REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
FACULTY OF MECHANICAL ENGINEERING
INDUSTRIAL ENGINEERING DEPARTMENT

DESIGN AND IMPLEMENTATION OF AN OUTFIT RECOMMENDATION SYSTEM: AN APPLICATION REGARDING FASHION UNDERSTANDING

İREM ATILGAN 17061036
GÖKHAN KÖSE 16061081

END4000
GRADUATION THESIS

ADVISOR
ASSOC. PROF. DR. CEYDA ŞEN

İSTANBUL, 2021

PREFACE

We are pleased to see that we have implemented an idea that we have been pursuing for almost two years. We would like to thank our thesis advisor Assoc. Prof. Dr. Ceyda ŞEN, who has been with us since the beginning of the journey of our idea, played a major role in laying the foundations of the project, and supported us at every stage of the process with her advice. She motivated us to always aim for the better in our work and taught us a lot about how to manage a long-term project.

We would also like to thank Res. Asst. Eyüp Ensar IŞIK who helped us by offering different perspectives on the challenges of the study.

We would also like to thank our families and friends who gave us feedback and support throughout the study.

June, 2021

İrem Atılgan, Gökhan Köse

TABLE OF CONTENTS

ABBREVIATION LIST

ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
Bi-LSTM	Bidirectional- Long Short Term Memory
BOW	Bag of Words
CNN	Convolutional Neural Networks
DOG	Difference of Gaussian
GAP	Global Average Pooling
HCM	Hierarchical Collocation Model
HOG	Histogram of Gradients
MAE	Mean Absolute Error
MAP	Mean Of The Average Precisions
MCN	Multi-Layered Comparison Network
MSD	Mean Squared Difference
MSE	Mean Square Error
NDCG	Normalized Discounted Cumulative Gain
NMAE	Normalized Mean Absolute Error
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SIFT	Scale Invariant Feature Transformation
SOM	Self-Organizing Feature Map

LIST OF FIGURES

LIST OF TABLES

ÖZET

DESIGN AND IMPLEMENTATION OF AN OUTFIT RECOMMENDATION SYSTEM : AN APPLICATION REGARDING FASHION UNDERSTANDING

Gökhan Köse

İrem Atılgan

Endüstri Mühendisliği Bölümü

Bitirme Çalışması

Danışman: Doç. Dr. Ceyda GÜNGÖR ŞEN

Moda sektöründe “uyum” kavramı günümüzde birçok nesnel ve öznel konunun yansıması olarak yer almaktadır. Kıyafetlerin kendine has fonksiyonel ve stilistik özelliklerini, başka kıyafetlerle olan ilişkilerini tarif etmek güçtür; Bu sebeple bu ilişkileri modellemek de büyük problem oluşturmaktadır. Bunun bir sonucu olarak, özellikle modanın yoğunlukla yer aldığı e-ticaret sektöründe bu karmaşık bileşenleri ve ilişkileri ifade edebilecek bir öneri sistemine oldukça ihtiyaç duyulmaktadır.

Bu çalışma, bir ürün havuzunda yer alan kıyafetlerin birbirleri ile olan alt ve üst seviye ilişkilerini kavrayabilen, bir kıyafetin tamamlayıcısı başka bir kıyafet sunabilen, tamamlayıcı bir ürün öneri sistemi tasarlamayı ve uygulamaya geçirmeyi hedeflemekte; bunu sağlamak için ise ürünlerin görsel ve metinsel (kategori, başlık) özelliklerini dikkate alacak bir derin öğrenme modeli oluşturmakta ve literatürde sıklıkla kullanılan Polyvore veriseti ile eğitim ve test aşamalarını gerçekleştirmektedir. Literatürdeki diğer çalışmalara kıyasla benzerlik ve uyum kavramları farklı ele alınmış ve bu durum sistemin tasarımına da yansımıştır.

