APPAREL RECOMMENDATION WITH TRANSFER LEARNING AND LOCALITY SENSITIVE HASHING

A Thesis

Presented to

The Faculty of the Department of Computer Science

Sam Houston State University

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science

---

by

Kubra Gundogan

December 2022

APPAREL RECOMMENDATION WITH TRANSFER LEARNING AND LOCALITY SENSITIVE HASHING

by

Kubra Gundogan

APPROVED:

Hyuk Cho, PhD
Committee Director

Qingzhong Liu, PhD
Committee Member

Min Kyung An, PhD
Committee Member

ABM Islam, PhD
Committee Member

John B. Pascarella, PhD
Dean, College of Science and Engineering Technology

DEDICATION

To my parents, Fatma Gundogan & Hasan Gundogan, and my brothers, Bekir Burhan Gundogan & Alp Sinan Gundogan, for their support, love, and encouragement throughout my life and education. With their support and love, I have been able to overcome any challenge I have faced.

To Dr. Ken W. Smith and Janet Smith, who have been like a second family to me and have supported me.

ABSTRACT

Gundogan, Kubra, *Apparel recommendation with transfer learning and locality sensitive hashing.* Master of Science (Computing and Data Science), December 2022, Sam Houston State University, Huntsville, Texas.

The textile and apparel industries have now grown a lot and there is a variety of clothing that is constantly renewed or changed throughout the world. Given the abundance of selection options available, we developed a system that takes an image a user provides and then offers a recommendation which matches the user's query image.

This study developed a cloth recommendation system, which employs transfer learning with a pre-trained deep learning model (VGG16) followed by locality sensitive hashing with random projection. The dataset was originated by the H&M company and was exhibited in a competition via Kaggle. This dataset contains *105K* image data in total by addressing *130* different categories in five *(5)* main groups. Among a total of *7,000* of the Ladieswear group, occupying about *37.7%* in the dataset, a balanced dataset was obtained by splitting the *7,000* images into seven *(7)* clothing groups. These groups are labeled dress, trousers, sweater, blouse, skirt, t-shirt, and vest top.

Specifically, we extracted embedded features of the image using transfer learning and achieved a fast recommendation using locality sensitive hashing. We demonstrated the effectiveness of the proposed recommendation system by comparing the average cosine similarity of top 6 recommendations before and after locality sensitive hash. Furthermore, we qualitatively visualized the quality of the recommendation.

KEY WORDS: Apparel recommendation, Transfer learning, Feature embedding, Locality sensitive hashing, Random projection

ACKNOWLEDGEMENTS

My sincere thanks go out to my advisor, Dr. Hyuk Cho, for his guidance, and great support throughout my master's program.

I am grateful to Dr. Min Kyung An for her mentorship and support, as well as her advice and positivity.

I am grateful to Dr. ABM Islam for encouraging me at the beginning of my master's degree program.

I would like to thank Dr. Qingzhong Liu for his support and valuable advice.

Lastly, I would like to thank the Computer Science Department at Sam Houston State University for all the opportunities they have provided to me.

TABLE OF CONTENTS

Page

DEDICATION iii

ABSTRACT iv

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vi

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER I: INTRODUCTION 1

Problem Context 1

CHAPTER II: RELATED RESEARCH 2

CHAPTER III: TRANSFER LEARNING 5

VGG16 6

ResNet50 6

InceptionV3 7

CHAPTER IV: RECOMMENDATION SYSTEM 12

Locality-Sensitive Hashing (LSH) 13

CHAPTER V: PROPOSED APPROACH 15

Data Description 15

Hardware & Software Tools 19

Our Approach 20

CHAPTER VI: EXPERIMENTAL STUDY 24

Discussion 24

Training 24

LSH 27

Results 27

CHAPTER VII: SUMMARY & FUTURE WORK 46

Summary 46

Future Work 47

REFERENCES 49

VITA 52

LIST OF TABLES

Table **Page**

1 Hardware 19

2 Software 19

LIST OF FIGURES

Figure		Page
1	VGG16 Architecture	6
2	ResNet50 Architecture	7
3	Inceptionv3 Architecture	8
4	Original Image and Embedded Feature	10
5	t-SNE Visualization for Feature Embedding	11
6	Random Projection Matrix	14
7	Dataset Information for Major Group	16
8	Dataset Information for Different Categories	17
9	Dataset Information for Different Subgroups	18
10	Schematic Diagram of Proposed Recommendation Model	21
11	Images with the Same Signature in LSH	23
12	Performances of VGG16, InceptionV3, and ResNet50	25
13	Model Layers	26
14	Hash Table with Signatures and List of Signatures	28
15	Visualization of Similar Images in Dress (Training)	29
16	Visualization of Similar Images in Vest Top (Training)	30
17	Visualization of Similar Images in Sweater (Training)	31
18	Visualization of Similar Images in Blouse 1 (Training)	32
19	Visualization of Similar Images in Blouse 2 (Training)	33
20	Visualization of Similar Images in Vest Top (Test)	35
21	Visualization of Similar Images in Skirts (Test)	36

22 Visualization of Similar Images in Trousers (Test) 37

23 Visualization of Similar Images in Blouse (Test) 38

24 Visualization of Similar Images in Sweater (Test) 39

25 Visualization of Similar Images in Dress (Test) 40

26 Visualization of Similar Images in T-Shirt (Test) 41

27 Training and Test Data for Embedded Feature 42

28 Training and Test Data Followed by LSH 43

29 Cosine Similarity Between LSH Signature and Top-6 by Signature 44

30 Cosine within Similarity [-1, 1] 45

31 Cosine Distance 45

CHAPTER I

Introduction

Problem Context

In recent years, the e-commerce industry has experienced rapid growth, especially in the fashion industry. Although many studies have reviewed recommendation systems in general, they may not be able to keep up with these rapid changes. It is becoming increasingly apparent that recommendation systems lack adequate functionality as this growth increases (Deldjoo, 2022). Since there is a great deal of demand in different layers of the fashion industry's value chain, it is important both for researchers and trade developers. One of the problems with recommendation systems studies is the lack of data, and this poses a problem for both researchers and industries.

This thesis focuses on closing the gaps in this rapidly developing industry by presenting a study. Despite a somewhat limited dataset, we provide recommendation systems that can maximize user’s expectation.

While designing this apparel recommendation study, we aim to increase the accuracy rates by employing the recently developed algorithms. In this way, we will increase the satisfaction of the users and the platforms selling clothes at the right rate. Additionally, we will provide easy-to-search products to online customers. It will be possible for users to find the products quickly and easily they are looking for. To quantify the success of the search results, the list of similarity values (e.g., cosine between embedded features) between the query and recommended images will be calculated. By making a list of both specific products and similar products that users may be interested in, we gave them more options.

CHAPTER II

Related Research

Due to the huge increase in online shopping, clothing companies that sell on websites need new methods to increase consumer satisfaction. Many articles and projects have proposed numerous methods that can partially address the issue. The following approaches focus on identifying target outfits and recommending alternatives, in chronological order.

Huang et al. (2013) used support vector regression (SVR) with a user interaction training phase to monitor user preferences based on users' feedback of previous results. The proposed system recommended features such as color and pattern without asking the user, depending on the shopping they have done in the past.

Lao et al. (2015) proposed a recommendation system that handled four tasks. The first is to classify the clothing type very well, the second is to classify the clothing quality, the third is to perceive the nearest neighboring clothes, and finally to perceive the clothing object. They revealed the results with the detection, which they pre-trained using the R-CNN model and determined their properties. But unfortunately, the accuracy rate is as low as *50%* for clothing style classification and *75%* for clothing feature classification.

Sha et al. (2016) developed an approach to analyze images for clothing recommendations that extract multi-features from their content, where a color matrix model was proposed to distinguish clothing with split joints. Fabric pattern attributes were represented by uniform local binary pattern (ULBP) features. Features were extracted to describe collar and sleeve attributes. A classifier was then trained to classify clothing fabric patterns and split joints. A variety of experiments was conducted based on every attribute and their combinations, and the results were satisfactory.

Wen et al. (2018) constructed a knowledge graph of user, knowledge graph of clothing, and knowledge graph of context, utilizing the Apriori algorithm to capture the intrinsic correlations between clothing attributes and context attributes. After that, the Top-N algorithm combined with the established knowledge graph to generate the recommendation results directly based on the user's requirements.

Feng et al. (2018) developed an object detection system based on You Only Look Once (YOLO) v2. As in a similar research paper before, only trousers, skirts, coats, T-shirts, and bags are discussed by including certain categories. The experiment has *84%* average precision and *73%* average recall rates, stating that image detection takes place in *56*ms.

Master's thesis by Dai (2021) used a piece-based human detection algorithm with PASCAL (Pattern Analysis, Statistical Modeling, and Computational Learning) visual object classes, so the algorithm detects the clothes on people. At the same time, a two-stage Web Application and an IOS application were developed using k-Nearest Neighbors (KNN) for a recommendation. However, its performance details and application usage were unavailable.

Lee et al. (2021) proposed fashion apparel detection method using YOLO, where only detection according to certain categories of apparel is provided. They created a Two-Phase detection model, first to detect fashion clothing images with complex backgrounds and second to use the image enhancement model, which achieved the model detection more accurately. After dividing target categories into the five categories that include jackets, tops, pants, skirts, and bags, they developed a target-based detection model. Therefore, only the specified categories can be detected.

CHAPTER III

Transfer Learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Transfer Learning can be defined as the reuse of existing models to solve a new challenge or problem. It is not used as a separate machine learning algorithm, but rather to train models and assist in the realization of a new task. The new task can be described as separating objects into a certain file category and using the desired categories (Brownlee, 2019).

Traditional machine learning is computationally expensive and requires huge amounts of data to achieve high performance. In addition, in these algorithms, data information is independent of the past, so it is difficult to integrate it according to the features of the tasks to be used. Unlike traditional learning algorithms, transfer learning enables the creation of models obtained from a sufficiently large and general dataset. With these models, the learned features are quickly used for other tasks without having to start from scratch. Reuse and transfer of knowledge learned in areas related to transfer learning can be realized. Transfer learning is a method in which a model made for one task is intended to be used in a later model so that it can be reused with a specific starting point for a second task. It is aimed to achieve faster or better performances by using the modeling created with transfer learning in subsequent tasks.

