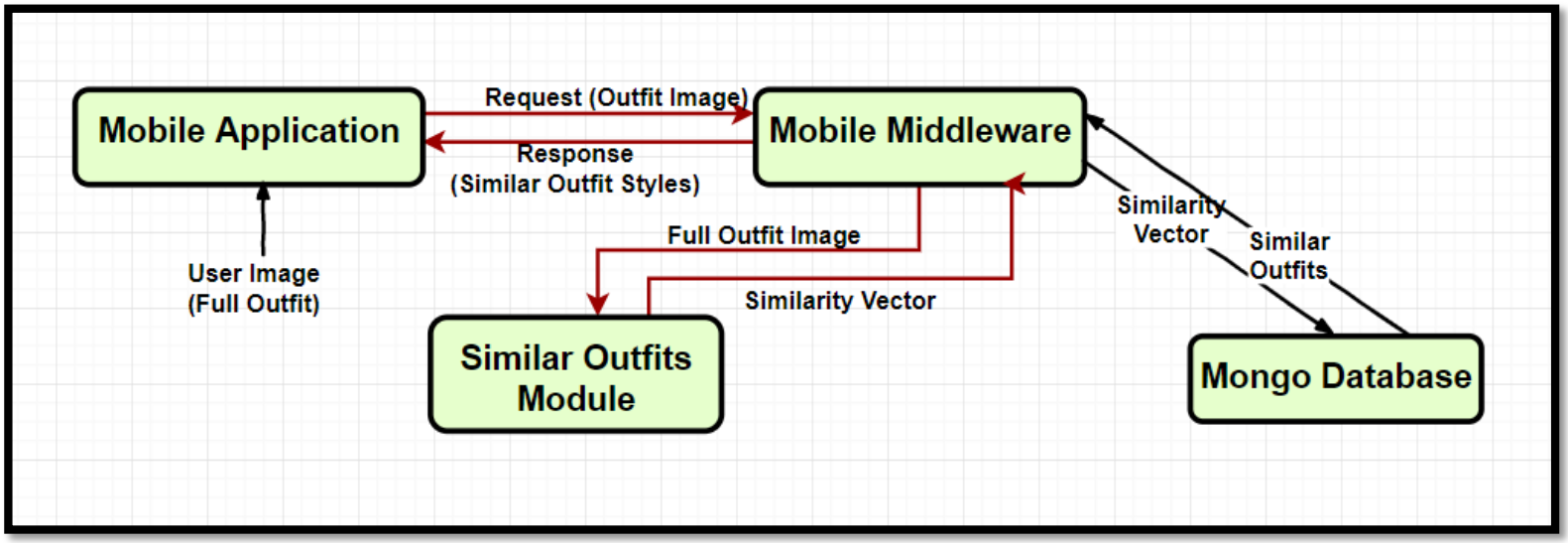


Figure 6-23 - Find Similar Outfits Scenario

Scenario steps are as follows:

1) The mobile applications sends a POST request to the mobile middleware in which the query image is embedded into it.
2) The mobile middleware extracts the image from the request and send it to the similar outfits' module that calculates the similarity vector of the image and sends it back to the mobile middleware.
3) The mobile middleware sends the similarity vector of the outfit to the database where it finds the most similar outfit styles to the query image by measuring the distance between the similarity vector of the query image and the similarity vector of the outfit styles and then the database sends to the mobile middleware the most similar outfit styles.
4) The mobile middleware sends back the most similar outfit styles to the mobile application in the response where they are displayed

Figures (6-24), (6-25), (6-26) show the scenario steps from the mobile application