Çalışmanın birinci bölümü olan “Giriş” bölümünde çalışmanın ortaya çıkmasına sebep olan faktörlere değinilmiş, çalışmanın amacına ve hipotezine yer verilmiştir. İkinci bölümde, çalışmada kullanılan temel teknik ve kavramların tarihçesi ve içeriğinden; Üçüncü bölümde çalışmanın konu alanına dahil olan akademik çalışmalardan; Dördüncü bölümde sistemin kavramsal tasarımından ve oluşturulan modelin çalışma prensibinden; Beşinci bölümde, tasarlanan sistem için gerekli girdilerden ve modelin gerçeklenmesinden; Altıncı bölümde, gerçeklenen modelin, sunmuş olduğumuz performans metriklerine göre elde ettiği sonuçlara ve bunların yorumlanmasından; Son bölüm olan yedinci bölümde ise özetle çalışmadan edindiğimiz sonuçlardan, bu konuda yapılan diğer çalışmalar için sunduğumuz önerilerimizden ve gelecek çalışmalar için planladıklarımızdan bahsedilmiştir.

Anahtar Kelimeler: Öneri sistemi, e-ticaret, moda, uyum, derin öğrenme, yapay sinir ağı.

DESIGN AND IMPLEMENTATION OF AN OUTFIT RECOMMENDATION SYSTEM : AN APPLICATION REGARDING FASHION UNDERSTANDING

İrem Atılgan

Gökhan Köse

Department of Industrial Engineering

Graduation Thesis

Advisor: Assoc. Prof. Dr. Ceyda Şen

The concept of “compatibility” in the fashion industry is a reflection of many objective and subjective issues today. It is difficult to describe the functional and stylistic features of the clothes and their relations with other clothes; For this reason, modeling these relationships also poses a big problem. As a result of this, there is a great need for a suggestion system that can express these complex components and relationships, especially in the e-commerce sector where fashion is heavily involved.

This study aims to design and implement a complementary product recommendation system that can comprehend the lower and upper level relations of the clothes in a product pool with each other, and can offer another outfit that is complementary to one outfit. In order to achieve this, it creates a deep learning model that will take into account the visual and textual (category, title) features of the products and carries out the training and testing stages with the Polyvore dataset, which is frequently used in the literature. Compared to other studies in the literature, the concepts of similarity and compatibility were handled differently and this situation was also reflected in the design of the system.

Compared to other studies in the literature, the concepts of similarity and compatibility were handled differently and this situation was also reflected in the design of the system.

In the first part of the study, "Introduction", the factors that led to the emergence of the study were mentioned, the purpose and hypothesis of the study were included. In the second part, the history and content of the basic techniques and concepts used in the study; In the third part, academic studies included in the subject area of the study; In the fourth chapter, the conceptual design of the system and the working principle of the created model; In the fifth chapter, the necessary inputs for the designed system and the implementation of the model; In the sixth section, the results of the implemented model according to the performance metrics we have presented and their interpretation; In the seventh chapter, which is the last chapter, the results we obtained from the study, our suggestions for other studies on this subject and our plans for future studies are mentioned.

Keywords: Recommendation systems, e-commerce, fashion, compatibility, deep learning, artificial neural network

CHAPTER 1

INTRODUCTION

Recommendation systems have been an area that has been frequently studied since the beginning of the 21st century. The main reason for this is that nowadays people have so many more options and therefore have difficulty identifying the right option for them. For this reason, the recommendation systems, which are mostly known with Amazon in the field of e-commerce, have achieved great success with Netflix, Instagram and TikTok, which are film and social media platforms. In addition, suggestion systems have utilized the selection process of people with personal recommendations such as books, diet suggestions or functional suggestions such as routes for pedestrians and vehicles.
Fashion, which is a sector where suggestion systems frequently take place, is one of the areas that attract attention with the development and widespread of e-commerce, especially in recent years. It is one of the fields where subjective and objective opinions are frequently mixed with each other and therefore it is difficult to define the right option for everyone.