Computer vision is expressed using transfer learning pre-trained models. These pre-trained models train datasets to solve similar problems to be solved and model them as layers for next uses. The most common of these models for image classification are listed as follows:

VGG16

VGG stands for Visual Geometry Group at Oxford University, which was introduced at the ILSVRC conference in 2014. It is one of the most popular pre-trained models for image classification. As shown in Figure 1 (Keras, 2017), VGG16 has *16* layers in total. Five pooling layers are included in this model, which has a total of *13* convolutional layers and three *(3)* dense layers. The last three *(3)* layers of VGG16 are fully connected. There are five *(5)* sets of convolution layers followed by a MaxPool in the overall structure. There is a difference in the set of five convolution layers that includes more cascading convolution layers. VGG16 uses an input size of *224*x*224*.

Figure 1

VGG16 Architecture

Note. The VGG16 has 16 layers in total and the VGG19 model has 19 layers in total. The general structure is formed within 5 sets of convolutional layers (Keras, 2017).

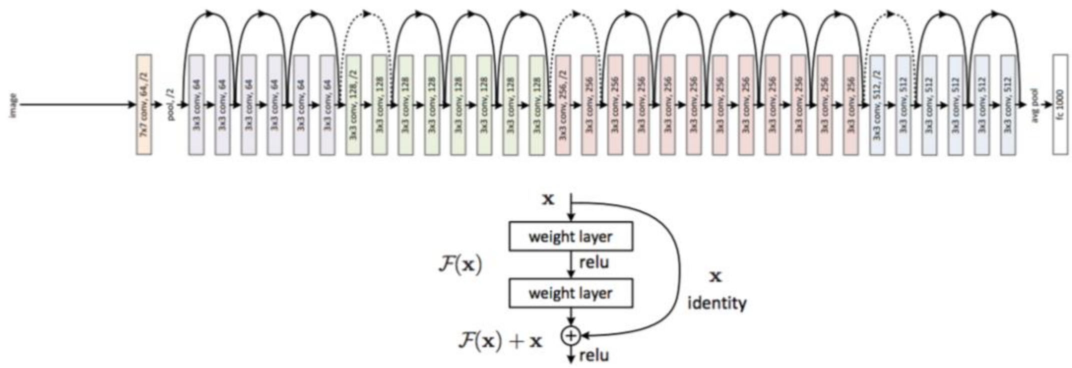
ResNet50

ResNet50 was introduced by Cornell University at the CVPR conference in 2015 (Keras 2015). As shown in Figure 2, the ResNet50 convolutional neural network has *50* layers deep (*48* convolutional layers, one MaxPool layer, and one average pool layer).

Pre-trained networks are capable of classifying images into *1,000* object categories. A *224*x*224* image input size is used for the network.

Figure 2

ResNet50 Architecture



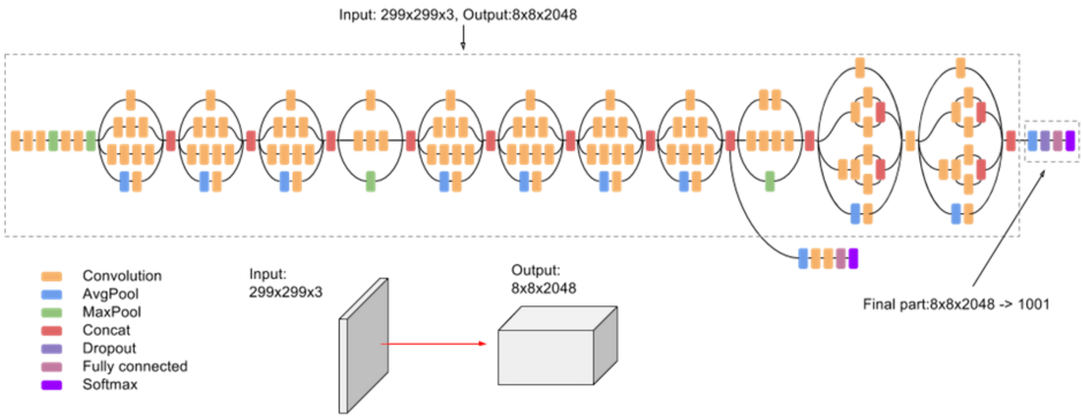
Note. ResNet50 image (JananiSbabu, 2015).

InceptionV3

InceptionV3 was introduced by Cornell University at the CVPR conference in 2016. Optionally loaded ImageNet weights can be used to enhance the Keras image classification model. As shown in Figure 3, InceptionV3 is a pre-trained convolutional neural network with a depth of *48* layers. This pre-trained mesh can sort images into *1,000* object categories. A *299*x*299* image input size is used for the network. In the first part of the model, general attributes are extracted from the input images, and in the second part, these attributes are used to classify the images.

Figure 3

Inceptionv3 Architecture



Note. InceptionV3 (Brain, 2017).

The basic idea of convolutional layers is to extract the most general, low-level features that can be applied between images, which can be patterns or border frames, and allow subsequent layers to recognize specific features in an image.

The basic idea in pooling layers works on each feature map separately to create a new set of the same number of pooled feature maps after the convolutional layer. It is smaller in size than the convolutional layer and reduces the size by 2x on almost any feature map.

Dense layers are deeply linked to previous layers. Each layer is connected to the other by neural networks and the dense layers are the most used layer. Since the results obtained from each neuron of the previous layers go to layers, the results of each neuron are included in smaller sizes with their most specific features. Therefore, the use of a dense layer is an important layer in terms of efficiency in areas where it will be used.

Image Embedding

Figure 4 shows an actual image of a Blouse class taken from the training dataset. It would be computationally expensive to compute similarities between images in original dimension. Therefore, instead of computing similarity in original space, it is more desirable to do it in reduced dimension. Particularly, feature embedding through well-known pre-trained CNNs (e.g., VGG16) has been used to capture image features in much reduced dimensional space, which is actually using pre-trained network as a feature extractor. By doing so, image features stored in a smaller size, such as *1x512*, still capturing important characteristic in original image, which allows them to be processed more quickly in reduced space. Embedded features from pre-trained deep learning model can be further reduced each dimension into two- or three-dimensional map for the visualization purpose. For example, Figure 5 illustrates 2D t-distributed Stochastic Neighbor Embedding (t-SNE) of embedded features (e.g., *1x512*) of images in the seven *(7)* ladieswear groups mentioned before. Specifically, one image in original dimension in Figure 4 (top) is *1522x1750* pixels, which is embedded into *1x512* pixels through VGG16 pre-trained feature extractor.

Figure 4

Original Image and Embedded Feature

(a)

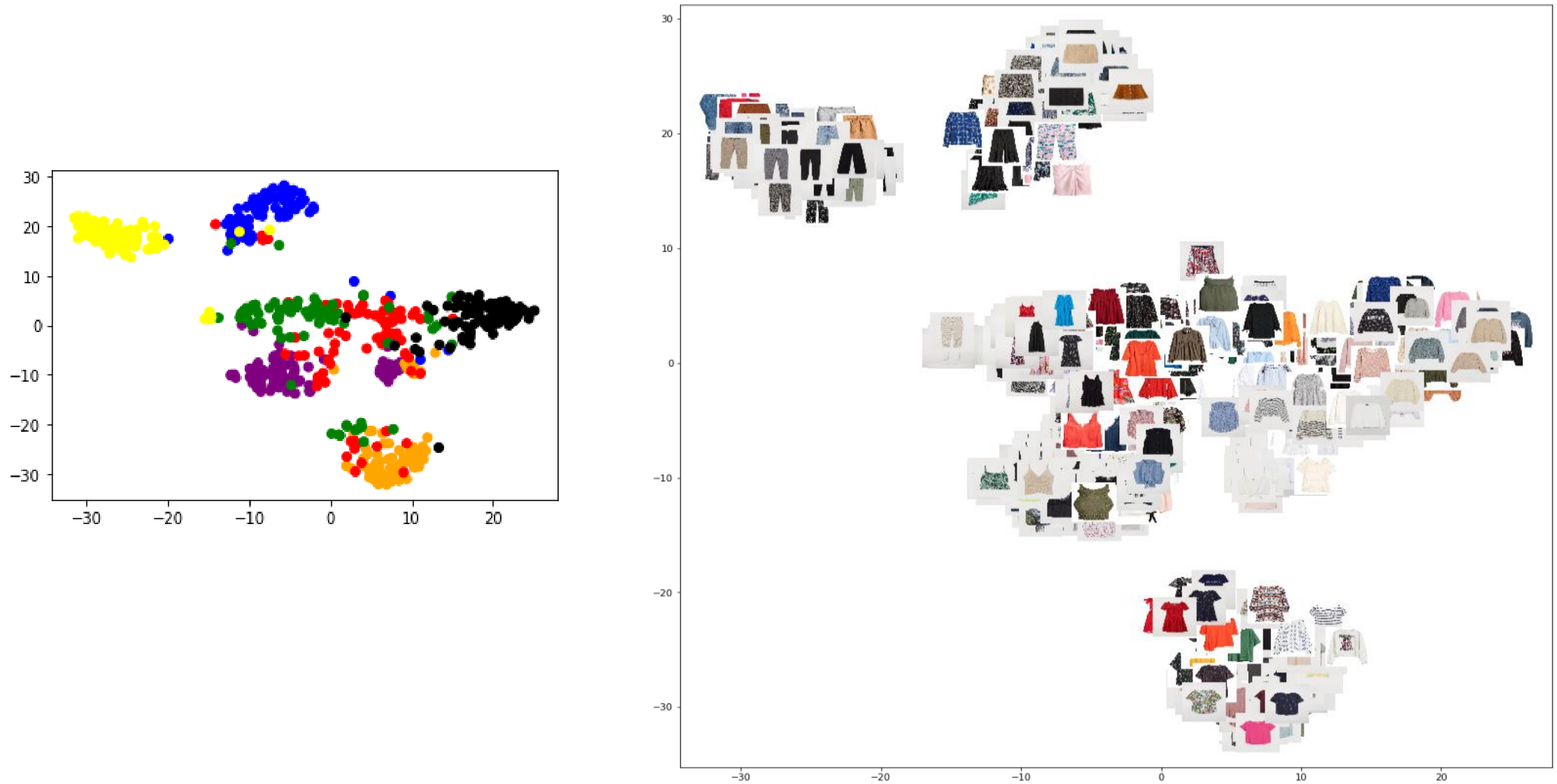
(b)

```
array([0.        , 0.26556313, 0.4007607 , 0.        , 0.        ,
       0.20494388, 0.34223843, 0.        , 0.2566481 , 0.6251735 ,
       0.39625946, 1.010883  , 0.47703704, 0.44093576, 0.        ,
       0.49360597, 0.10068506, 0.25343043, 0.5380185 , 0.40362164,
       0.        , 0.36461294, 0.11112449, 0.09034483, 0.        ,
       0.47911772, 0.61251837, 0.2635044 , 0.        , 0.4476055 ,
       0.16386886, 0.50694853, 0.13382745, 0.        , 0.32493573,
       0.        , 0.14790931, 0.        , 0.        , 0.31346706,
       0.23352994, 0.83969563, 0.        , 0.27018353, 0.31560022,
       0.24314828, 0.        , 0.        , 0.        , 0.        ,
       0.61707073, 0.42639577, 0.26000097, 0.6753903 , 0.46612534,
       0.47622257, 0.8753373 , 0.46646965, 0.        , 0.30475613,
       0.12353287, 0.        , 0.04667654, 0.        , 0.7154007 ,
       0.        , 0.        , 0.        , 0.        , 0.274435  ,
       0.        , 0.3691245 , 0.41327152, 0.2913917 , 0.31095487,
       0.5176617 , 0.08230603, 0.49421728, 0.2268299 , 0.        ,
       0.3669654 , 0.06157685, 0.        , 0.        , 0.53836995,
       0.5674747 , 0.        , 0.27896544, 0.10452886, 0.5903275 ,
       0.        , 0.5845432 , 0.4292739 , 0.20139423, 0.        ,
       0.        , 0.5300281 , 0.41629174, 0.        , 0.        ,
       0.        , 0.        , 0.626887  , 1.467924  , 0.        ,
       0.        , 0.02795857, 0.1823161 , 0.28673667, 0.35965872,
       0.48255223, 0.40727976, 0.        , 0.        , 0.        ,
       0.        , 0.25235114, 0.        , 0.        , 0.00329466,
       0.5587579 , 0.65823066, 0.42414263, 0.21832736, 0.08794534,
       0.71697766, 0.33258927, 0.979454  , 0.20508802, 0.0397873 ,
       0.52836496, 0.        , 0.        , 0.        , 0.4998151 ,
       1.013838  , 0.75547343, 0.        , 0.24548376, 0.        ,
       0.5681555 , 0.4560763 , 0.        , 0.        , 0.5164587 ,
       0.        , 0.        , 0.16162673, 0.3159766 , 0.2666575 ,
```