1) The user picks an image from his camera roll

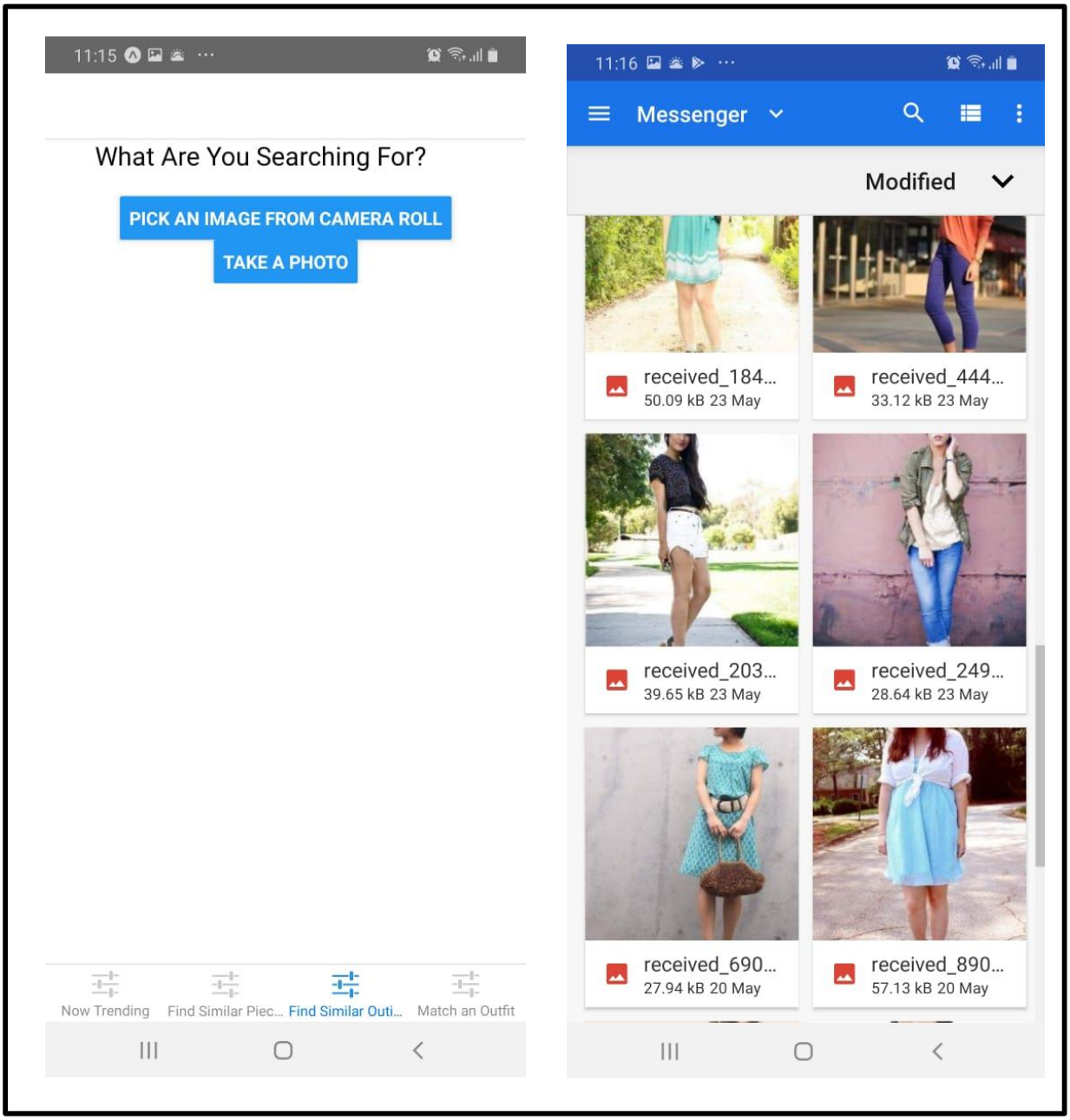


Figure 6-24 - User Choosing an Image

2) The query image is sent to the middleware to find similar styles to the style in it. The query image contains a short dress with a high waist belt on it.

Figure 6-25 - User Chosen Image

3) The application displays to the user similar outfit styles which also contain short dresses with a high waist belt