E-commerce sites have gained great popularity in recent years, especially with the decrease in close contact shopping with the pandemic period brought by the COVID-19. However, according to the statistical data in ECommerceDB [1], the fashion industry in e-commerce ranks as the largest sector in Turkey, which constitutes 42% of e-commerce revenue. The reason for this is that, although it is not the most essential, clothing has an important place in people's lives. Determining what to wear during the day is a complex task; It requires both subjective and objective thinking. For example, which colors to use together or which patterns and cuts will be chosen may depend entirely on the individual, but factors such as weather, the type of clothing, the type of event may place the decisions to be made within a certain logic. When examined from a sociological perspective, even the opinions of the society can play a big role in the decisions taken. Considering all these factors and making a decision by making plans for certain time intervals accordingly is both time consuming and tiring for many people. Apart from this, trying to keep in mind a lot of things in daily life generally leads to the inability to recognize the options available and to spend limited resources like money and time unnecessarily to buy clothes.

All these problems have attracted the attention of fashion retail companies and have gained great importance in recent years, and their solutions have gained momentum mostly in the field of e-commerce. Many fashion retail companies such as LC Waikiki, Zara, Mango, H&M now feature products that are "compatible" or "similar" to the products that customers have examined for customers. Offering such recommendations, of course, improves the customer experience, thereby increasing the seller's product sales and profits. In the "State of the Fashion Industry in 2020" report [2] published by McKinsey, it is mentioned that the importance of e-commerce companies in the clothing sector, especially during the pandemic period, and that companies' recognition of target users and being able to offer different ideas to these users will help them stand out in the industry in the future.

When such systems offer suggestions, traditional suggestion systems techniques are often used. The typical approach to generating recommendations is to create product relationships using product group extraction techniques such as the Association Rule Extraction (ARM) method using the customers' knowledge to buy products together [3] [4] or segmentation using customer information. Such approaches rely heavily on purchasing history information to identify compatible products. However, making suggestions has become more difficult with the emergence of the "cold start" problem for clothes that are compatible with each other but have not been seen together yet [5]. In addition, in an area such as fashion, where low-level (color, texture, pattern, etc.) and high-level (style, functionality, etc.) features of products play a very large role in preferences, using information such as purchasing/reviewing products can ignore many complex and valuable relationships.

In the last decade, various academic studies have been carried out under the concept of "fashion compatibility". Especially the subject of "Deep Learning" has achieved great success in deriving complex relationships between products. The fact that it does not require expertise in the field for feature design and that it is open to heterogeneous information (text, visual, audio, video, etc.) has caused deep learning to be preferred frequently in suggestion systems studies.

Academic studies on the inference of clothes and recommendation of complementary clothes are divided into three bases: textual [6], visual [7] [8] and textual-visual [9] depending on the type of information used. In [6], complementary products were created by using only the information of the product titles; Siamese LSTM (Long-Short Term Memory) and the "compatibility matrix" presented by the study were used to grasp compatibility and taste. In [7], using only the clothing images, it was provided to make suggestions among the clothes from different categories with the Convolutional Neural Networks (CNN) and Mixture of Experts and Mahalanobis Transformation methods, in a non-metric way and without the need for category names. In [8], Siamese CNNs were trained by using the Nearest Neighbor Retrieval method in order to keep the pairs close to each other in space. In [9], a mixed model was created with the help of CNNs, RNNs and LSTMs by using both textual and visual information of the clothes, and as a result, complementary products were presented.

While measuring the success of the models created in these studies, mainly the data of buying and examining together clothes from different e-commerce sites (Amazon [6-9], Taobao [6]) were used. Although this information gives clues about the compatibility of the clothes, they do not get close enough to the truth and can lead to the wrong conclusion. The main reason for this situation is that the products purchased together do not have to be compatible with each other; a user may have made a purchase for two separate occasions. Considering this situation, [9] determined a threshold level as a solution, and accepted pairs of clothes purchased exceeding this level as compatible. However, this type of filtering is not enough to prevent errors.

In this study, a complementary outfit recommendation system which returns a sequence of suggestions of products to complement a given product and also suits the general fashion taste has been designed. In this system, the user will be able to receive complementary product suggestions if he/she chooses one of the clothes he/she has chosen.

However, unlike many other academic studies, the study highlighted the types of clothing while observing compatibility and handled concepts of similarity and compatibility in a different way.