Note. The image above is from the Blouse class in Training. The original image (a), and the embedded feature (b). Dimension sizes for original image are *1522x1750* pixels. Dimension for embedded image is *1x512.*

Figure 5

t-SNE Visualization for Feature Embedding



Note. The left figure shows the t-SNE 2D projection of extracted features from VGG16, each of which is an image in one of the seven (7) ladieswear group. Each colored dot denotes one of the seven *(7)* groups of ladieswear. The right figure shows the actual image that correspond to the left figure.

CHAPTER IV

Recommendation System

A recommendation system evaluates product features based on data collected from a user to identify products that customers may prefer. Therefore, after evaluating it first evaluates a relationship between the user and product. It identified what products can be recommended to the user, based upon the relationship. Usual recommendation systems return a recommended list of images from the images already stored in a database, upon getting user's input (i.e., query image). Therefore, the overall recommendation performance can be affected by the quality of database and the computation of similarity search.

In this study, we proposed an apparel recommendation system that consist of the following specific steps:

1. One pre-trained CNN model (such as VGG16, ResNet50, or InceptionV3) is trained with training data.
2. Embedded feature vectors are obtained from the second to the last fully connected layer from the model created by training.
3. Binary signatures are generated by passing embedded features to locality sensitive hashing by random projection.
4. Query image is converted to a vector of embedded features by step 2 and the signature of embedded feature vector of query image can be obtained by step 3.
5. Based on the signature of query image at steps 4 is used to access LSH location where similar images are stored, and the sorted list of similar images are obtained.

Locality-Sensitive Hashing (LSH)

When two points in the feature space are close together, there is a high probability that they will have the same hash which is a reduced representation of the data. A primary difference between LSH and conventional hashing is that LSH tries to maximize collisions between similar points instead of aiming to avoid them. LSH ignores slight distortions so the main content of the input can be easily identified despite slight perturbations to the input, as in hashing. Due to the collisions between hash values, similar items are more likely to have similar hashes.

Random Projection

In general, random projections are based on the idea that given a high-dimensional vector of data, similar vectors should be grouped together into the same bucket. A random projection reduces the dimensionality of high-dimensional data by transforming it into low-dimensional features. In low-dimensional spaces, it is computationally less expensive because it approximates relations with cosine similarity (Santhosh, 2018).

In Figure 6, n observations and d features are represented as columns and rows, respectively, in a high-dimensional data matrix D. By projecting the matrix onto a k-dimensional space d, we can obtain a lower-dimensional representation P. This lower dimensional representation of the matrix can be calculated mathematically as a lower dimensional representation by projecting it onto a k-dimensional space d. Random vectors are column elements of the random projection matrix R, whose elements are independent of gaussian distributions which is zero mean, and unit variance (Santhosh, 2018).

Figure 6

Random Projection Matrix

Note. Explanation of the Johnson-Lindenstrauss lemma.

CHAPTER V

Proposed Approach

Data Description

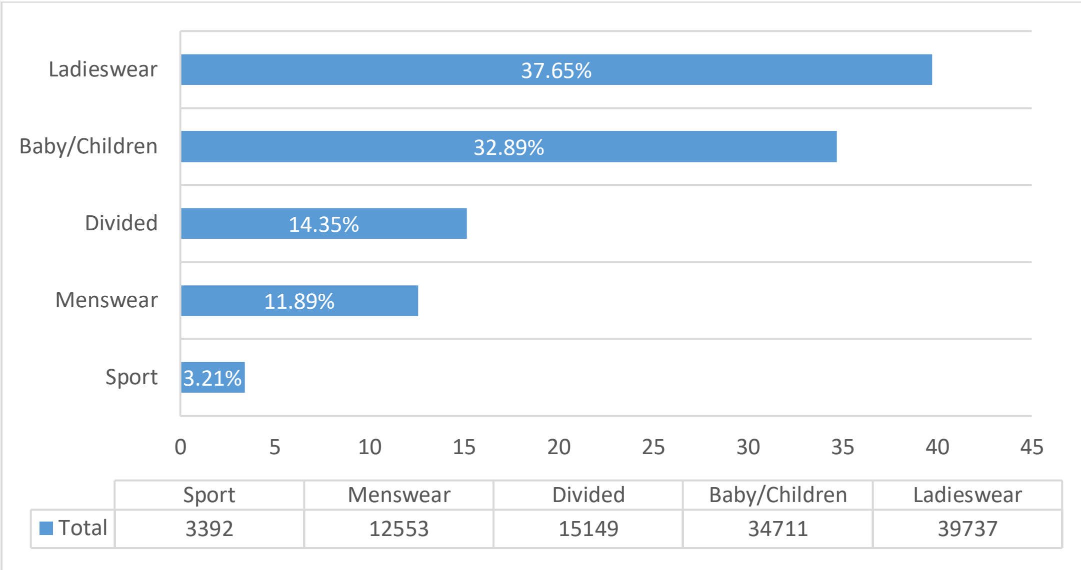
The entire dataset used in this study is publicly available on Kaggle under the H&M personalized fashion recommendations competition and provided by H&M Hennes & Mauritz AB (H&M Group). H&M is a multinational group based in Sweden and its focus is clothing. It is a long-standing company established for fashion and design services for women, men, youth, and children in a fashionable, sustainable way and suitable for everyone.

H&M Group is a business that owns around *4,850* stores around the world and sells on *53* online platforms. The dataset shared via Kaggle contains an image list with the definitions of more than *105K* products that it currently sells, as well as age, location, and activity information along with the IDs of its customers.

As shown in Figure 7, original H&M dataset has a total of *105K* images over five (*5)* main groups, which include Ladieswear *(37.65%)*, Baby/Children (*32.89%)*, Divided *(14.35%)*, Menswear (*11.89%)*, and Sport (*3.21%)*.

Figure 7

Dataset Information for Major Group

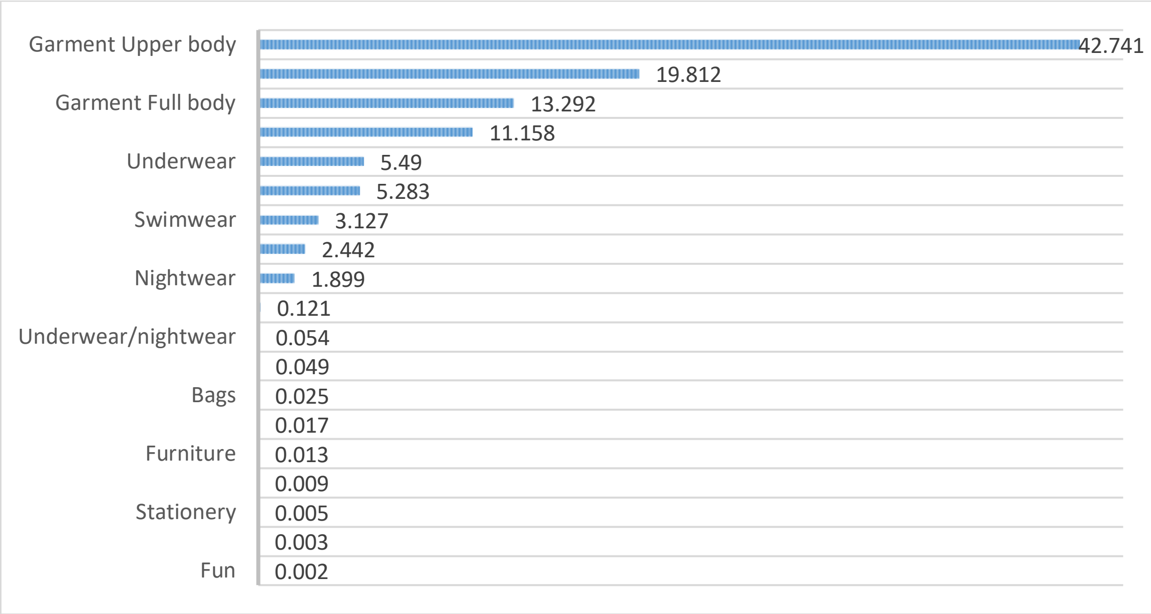


Note. The original H&M dataset content consists of five *(5)* main groups, which are divided into Ladieswear, Baby/Children, Divided, Menswear by Sport.