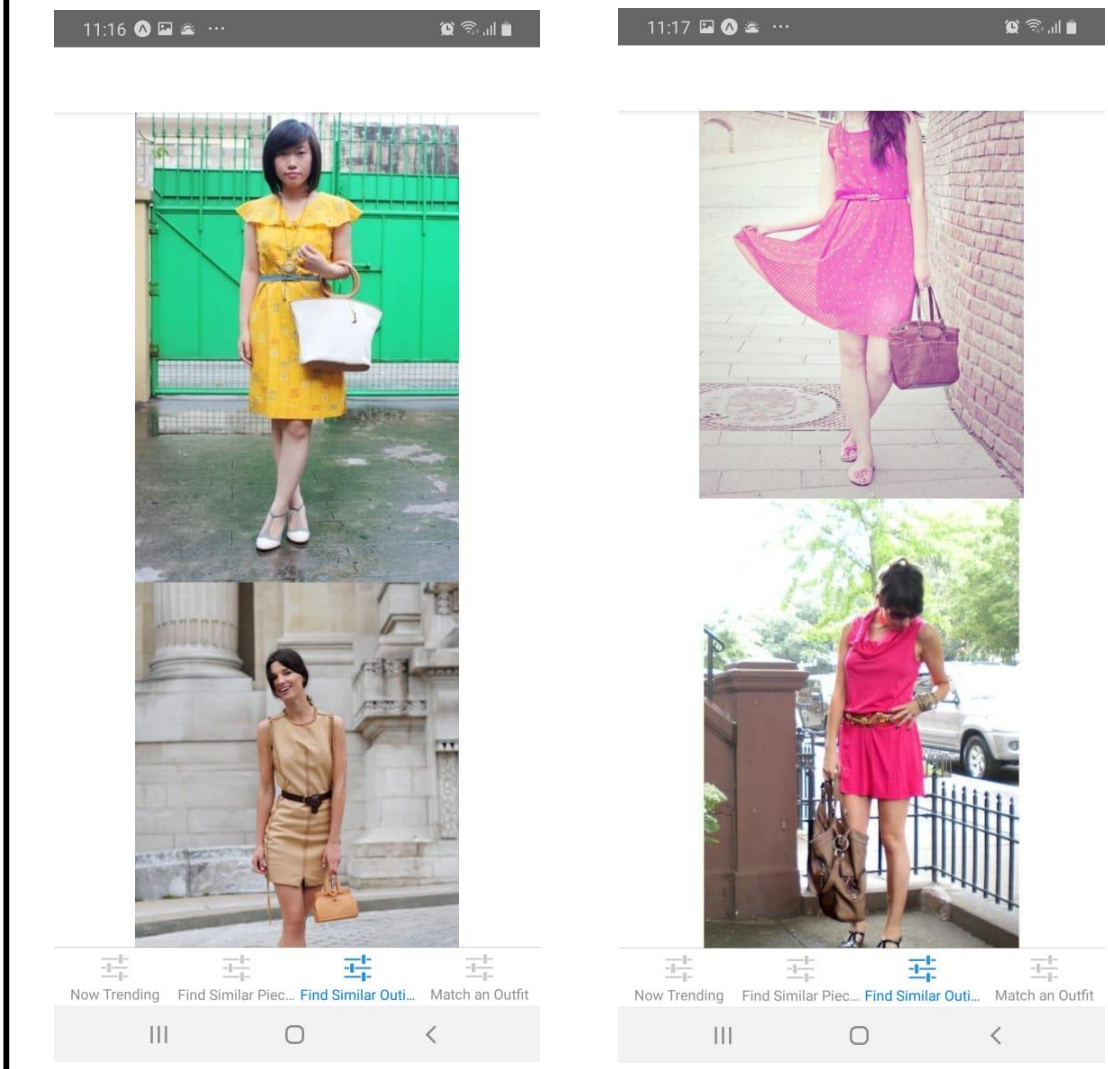


Figure 6-26 - Similar Styles Retrieved

6.6.3 Find Similar Pieces Scenario

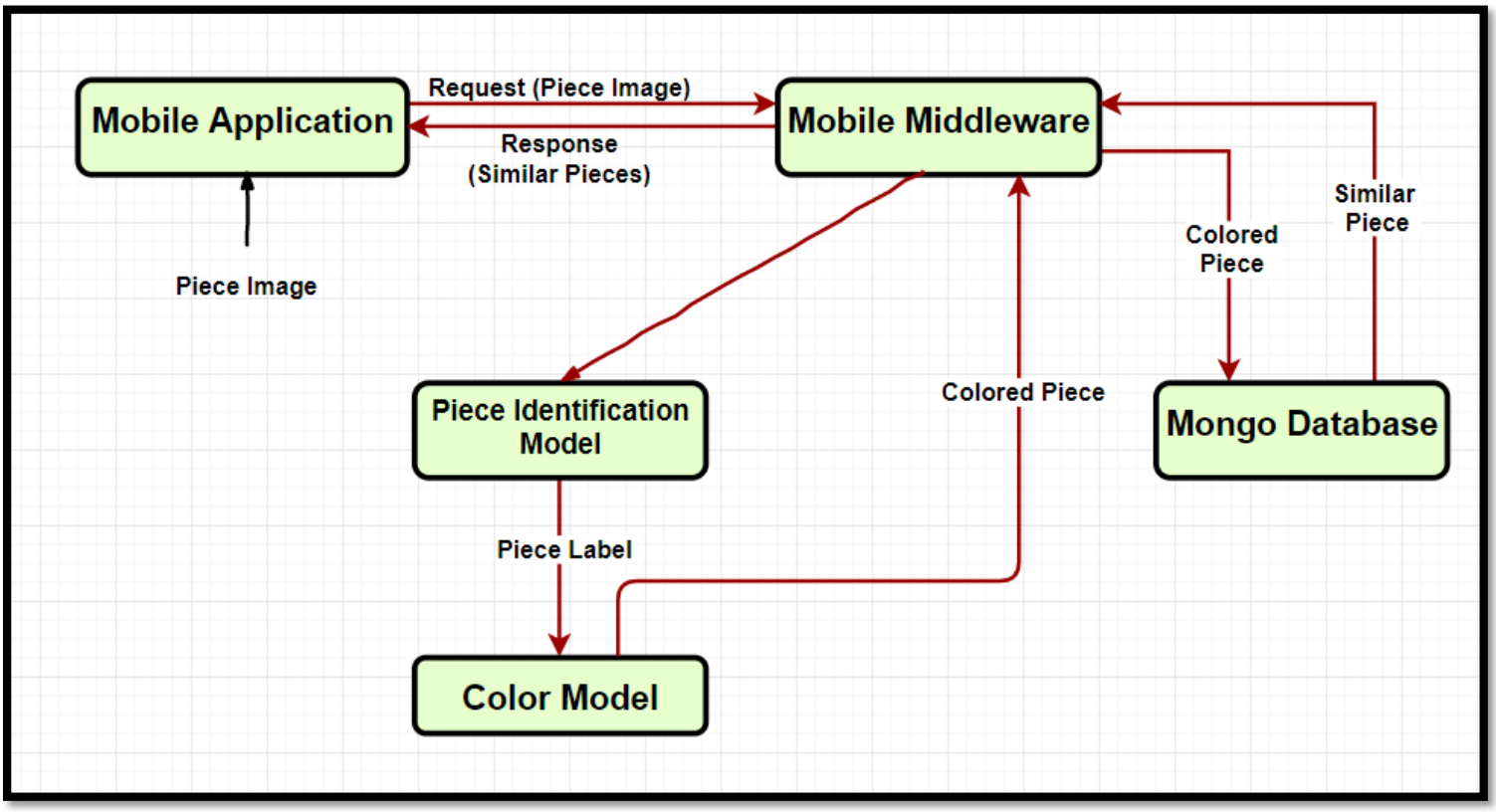


Figure 6-27 - Find Similar Pieces Scenario

Scenario steps are as follows:

1) The mobile applications sends a POST request to the mobile middleware in which the query image is embedded into it.
2) The mobile middleware extracts the image from the request and sends it to the piece identification module which identifies the piece label of the image
3) The pieces identification module sends the piece label to the color extraction module to find the color of the image
4) The colored piece label is sent back to the mobile middleware
5) The mobile middleware finds pieces similar to the one in the query image from the database
6) The mobile middleware sends back the similar pieces retrieved from the database to the mobile application using the response.

Figures (6-28), (6-29), (6-30) show the scenario steps from the mobile application