Since both visual and textual data were needed in the realization of the system, natural language processing and image processing techniques were used and deep learning models were exploited to infer the relationships between clothes. For the training and test stage of the deep learning model, the Polyvore dataset [10], which has been frequently subjected to many studies [11], is used.

1.1 Purpose of the Thesis

The aim of the research is to design and implement an outfit recommendation system that can define the fashion understanding of the communities in a measurable way and accordingly offer a complementary product for another product, selected by individuals.

1.2 Hypothesis

In a subjective and controversial area such as fashion, the concept of "compatibility" can be explained with sufficient information and taking into account a particular context (taste of the firm, user or a community).

CHAPTER 2

OVERVIEW OF RECOMMENDER SYSTEMS AND ARTIFICIAL NEURAL NETWORKS

In this part of our study, we have made an overview of recommendation systems developed to offer suggestions to users and artificial neural networks that are increasingly used in this regard. First, we examined the methods and techniques used in recommender systems. We specified the metrics discussed in this systems' evaluation. Secondly, we examined the artificial neural network structure, elements and frequently used artificial neural network architectures. Convolutional neural networks are also emphasized for image processing operations. Finally, we finished our work in this part by examining deep learning performance metrics.

2.1 Recommendation Systems

Unlimited selection range was brought to customers by increasing the number of products and product variety. Among so many options, customers have begun to find it difficult to find the product that best suits them. The increase in internet usage leads us to the search for meaning from the data that emerged. The recommendation system is the solution to this problem. Recommendation systems by definition: calculates and provides relevant content to the user based on knowledge of the user, content, and interactions between the user and the item [12]. The rapid growth of information in the virtual space brings more opportunities in this area. Recommendation systems have become very advanced, especially with the introduction of artificial intelligence. Before understanding the current situation of the recommendation system we have to know the history of it. Thus, we can see the evolution of the recommendation system and observe other methods. Why were they used, and why we don't prefer to use them nowadays.

2.1.1 History Of The Recommendation Systems

The development of Recommendation systems that generate suggestions for users in modern terms has taken place in the last 25 years. Recommendation systems base their origins in a global discussion/opinion exchange system called Usenet. In this system established by Duke University, users' entries were classified by throwing them into certain newsgroups, and these groups were divided into subgroups if needed. A limited number of studies could be conducted from this system in the late 1970s until 1992, and these were mostly on content filtering [13].

It can be said that the establishment of the first recommendation system its publication at Xerox's research center in Palo Alto is the modern beginning of the recommendation systems literature [13]. After that day, big companies and technology firms increased their research on the Recommendation system. The use of recommendation systems has become widespread to a great extent, and the volume of practical work and literature has increased considerably since 2005.

Academic conferences and various competitions were organized under the sponsorship of large companies. Searching in academic databases with the keyword "recommender system", there were 1531 records between 1969-2005, while there are 30945 records from 2005 to the present. This is enough to show the growth momentum in the literature. [13]

Netflix, Zara, Amazon are prime examples of companies that use a recommendation system and help directly or indirectly to the improvement of it. One of the key points behind youtube's success is the recommendation algorithm. Another area is Social media platforms. They are using good recommendation systems. Facebook offers suitable friends for you with the people you may know section. Instagram discover section recommend feasible videos according to your likes and your friends like. The Tiktok which is a social media app uses a great recommendation system, thus allows users to spend more time on the app. Let’s examine the methods used in this process with their pros and cons.

2.1.2 Filtering Model

Before we started to introduce the models, we have decided that it is necessary to explain some terms that we see in all researches and that we will use.

Explicit: When a user manually gives a content item a rating, it’s called an explicit rating. Like and rating buttons are the most known way to reach explicit ratings. The easiest way for a system to populate the user-item matrix, in theory at least, is to ask users to do it themselves [12].

Implicit: Implicit ratings are deduced from monitoring people’s behavior. Best examples are activities recorded by monitoring the use such as click number on the website, time spent on that page.