At the same time, the clothes and accessories in the dataset are divided into *19* different categories in total according to their usage patterns or different parts of the body. Frequent labels are Garment Upper Body, Garment Lower Body, Garment Full Body, Accessories, Underwear, Shoes, Swimwear, Socks & Tights, and Nightwear, as shown in Figure 8.

Figure 8

Dataset Information for Different Categories

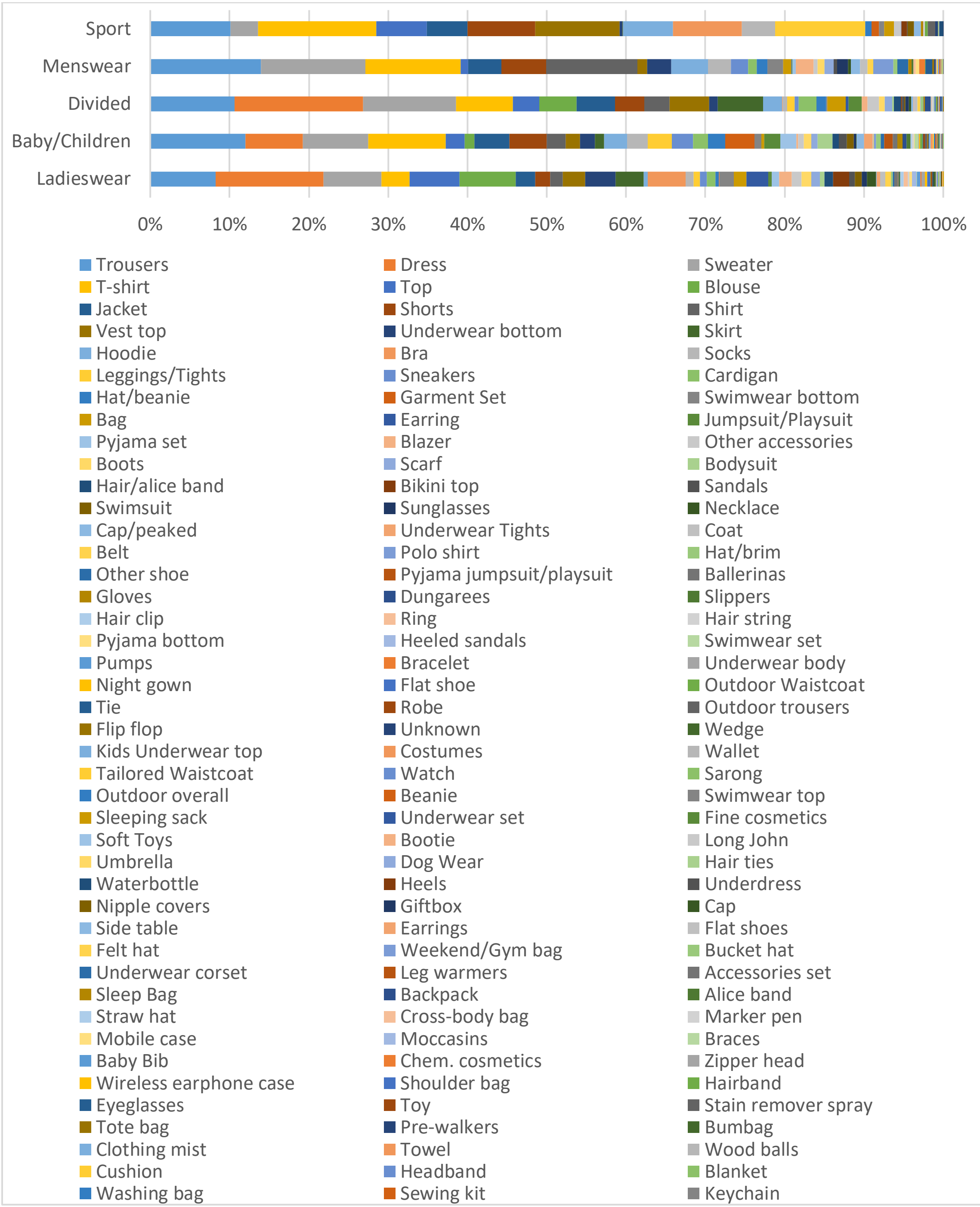


Note. The frequencies of *19* different categories in original H&M dataset.

The distribution of all *130* subgroups in the dataset in five *(5)* main classes is shown in Figure 9. As we can see from the colors determined for each subclass, the most crowded clothing groups for Ladieswear are trousers, dresses, sweaters, and tops. There are too many subclasses in each main group, even though not all the *130* subclasses are contained in all five *(5)* main groups. Among *130* subgroups, ladieswear includes *98* subgroups, baby/children have *94* subgroups, divided contain *74* subgroups, menswear has *65* subgroups, and sport has *33* subgroups.

Figure 9

Dataset Information for Different Subgroups



Note. The distribution of 130 different subgroups over five *(5)* major groups.

Hardware & Software Tools

Tables 1 and 2 summaries hardware and software used in this study, respectively. All the hardware is available in Google Colab, which is a computing platform for Google users, writing Python code via the browser, and especially for machine learning, data analysis, and education. With Google Colab, users can take advantage of the processing resources on the virtual and cloud platforms without using their own local hardware. Google Colab has three *(3)* different subscriptions, which are standard Google Colab, Colab Pro, and Colab Pro+ (Google).

Table 1

Hardware

GPU	GPU Memory	CPU	System Memory
NVIDIA K80	12GB	AMD EPYC™ 7742 @ 3.4GHz	12GB
NVIDIA T4	16GB	AMD EPYC™ 7742 @ 3.4GHz	26.75GB
NVIDIA P100	16GB	2 x vCPU	32GB
NVIDIA V100	32GB	2 x vCPU	52GB

Table 2

Software

Software	Version
Operating System	Jupyter notebook environment Google Colab
Python	>=3.8
Keras - neural networks	2.9.0
CNN Model	VGG16

In this study, all the three Google Colab subscriptions were used. The benefits of these subscriptions are to use different GPUs, CPU Memory, CPU, and System Memory. The priorities in the use of this hardware are determined according to the subscription.

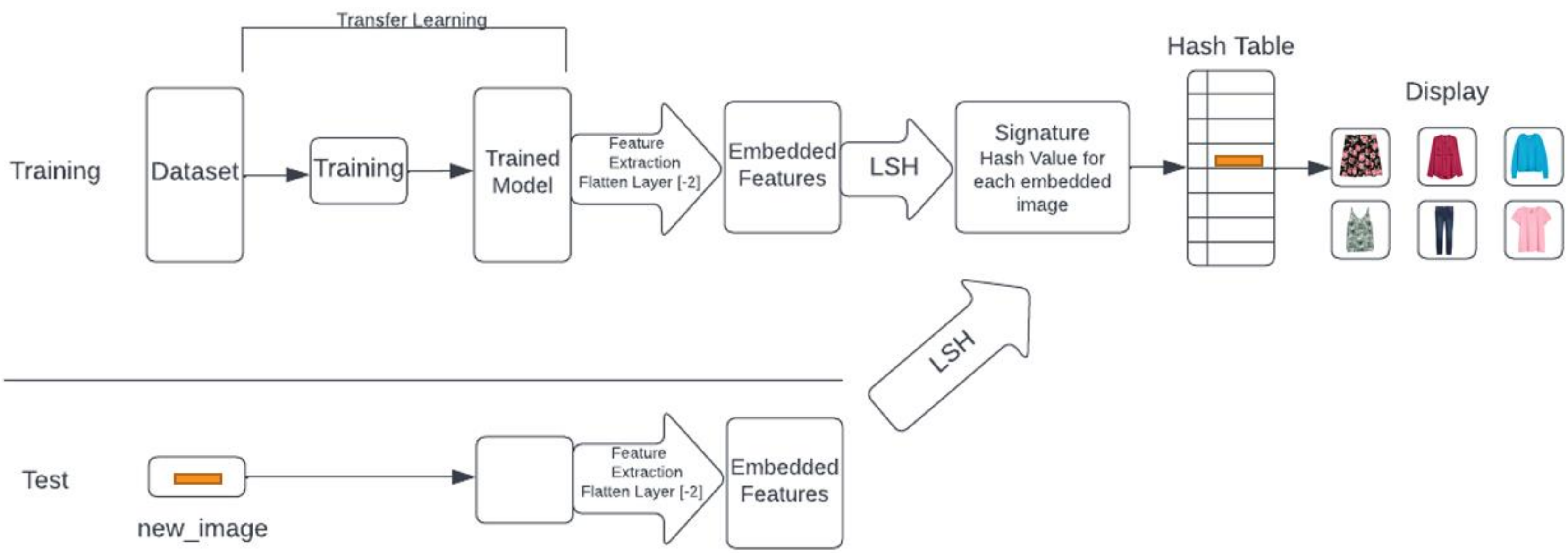
Four different CPUs and their features, NVIDIA K80, T4, P100, and V100, were used in this study. Additionally, Keras is one of the leading high-level neural network APIs, and most CNN models (such as VGG, ResNet, Inception, etc.) are implemented in Keras.