1) The user picks an image from his camera roll

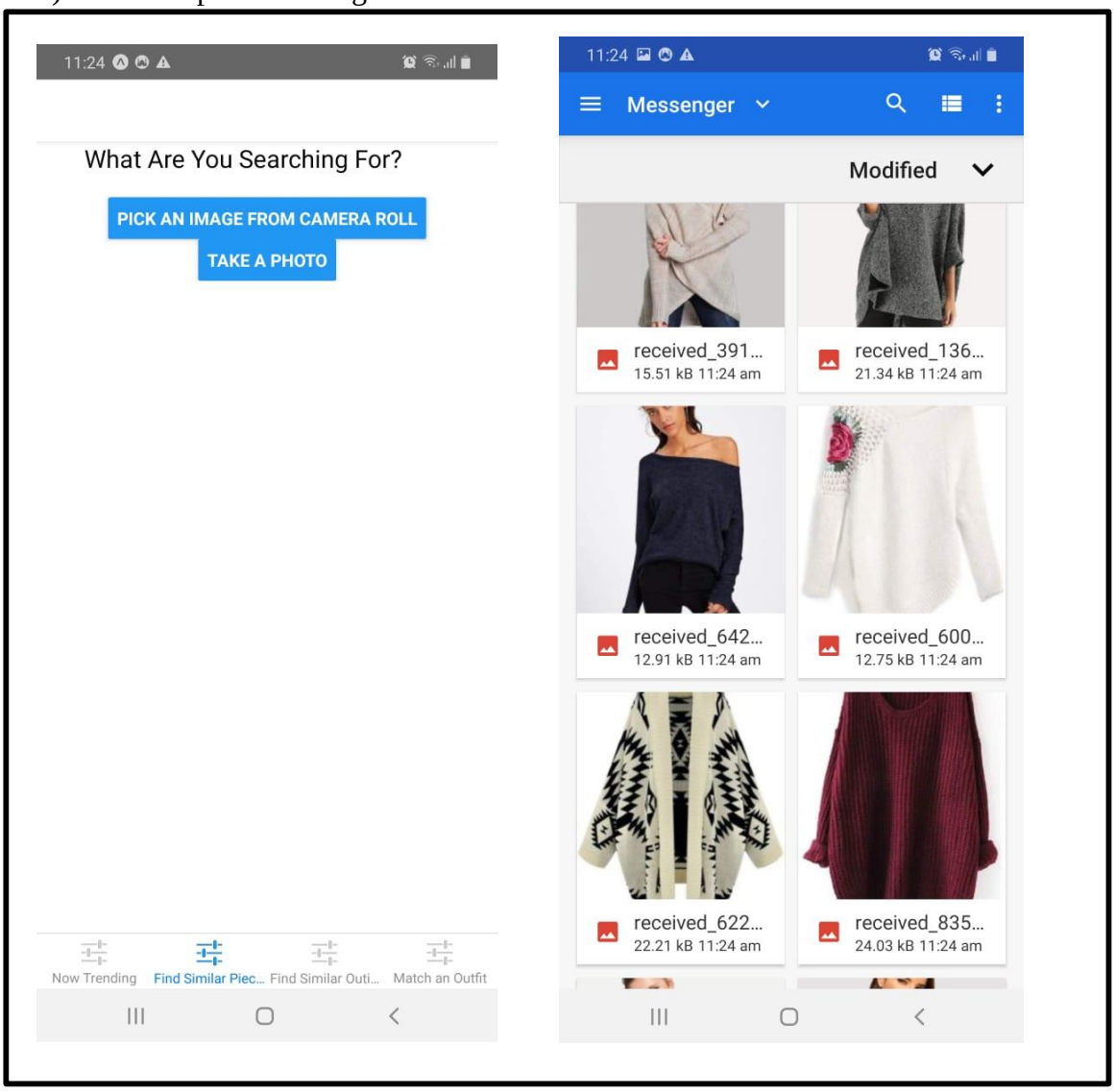


Figure 6-28 - User Choosing an Image

2) The identified piece category is returned to the user and he chooses to find similar pieces to it.

Figure 6-29 - Identified Piece Category

3) The application displays to the user similar pieces to the query image from the database