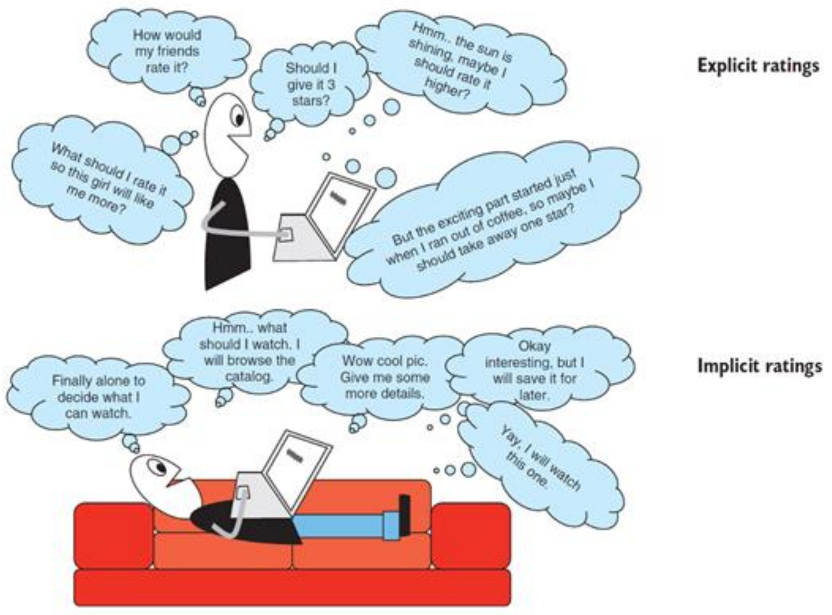

How would my friends rate it?
Should I give it 3 stars?
Hmm.. the sun is shining, maybe I should rate it higher?
Explicit ratings
What should I rate it so this girl will like me more?
But the exciting part started just when I ran out of coffee, so maybe I should take away one star?
Hmm.. what should I watch. I will browse the catalog.
Finally alone to decide what I can watch.
Wow cool pic. Give me some more details.
Okay interesting, but I will save it for later.
Implicit ratings
Yay, I will watch this one.

Figure 1 Explicit and Implicit Ratings [12]

Cold-start problem: If you don’t have knowledge of your users, you can’t personalize them. And having no personalization is a huge issue because you want to make new visitors feel welcome so they’ll become loyal returning customers. Repeat customers are ideal and you’ll want to keep them happy, but there’s nothing like adding a new one to the list. Cold start problem is a term used not only for serving recommendations to new users but also for introducing new items into your catalog. New items won’t show up in any of the non-personalized recommendations because they don’t have the numbers to enter into sales statistics, and they won’t appear in personalized recommendations because the system doesn’t know how to relate those to other items [12].

There are 5 different types of recommendation system working models. These are:

1) Collaborative filtering system

2) Content-based filtering system

3) Hybrid filtering system

4) Knowledge based recommendation system

5) Deep learning-based recommendation system

2.1.2.1 Collaborative Filtering

The core of collaborative recommender systems is to find users similar to an active user i.e. the user for whom a recommendation is generated. This will be achieved by considering the opinions and previously stated interests of other like-minded users. The collaborative filtering techniques are the most familiar and widely applied techniques in various applications such as e-learning, digital library, multimedia [14].

Discovering implicit information is a difficult task because it is usually hidden. Depending on the domain of the application, there are different methods for extracting implicit information from available data. Examples of implicit data include the followings:

- User’s behaviors and activities.
- User’s social and relational behaviors in a group.
- Items being visited by users.
- Expended observation time for the items.

The main drawback of the collaborative filtering approach is that any added new item will not be recommended until being rated or selected as an interesting one by other users. This is one of the possible ‘’Cold Start Problems’’. The second drawback of the collaborative filtering technique is the ‘’Data Sparseness Problem’’. It means when the number of required user ratings compared to the number of available items is very small, the quality of recommendation will be affected [14]. In turn, Collaborative

Filtering-based recommendations produce ranked results where the success rate can be improved. It is observed that examples of collaborative filtering method are put under two main categories such as user-based collaborative filtering and item-based collaborative filtering [15]. In the user-based category, there are examples that generate predictions about a given user's interest by observing users having the same appreciation pattern as the predicted user. In the item-based category, there are examples of collaborative filtering which do predictions by establishing a relationship matrix between the products. On such a relationship matrix, it is possible to showcases where the users who have purchased a product X have also purchased another product Y. After identification of these cases, the user with interests to be predicted is recommended other products, in case he/she has purchased one of the products with an identified relationship [15].