Our Approach

Figure 10 summarize that schematic diagram of proposed recommendation model's all the paths we created and followed. Our dataset was first divided into two different groups, training and test. The training process begins when we create the trained models using Transfer Learning. Initially, three pre-trained CNN models including VGG16, ResNet50, and InceptionV3 were tested; however, VGG16 model produced the best results for feature extractor in terms of both training and test accuracy. A hash table with LSH was used to group embedded features into the same buckets based on signatures.

Figure 10

Schematic Diagram of Proposed Recommendation Model



Note. The schematic diagram of the recommendation model proposed in this study.

The LSH implementation creates a hash table of all possible buckets, each containing similar items. Bitwise hash values represent each split as a sequence of 1s and 0s. In general, according to the random projection matrix, bitwise hash tables are generated as follows:

- The dimension of each feature vector should be d, and the bitwise hash value should be k.
- For each random vector, the dot product of the vector and the observation should be calculated. Dot product results that are positive should be assigned a *1*, otherwise a *0*.
- Use k dot products to combine all the bit values.
- Repeat the above two steps for all observations to compute hash values.
- By grouping observations based on their hash value, create an LSH table.

As shown in the Figure 11, the Hash Table keeps the clothing pictures that are close to each other in the same list under each signature. With three random vectors, there are *8* different signature groups can be generated, each of which keeps similar bit values representing the images in the training together. A larger random vector (for example, four random vectors have *16* different signatures total) can generate a greater number of signatures.

Figure 11

Images with the Same Signature in LSH

```
[0]=> [7, 8, 9, 11, 12, 13, 15, 16, 17, 18, 20, 33, 34, 35, 36, 37, 38, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
[1]=> [10, 19, 22, 70, 80, 140, 206, 249, 259, 289, 321, 518, 519, 520, 532, 561, 562, 563, 591, 604, 659, 696, 697, 832
[2]=> [0, 6, 14, 24, 30, 61, 62, 97, 108, 110, 123, 131, 143, 152, 160, 166, 170, 173, 178, 180, 181, 186, 201, 204, 217
[3]=> [43, 189, 313, 1752, 1869, 1877, 1929, 1933, 1938, 2036]
[4]=> [76, 99, 126, 167, 172, 200, 203, 205, 211, 215, 233, 235, 264, 272, 284, 303, 315, 320, 322, 344, 346, 414, 433,
[5]=> [2, 3, 4, 5, 31, 58, 59, 68, 69, 79, 82, 90, 107, 122, 138, 139, 192, 193, 212, 216, 245, 246, 247, 248, 283, 290,
[6]=> [67, 120, 134, 137, 144, 145, 171, 174, 175, 183, 266, 267, 274, 282, 311, 348, 945, 1101, 1113, 1159, 1170, 1191,
[7]=> [1, 21, 23, 25, 26, 27, 28, 29, 32, 39, 40, 41, 42, 44, 57, 105, 159, 188, 230, 241, 242, 243, 261, 262, 263, 341,
```

Note. There are eight different signature groups for all training data. The lists of all eight groups have similar cosine similarity values.

In the Test process of our approach, for every single test image, feature extraction and random projection were applied to assign its signature in LSH. Similarly, the signature of each test image is assigned to access similar images with a same signature, from which a sorted list of similar images can be retrieved by computing cosine similarity between the test image (i.e., search query) and every image only in that LSH location, which therefore results in fast retrieval.

CHAPTER VI

Experimental Study

Discussion

People perceive the contents of the images they see differently, and this is different for each person. The perceptions made by different people may be different, for example, some people pay attention to the patterns, colors, and brightness of the images they see (Chakraborty, 2021). Using the LSH Table, we aimed to find similarity measures based on color, shape, and pattern features between the clothing images, which are eventually represented in the embedded vector form.

Apparel recommendation deals with finding similar images for a particular query image. This undoubtedly involves an underlying process for classifying images. It is a system that uses features of a query image to guide matching suggestions based on distinctive and discriminating features of training images.

Training

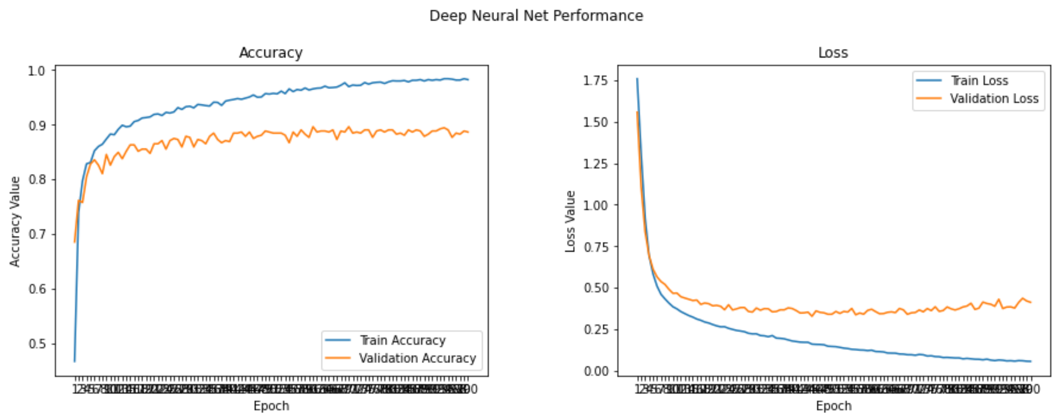
As pilot models, we used three well-known pre-trained models including InceptionV3, ResNet50, and VGG16, and checked how they performed on our dataset, and then chose the model that gave the best training and test performance.

Figure 12 shows training accuracy results for VGG16, InceptionV3, and ResNet50. Our first step was to find the CNN model with the best performance, to learn the training and test accuracy for all three models. The training accuracy results over 100 epochs were obtained by VGG16, InceptionV3, and ResNet50. Each model achieved following test accuracy: VGG16 had *90.56%;* InceptionV3 had *76.7%*; and ResNet50 had *77.5%.* We chose VGG16 as it performed best with our dataset.

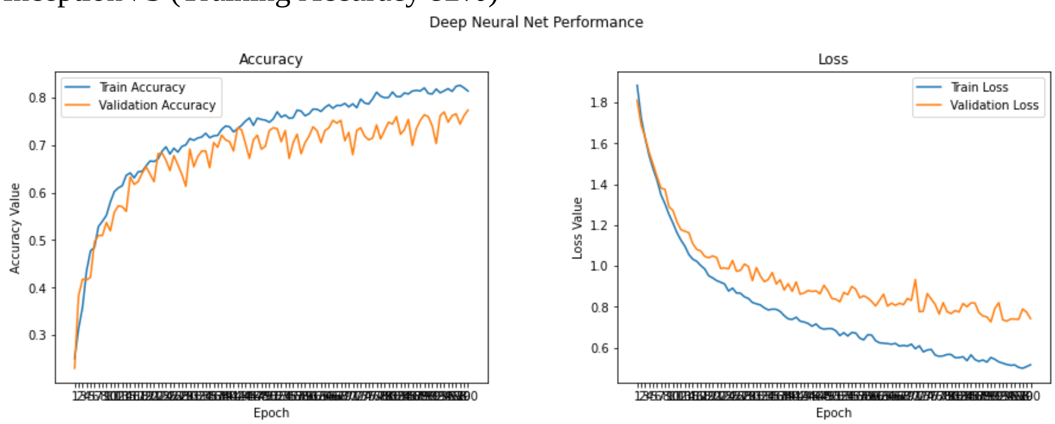
Figure 12

Performances of VGG16, InceptionV3, and ResNet50

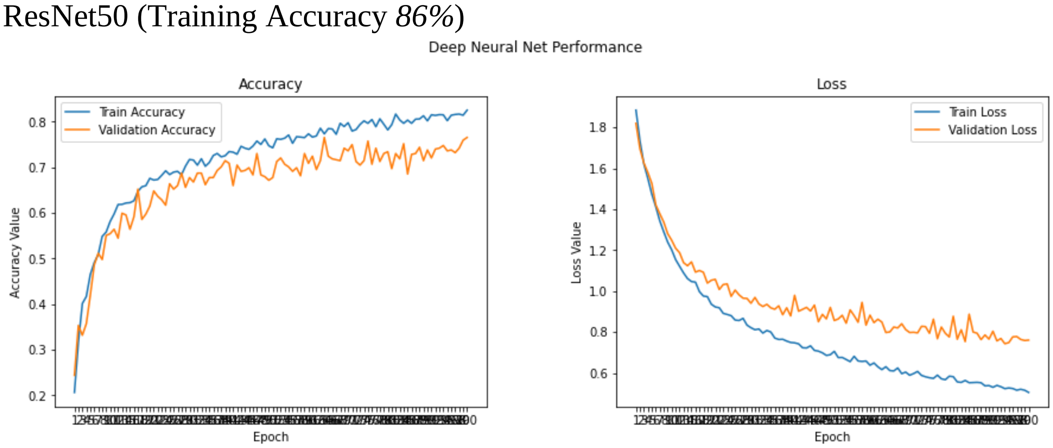
(a) VGG16 (Training Accuracy *98%*)



(b) InceptionV3 (Training Accuracy *82%*)



(c) ResNet50 (Training Accuracy *86%*)



Note. Training accuracies for three models

Specifically, we used VGG16 model as feature extractor. Meanwhile, it continued with a reduced convolutional layer with a flatten layer, to obtain an embedded feature vector for each image. This flatten layer was defined as the second to the last, which is a fully connected layer before the final classification layer (top layer). This layer was made to keep the feature embedding vector, as shown in Figure 13.

Figure 13

Model Layers

(a)

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 512)	262656
dense_1 (Dense)	(None, 512)	262656
dense_2 (Dense)	(None, 7)	3591

Total params: 15,243,591
Trainable params: 528,903
Non-trainable params: 14,714,688

(b)

Model: "model_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 512)	262656
dense_1 (Dense)	(None, 512)	262656

Total params: 15,240,000
Trainable params: 525,312
Non-trainable params: 14,714,688

Note. Figure 2 (a) is a summary of layers in VGG16, and Figure 2 (b) is a summary of layers in VGG16 for feature extraction.