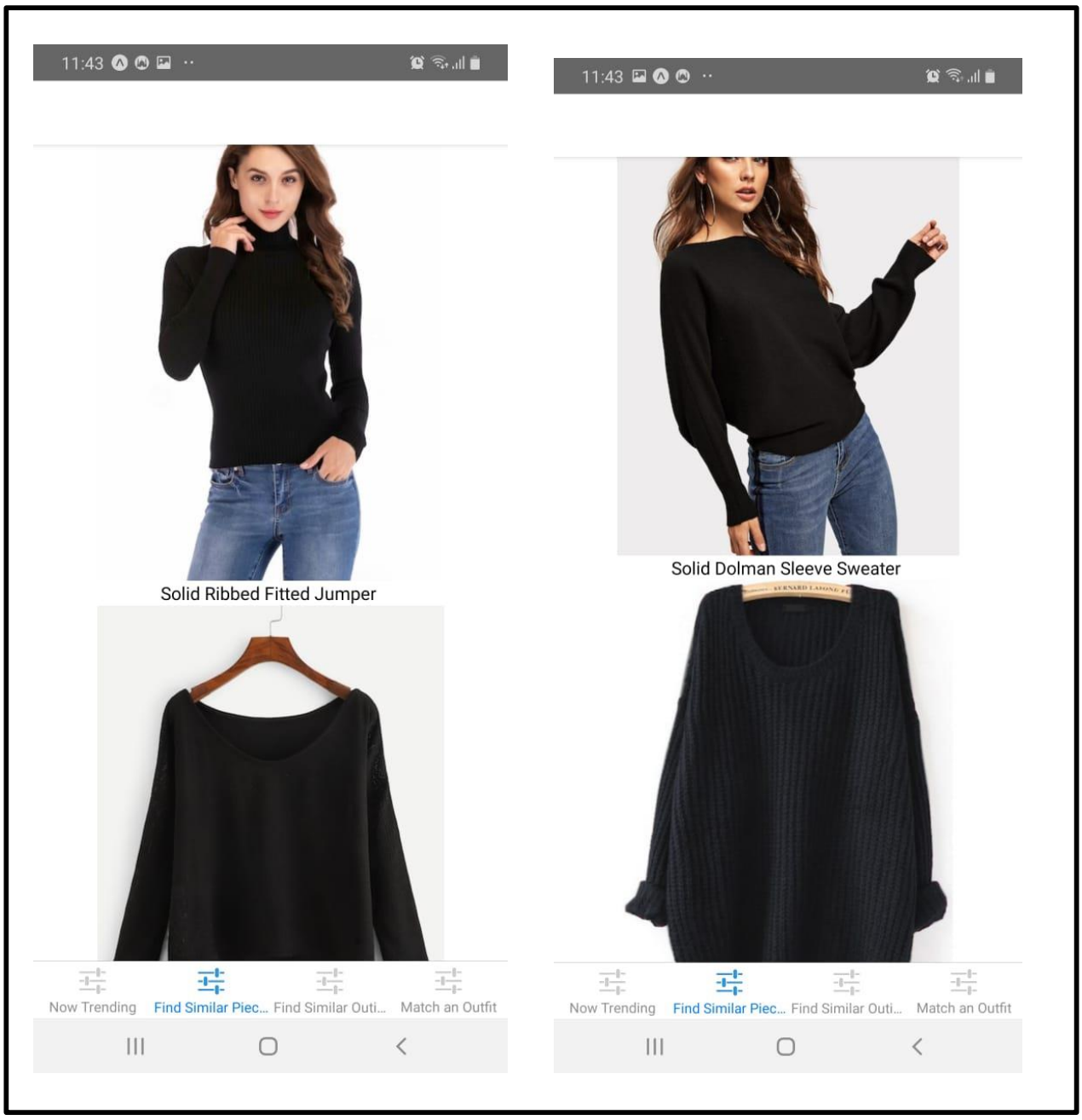


Figure 6-30 - Similar Pieces Retrieved

6.7 4.6.4 Now Trending Scenario

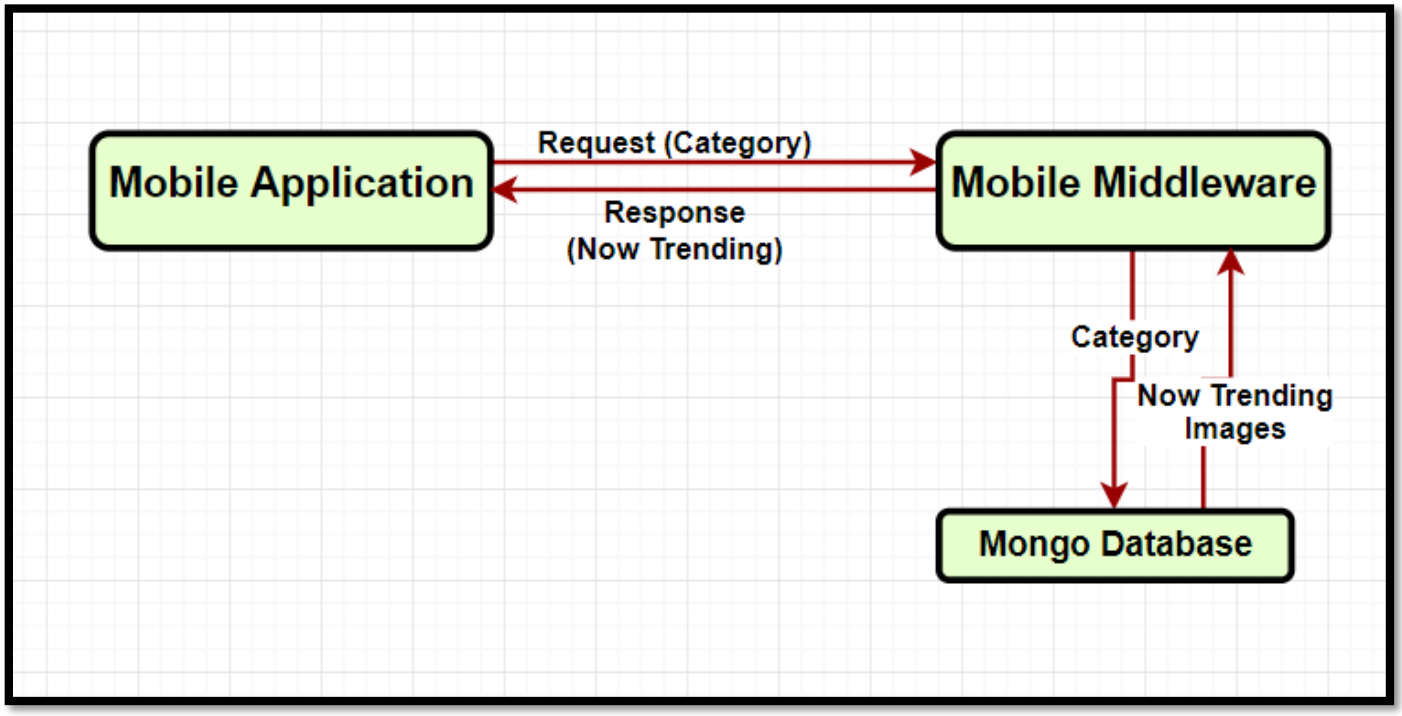


Figure 6-31 - Now Trending Scenario

Scenario steps are as follows:

1) The mobile applications sends a request to the mobile middleware with which category he wants to find the trending outfits in it.
2) The mobile middleware retrieves the trending outfits in the chosen category from the database.