LSH

As the new embedding model created still has a large scale to use LSH, it is necessary to help reduce the dimensionality of the embedded vectors by leaving the information content of each image the same. Therefore, a hash function has been created to approximately maintain the distances between group points in a certain plane. The hash function reduced the dimensionality of embedded vectors. With LSH, a signature for a vector of embedded features calculated so that the images with the same signature value can be likely to be in the same LSH table location.

We implemented random projection based LSH to create a table of all possible signatures that are keys for LSH table. Bitwise hash values of 1s and 0s are used to describe each partition (example: *001*, *010*, *011*). There is a greater likelihood of similarity between two observations with the same bitwise hashes than between those with different bitwise hashes when this notation is used.

Results

To see the results of our system, we used *7,000* images out of *39,737* images in Ladieswear, which has seven *(7)* classes including blouse, dress, sweater, skirt, trousers, t-shirt, and vest top. Ladieswear is the only gender used within the main groups. Each class has *1,000* images and they divided into three datasets: 70% training, *15%* validation, and *15%* test in this study.

The images that were trained with the Transfer Learning model VGG16 achieved a test accuracy result of *90.56%*.

This Figure 14 shows a LSH table that has been populated. In this table, we have three *(3)* random vectors that help generate eight *(8)* signatures, which other keys for

LSH location that hold the lists of all training images that are similar to each other. This recommendation system will rank similar *top 6* images in the same signature list when the query image is searched. Due to the fact that eight (8) different similar classes could not be created, not every signature in the Figure is complete.

Figure 14

Hash Table with Signatures and List of Signatures

```
[0]=> [52, 219, 1060, 1261]
[1]=> [1, 6, 35, 45, 53, 72, 94, 114, 124, 130, 153, 186, 218, 227, 239, 240, 279, 321, 329, 3
[2]=> []
[3]=> [5, 150, 190, 205, 238, 270, 426]
[4]=> [4, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 40, 44, 57, 59, 62, 67, 68, 70, 73, 76,
[5]=> [0, 2, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 33, 34, 36, 37, 38, 43, 46, 4
[6]=> [27, 39, 41, 42, 264, 383, 432, 476, 480, 516, 517, 584, 599, 639, 654, 696, 697, 1054,
[7]=> [14, 93, 136, 149, 291, 350, 351, 352, 357, 359, 361, 375, 387, 389, 397, 400, 404, 408,
```

Figure 15 shows the sorted list of the top 6 similar images with the highest cosine similarity, when image index *387* is searched. The first returned image is identical to query image, thus is cosine similarity is *1*, and the five (5) remaining images are ordered by cosine similarity. Figures 16 – 19 qualitatively demonstrate, when query images from vest top, sweater, and blouses classes are selected.

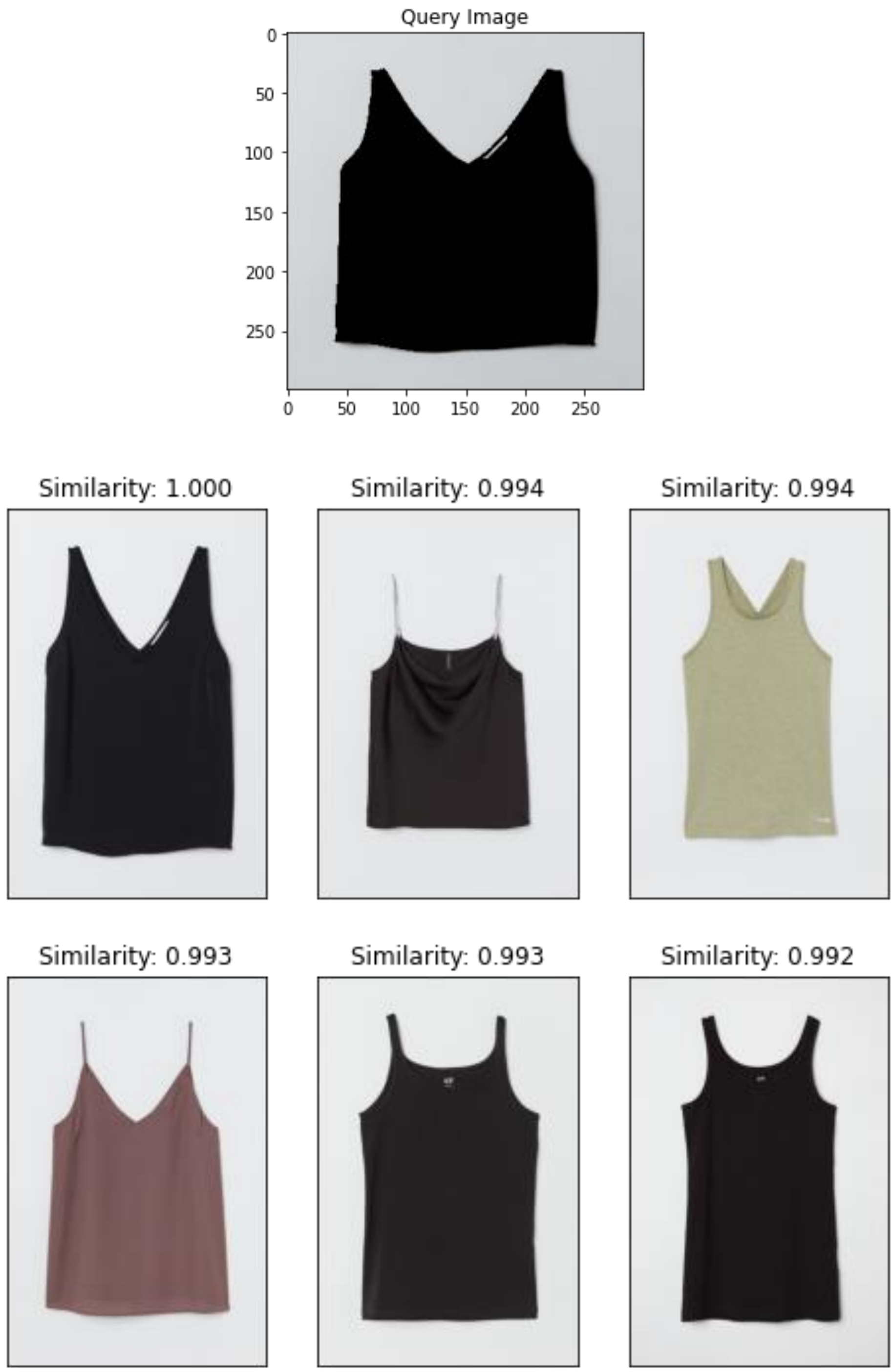
Figure 15

Visualization of Similar Images in Dress (Training)

Note. Image index 387 is used as query image. In fact, the query image is among to training images; thus, it matches to one trained image with cosine similarity of 1. A similar image is one in the dataset that has a similar hash value to this dress. The returned images are sorted by cosine similarity.

Figure 16

Visualization of Similar Images in Vest Top (Training)



Note. The query image was selected from the LSH table created with the training dataset. There are the top six similar images shown based on the query image. The returned images are sorted by cosine similarity.

Figure 17

Visualization of Similar Images in Sweater (Training)

Note. The query image was selected from the hash table created with the training dataset. There are the top six similar images shown based on the query image. The returned images are sorted by cosine similarity.

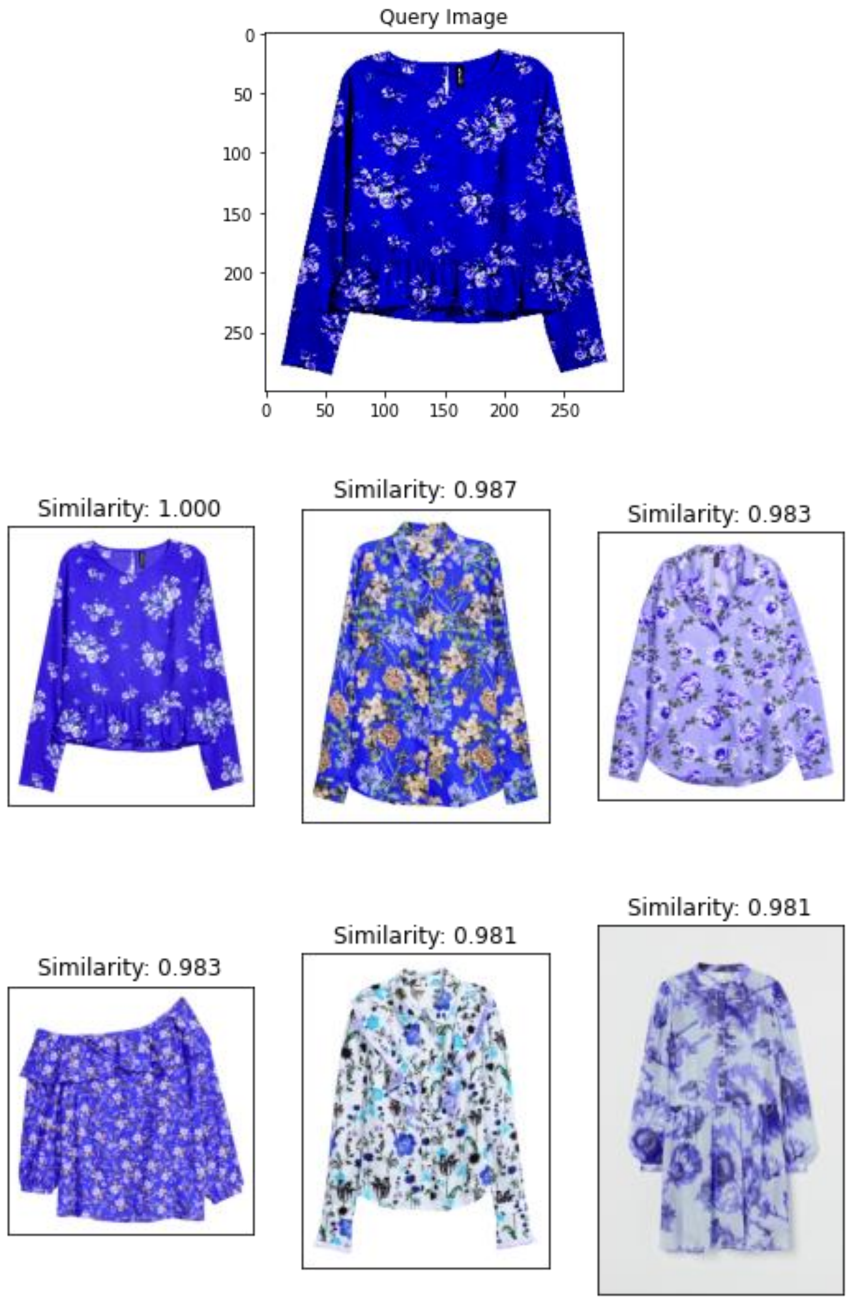
Figure 18

Visualization of Similar Images in Blouse 1 (Training)

Note. The query image was selected from the hash table created with the training dataset. There are the top six similar images shown based on the query image. The returned images are sorted by cosine similarity.

Figure 19

Visualization of Similar Images in Blouse 2 (Training)



Note. The query image was selected from the hash table created with the training dataset. There are the top six similar images shown based on the query image. The returned images are sorted by cosine similarity.

For Figures 15 – 19, all the query images happened to be belonged to the images selected from training dataset. That's why first returned recommended images has cosine similarity of 1. However, Figures 20 – 26 were with query images (from vest top, skirts, trousers, blouse, sweater, dress, and t-shirt) are not in training dataset. Therefore, through this experiment we validate how the proposed recommendation system perform with brand-new test data. As before, each test image was embedded through VGG16, and the signatures of each embedded feature vector is obtained through random projection based LSH.

Figure 20

Visualization of Similar Images in Vest Top (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 21

Visualization of Similar Images in Skirts (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 22

Visualization of Similar Images in Trousers (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 23

Visualization of Similar Images in Blouse (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 24

Visualization of Similar Images in Sweater (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 25

Visualization of Similar Images in Dress (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 26

Visualization of Similar Images in T-Shirt (Test)

Note. The image that is added later as a test, is embedded, and appears in the signature list with LSH, is the first image from the top (query image). Among the items in the dataset, the top six similar images appear.

Figure 27 shows the similarities between seven *(7)* classes for both training and test. Furthermore, it shows the similarities between the images in each class. In Figure 28 (a) each class contains 700 training images, and in Figure 28 (b) each class contains 150 test images. In the figure, embedded similarities between images are visualized in relation to the number of images.

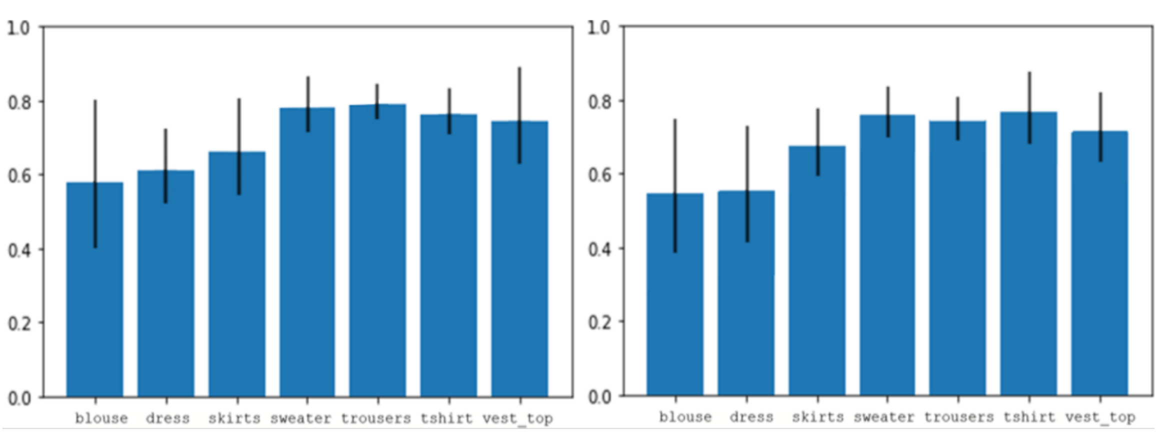
Figure 28 shows the similarities between the signatures by LSH for both training and test. Also, it shows the similarities between the images in each signature. It is not necessary for every signature to have an image. Random vector creates eight *(8)* different signatures in aggregation, but one of them is left blank since eight *(8)* groups could not be created. As shown below, the signatures contain the following number of images.

[0]=> 4
[1]=> 25
[2]=>
[3]=> 7
[4]=> 1002
[5]=> 989
[6]=> 125
[7]=> 299

Figure 27

Training and Test Data for Embedded Feature

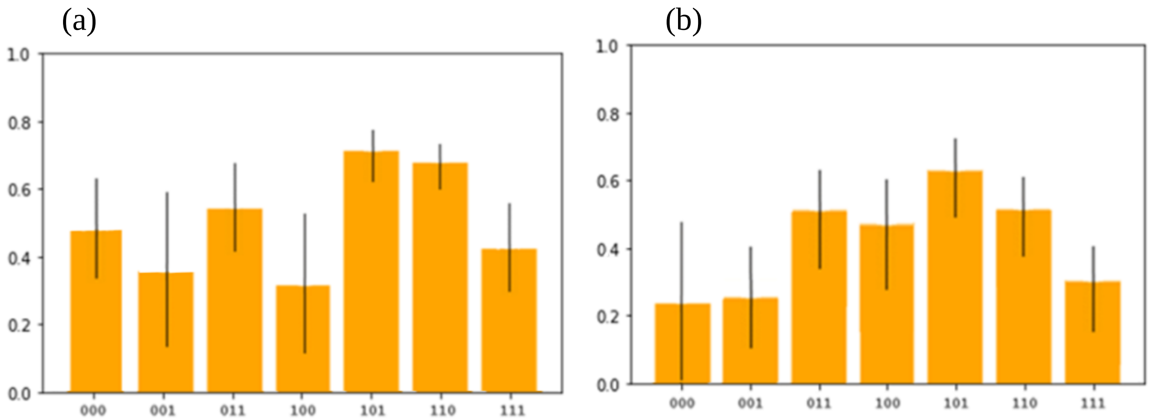
(a) (b)



Note. (a) training data (*700*/class) (b) test data (*150*/class)

Figure 28

Training and Test Data Followed by LSH

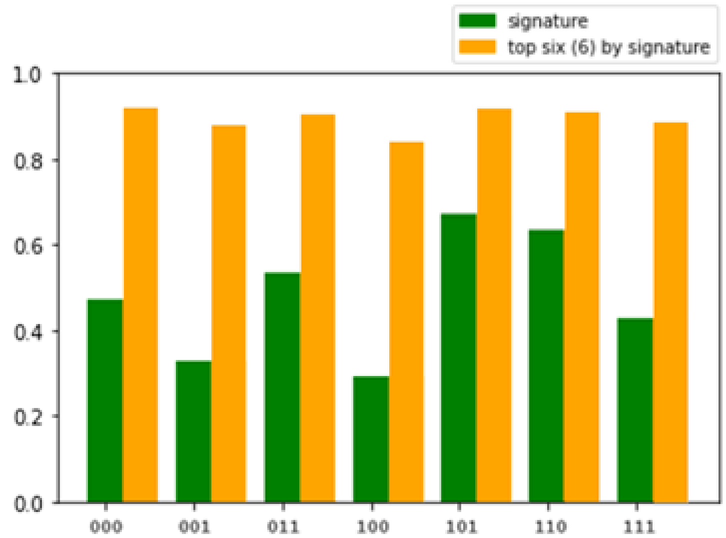


Note. (a) training data (number of images for each signature/signature) (b) test data (images with close cosine similarity/ total images for each signature)

Figure 29 shows the cosine similarity averaged over all possible cosine similarities for images in each of the seven *(7)* signatures. Each green bar is based on the signature after LSH (Figure 28 (a)). In the Hash table, one signature contains images different from actual class neighbors. It is called signature neighbor. Signatures contain maybe 20, 30, or 100 similar images. When we do a query of image search, it is going into hash signature and there finds the same signature images and get the cosine values. As previously mentioned in Figure 14, one of the signatures is empty (signature 2) and therefore there is no binary number (010).

Figure 29

Cosine Similarity Between LSH Signature and Top-6 by Signature



The orange bars are based on the signature after LSH. Different from blue bars, it shows the most cosine similarity with the top 6. This figure shows that hash signature neighbors have more similarity than images with an identical original class label.

Cosine within similarity is shown in Figure 30. It compares 700 images of training datasets in seven *(7)* classes with values between cosine [-1, 1].

The cosine distance is shown in Figure 31. Based on the formula "1 - cosine" and the range [0, 2], a cosine distance was calculated. Thus, the distance between classes was calculated in this way, explaining their differences.