3) The mobile middleware sends back the Now Trending outfits to the mobile application using the response.

Figures (6-32), (6-33) show the scenario steps from the mobile application

1) The user chooses the Now Trending category

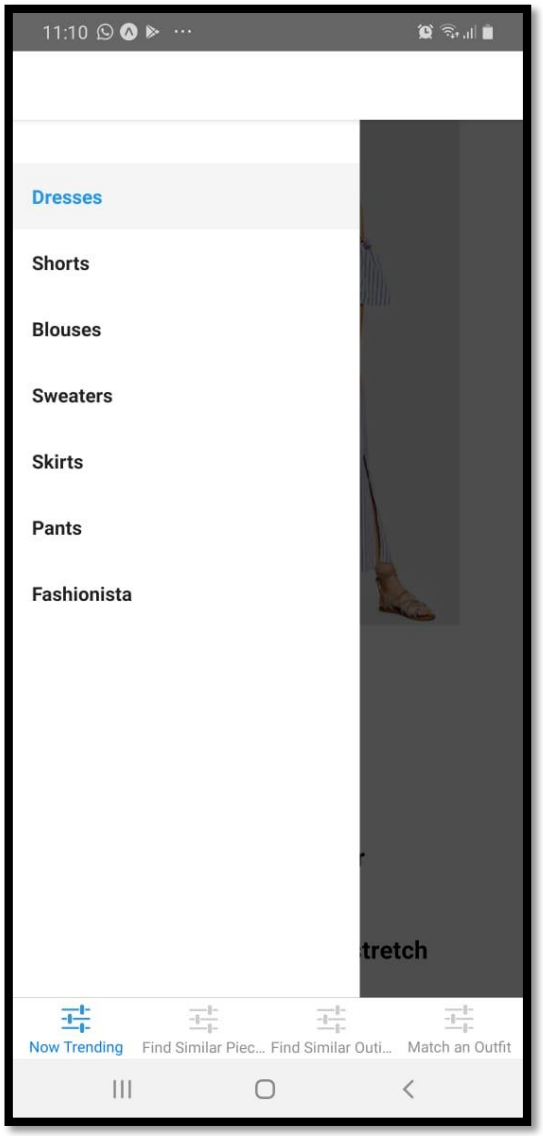


Figure 6-32 - Now Trending Categories

2) The Now Trending outfits in the dresses categories are displayed to the user with some information about each of them.