Figure 30

Cosine within Similarity [-1, 1]

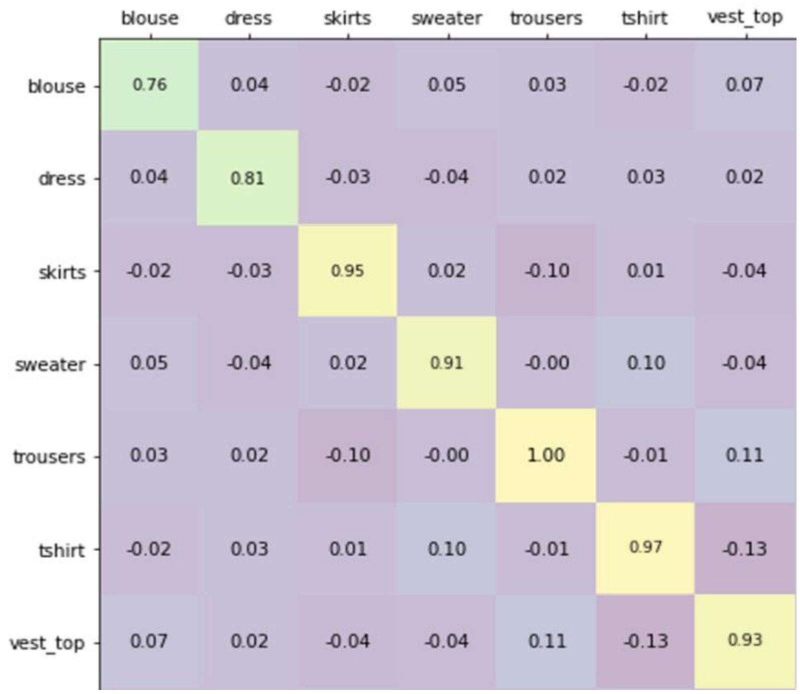
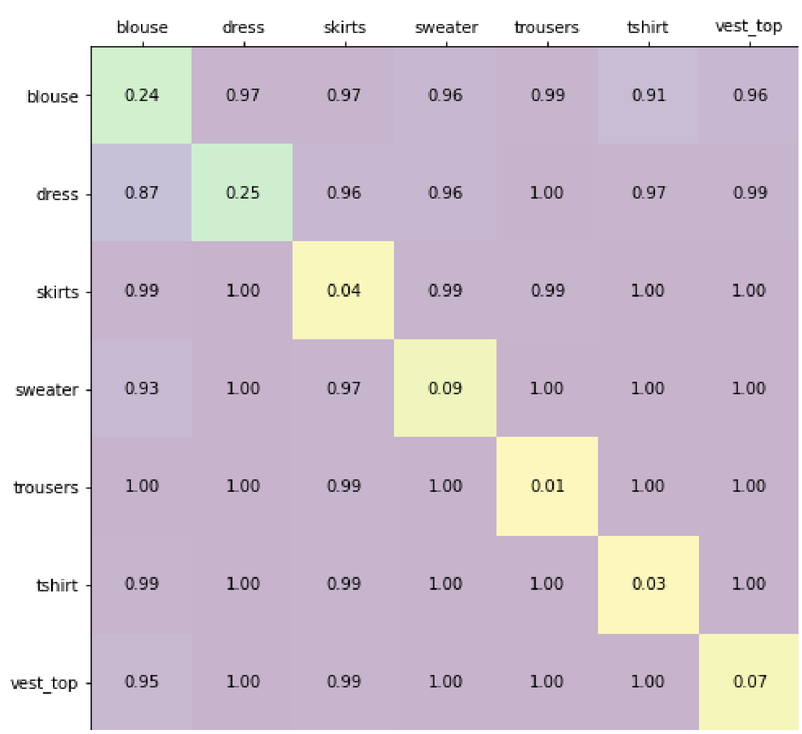


Figure 31

Cosine Distance



CHAPTER VII

Summary & Future Work

Summary

A recommendation system is one of the most useful technologies for users and the industry in online sales as it online facilitates and optimizes searches. A recommendation system uses data from interactions between people and products to detect preferences, making it a very useful tool in helping users discover new products.

This study developed a cloth recommendation system, which employs transfer learning with three *(3)* pre-trained deep learning models: VGG16, ResNet50, and InceptionV3. Furthermore, locality sensitive hashing with random projection to improve search performance from the candidate image search. The embedded features of the image were extracted using transfer learning and a fast recommendation was achieved by using locality-sensitive hashing.

We used *7,000* images out of *39,737* images in original dataset (H&M Group), which has seven (*7)* classes including blouse, dress, sweater, skirt, trousers, t-shirt, and vest top. There are *1,000* images in each class divided into three datasets: *70%* training, *15%* validation, and *15%* test. The images that were trained with the Transfer Learning model VGG16 achieved training accuracy result of 98% and test accuracy result of *90.56%*.

A recommendation system is implemented, and its performance is measured in terms of training accuracy, test accuracy, and cosine similarity between images in a same class.

Particularly, the proposed system combines two methods, which include feature embeddings through a pre-trained CNN model, VGG16, and LSH through random projection. This study has shown an example of the combination of the two methods for apparel recommendation; thus, an improved online shopping experience is provided, and technology is developed to help industrial brands sell more products online.

Future Work

Currently, the prototype of apparel recommendation systems with minimal functionalities. Therefore, there exist various directions to improve the current model. Below is the list of future directions.

Only color images are used for the study; thus, it is good to investigate how the performances change with black and white images as well as with the mixture of both color and black and white images.

Considering the effects of hashing on both storage and speed, currently only one image group with seven (7) classes; thus, the feasibility and the applicability of the proposed system need to be validated with large scale datasets with varied data types.

Together with the current random projection based LSH, it is interesting to see how other existing LSH methods, which have been widely used in other application domains, can be employed.

We observed that randomly generated vectors sometimes result in an undesirable performance. Therefore, we plan to explore a way how we can find the best random vectors.

As more tables will perform more comparable candidate images, it is observed that working with more tables will yield good results. Therefore, the best scenario will be to generate multiple (i.e., three, five, or any number to use) LSH tables, each of which has a different set of three random vectors.

For better virtualization, we would like to take advantage of dimensional reduction techniques such as TriMAP and 3D t-SNE (Gruber, 2021).

Last but not the least, we will examine more advanced subscriptions or more advanced alternatives to high performance computing platforms such as Google Colab in order to handle large datasets with faster computation.

REFERENCES

Brain, G. (2017, June 30). *Inception V3 model* [Dataset]. Kaggle. https://www.kaggle.com/datasets/google-brain/inception-v3.

Brownlee, J. (2019, September 16). *A gentle introduction to transfer learning for Deep learning*. Machine Learning Mastery. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Chakraborty, S., Hoque, M. S., Rahman Jeem, N., Biswas, M. C., Bardhan, D., & Lobaton, E. (2021). Fashion recommendation systems, models and methods: A Review. *Informatics*, *8*(3), 49. https://doi.org/10.3390/informatics8030049

Dai, W. (2021). *Clothing fashion style recommendation system* [Master's thesis, Northeastern University]. Digital Repository Service. https://repository.library.northeastern.edu/files/neu:rx914n60r.

Deldjoo, Y., Nazary, F., Ramisa, A., Mcauley, J., Pellegrini, G., Bellogin, A., & Di Noia, T. (2022, February 6). *A review of modern fashion recommender systems*. arXiv. https://arxiv.org/abs/2202.02757

Feng, Z., Luo, X., Yang, T., & Kita, K. (2018). *An object detection system based on YOLOV2 in fashion apparel* [Conference paper]. 2018 IEEE 4th International Conference on Computer and Communications (ICCC). Chengdu, China. https://doi.org/10.1109/compcomm.2018.8780944

Google. (n.d.). *Colaboratory*. Google Colab. https://research.google.com/colaboratory/faq.html

Guan, C., Qin, S., Ling, W., & Ding, G. (2016). Apparel recommendation system evolution: An empirical review. *International Journal of Clothing Science and Technology (28)*6, pp. 854-879.

Gruber, M. (2021). *Why you should not rely on t-sne, umap or trimap*. https://towardsdatascience.com/why-you-should-not-rely-on-t-sne-umap-or-trimap-f8f5dc333e59

H&M Group. (n.d.). *H&M personalized fashion recommendations*. Kaggle. https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations.

Huang, C.-M., Chia-Po Wei, & Wang, Y.-C. F. (2013). *Active learning-based clothing image recommendation with implicit user preferences* [Conference paper]. 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW). San Jose, CA, USA. https://doi.org/10.1109/icmew.2013.6618318

JananiSBabu. (2015). ResNet50_From_Scratch_Tensorflow. https://jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow/

Keras. (2017). *VGG-16 Pre-trained Model for Keras* [Dataset]. Kaggle. https://www.kaggle.com/datasets/keras/vgg16

Keras. (2015). *ResNet-50* [Dataset]. Kaggle. https://www.kaggle.com/datasets/keras/resnet50

Lee, C.-H., & Lin, C.-W. (2021). A two-phase fashion apparel detection method based on yolov4. *Applied Sciences*, *11*(9), 3782. https://doi.org/10.3390/app11093782.

Lao, B., & Jagadeesh, K.A. (2015). Convolutional neural networks for fashion classification and object detection. *CCCV Computer Vision*.

Santhosh, H. (2018). *Locality sensitive hashing for similar item search*. https://towardsdatascience.com/locality-sensitive-hashing-for-music-search-f2f1940ace23

Sha, D., Wang, D., Zhou, X., Feng, S., Zhang, Y., & Yu, G. (2016). An approach for clothing recommendation based on multiple image attributes. *Web-Age Information Management*, 272–285. https://doi.org/10.1007/978-3-319-39937-9_21

Wen, Y., Liu, X., & Xu, B. (2018). *Personalized clothing recommendation based on knowledge graph* [Conference paper]. 2018 International Conference on Audio, Language and Image Processing (ICALIP). Shanghai, China. https://doi.org/10.1109/icalip.2018.8455311

VITA

Kubra Gundogan

EXPERIENCES

Graduate Teaching & Research Assistant-- Department of Computer Science, Sam Houston State University

Thesis: Apparel Recommendation with Transfer Learning and Locality Sensitive Hashing 2021-2022

- Assisting the instructor in the tasks of taking attendance, exam proctoring, collecting assignments, grading, and recording all data.
- Tutoring and mentoring for Database Systems, Database Security, Data Mining, Computer Networks, and Programming Fundamentals courses.
- U.S. Department of Homeland Security - Towards efficient and resilient wireless networks for public safety applications
- Research with undergraduates' students in Genome Science and Computational Biology (REU)

Math and Statistics Tutor – Academic Success Center, Sam Houston State University 2019-2021

- To encourage learning and development in a safe way for all students
- Ability to teach and learn mathematics and statistics as a peer tutor

Other Experiences

- Academic Journal ACM Digital Library
  ICSCA 2022

SKILLS

My SQL | SQL | PHP | Python | Keras | C++ | Android | SPSS | Math | Statistics methods | Operating Systems (Windows, MacOS, Linux) | Deep & Machine Learning Algorithms | MATLAB | Google Colaboratory

EDUCATION

Sam Houston State University

Master of Science in Computing & Data Science 2022

Bachelor of Science in Software Engineering 2020

Firat University

Bachelor of Science in Software Engineering 2020

AWARDS

- College of Science and Engineering Technology Graduate Achievement Scholarship 2021 – 2022