Figure 6-33 Now Trending Dresses

7 Conclusion & Future Work

7.1 Conclusion

Working on an innovative idea is an important part of applying all the learnt things in a journey of five years of engineering. We gathered different forms of sciences in our project in order to come up with a final product that can sum up our work and knowledge.

We used ‘Deep Learning’ in our models. ‘Fully Convolutional Networks’ (FCNs) are used for ‘Semantic Segmentation’ where the image is segmented into its pieces. ‘Semantic Compositional Networks’ are used for single piece identification where we classify the type of the clothes’ piece in the image. This is used when the user upload a piece of clothes on its own not wore on a model. ‘Semantic Compositional Networks’ are also used for similarity retrieval where the model identify clothing styles similar to the one in the query image.

Moreover, ‘Image Processing’ is used for color feature extraction where we identify the colors of each clothes’ piece in the image.

‘Big Data Analytics and Mining’ are also used for the Trending and Matching module where we gain insights about the trending colors and styles now and we find the outfits matching rules based on the current trends using ‘Association Rules Mining’.

For crawling Instagram and online shopping websites, we used ‘Information Extraction’ in order to get the images of fashion bloggers and the images of trendy outfits that are currently into market and extract the needed information from them.

Finally yet importantly, we made full stack development of a mobile application in order to integrate all the modules together and visualize all the results to the user. The application offers to the user different functionalities where: 1) the user can upload an image and the application match an outfit with the piece of clothes in the image. 2) The user can upload an image and the application finds for him all the pieces similar to the piece he is searching with in the online stores. 3) The user can ask the application to find for him similar outfit styles to the outfit in the

query image. 4) The user can retrieve the trending styles and colors now in all types of clothes: dresses, blouses, skirts, .etc.

To conclude, “My Shopping Pal” is a project that will make a difference in the field of shopping and it will be of a great value for most of its users. Our aim is to be unique and keep learning new things that will be beneficial to us in our future career life that will start soon.

7.2 Future Work

Enhancements will be made on the current model to include some future work that is of great value. The future work includes:

1) Use Gabor Feature Extractor along with the colors extractor in order to find both textures and colors in each image. This way we will have more specific results for each query image and will generate matching rules more specific than before.
2) Usage of the matching module implemented in shopping stores for Pieces Placement/Organization, such that pieces that match together would be placed together in stores.
3) Usage of the matching module obtained in online shopping such that when a person is about to buy a certain clothing piece, the website recommends to him other pieces to buy that matches with this piece (cross-sell).
4) Making contracts with the shopping stores so that when the user views an item that he likes, he can also buy it online from our application.

8 References

[1] Xiaodan Liang, Si Liu, Xiaohui Shen, Jianchao Yang, Luoqi Liu, Jian Dong, Liang Lin, and Shuicheng Yan, “Deep human parsingwithactivetemplateregression,” IEEEtransactionson pattern analysis and machine intelligence, vol. 37, no. 12, pp. 2402–2414, 2015.

[2] Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan, “Human parsing with contextualized convolutional neural network,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1386–1394.

[3] Tarasha Khurana, Kushagra Mahajan, Chetan Arora, and Atul Rai, “Exploiting Texture Cues for Clothing Parsing in Fashion Images,” in 25th IEEE International Conference on Image Processing (ICIP), 2018.

[4] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang, “Deepfashion: Powering robust clothes recognition and retrieval with rich annotations,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1096–1104.

[5] Junshi Huang, Rogerio Feris, Qiang Chen, and Shuicheng Yan, “Cross-domainimageretrievalwithadualattribute-aware ranking network,” in Computer Vision (ICCV), 2015 IEEE International Conference on. IEEE, 2015, pp. 1062–1070.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell, “Fully Convolutional Networks for Semantic Segmentation”, Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[7] Dennis Gabor, “Theory of communication. part 1: The analysis of information,” JIEE-Part III: Radio and Comm. Engg., vol. 93, no. 26, pp. 429–441, 1946.

[8] John G Daugman, “Two-dimensional spectral analysis of corticalreceptive fieldprofiles,” Vision research, vol. 20, no. 10, pp. 847–856, 1980.

[9] Timo Ojala, Matti Pietikainen, and Topi Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” TPAMI, vol. 24, no. 7, pp. 971–987, 2002.

Notes:

- Ezay el idea before the problem statement?! Ne3keshom?