There are various metrics which are applied in the collaborative filtering techniques to find the users' similarities. The most common metrics are cosine, Pearson correlation, and mean squared difference (MSD). These metrics are defined in Eqs. (1)–(3), respectively [14]:

$$sim(x,y) = \frac{\sum_{i=1}^{m} r_{x,i} r_{y,i}}{\sum_{i=1}^{m} r_{x,i}^2 \sum_{i=1}^{m} r_{y,i}^2} \tag{2.1}$$

$$sim(x,y) = \frac{\sum_{i=1}^{m} (r_{x,i} - \bar{r}_x)(r_{y\cdot i} - \bar{r}_y)}{\sum_{i=1}^{m} (r_{x,i} - \bar{r}_x)^2 \sum_{i=1}^{m} (r_{y,i} - \bar{r}_y)^2} \tag{2.2}$$

$$sim = \frac{1}{m} \sum_{i=1}^{m} (r_{x,i} - r_{y,i})^2 \tag{2.3}$$

where $r_{x,i}$ and $r_{y,i}$ are assigned as the average rating of the rates issued by user x and user y for item i. These functions compute the similarity between users x and y. The aforementioned metrics have several problems. These metrics are dependent on the users' ratings. The most important problem related to cosine and Pearson metrics is that users who have rated the items diversely may be regarded as similar users. The weakness of MSD metric is related to the fact that users who have rated a small number of items can be considered as similar users to others [14].

2.1.2.2 Content-Based Filtering System

In content-based recommender systems, the descriptive attributes of items are used to make recommendations. The term “content” refers to these descriptions. In content-based methods, the ratings and buying behavior of users are combined with the content information available in the items. In content-based methods, the item descriptions, which are labeled with ratings, are used as training data to create a user-specific classification or regression modeling problem. For each user, the training documents correspond to the descriptions of the items she has bought or rated. The class (or dependent) variable corresponds to the specified ratings or buying behavior. These training documents are used to create a classification or regression model, which is specific to the user at hand (or active user). This user-specific model is used to predict whether the corresponding individual will like an item for which her rating or buying behavior is unknown [16].

Content-based methods have some advantages in making recommendations for new items when sufficient rating data are not available for that item. This is because other items with similar attributes might have been rated by the active user. Therefore, the supervised model will be able to leverage these ratings in conjunction with the item attributes to make recommendations even when there is no history of ratings for that item.

Content-based methods do have several disadvantages as well:

1. In many cases, content-based methods provide obvious recommendations because of the use of keywords or content. For example, if a user has never consumed an item with a particular set of keywords, such an item has no chance of being recommended. This is because the constructed model is specific to the user at hand, and the community knowledge from similar users is not leveraged. This phenomenon tends to reduce the diversity of the recommended items, which is undesirable.

2. Even though content-based methods are effective at providing recommendations for new items, they are not effective at providing recommendations for new users. This is because the training model for the target user needs to use the history of her ratings. In fact, it is usually important to have a large number of ratings available for the target user in order to make robust predictions without overfitting.

The main components of content-based systems include the (offline) preprocessing portion, the (offline) learning portion, and the online prediction portion. The offline portions are used to create a summarized model, which is often a classification or regression model. This model is then used for the online generation of recommendations for users. The various components of content-based systems are as follows:

1. Preprocessing and feature extraction: Content-based systems are used in a wide variety of domains, such as Web pages, product descriptions, news, music features, and so on. In most cases, features are extracted from these various sources to convert them into a keyword-based vector-space representation. This is the first step of any content-based recommendation system, and it is highly domain-specific. However, the proper extraction of the most informative features is essential for the effective functioning of any content-based recommender system.

2. Content-based learning of user-profiles: As discussed earlier, a content-based model is specific to a given user. Therefore, a user-specific model is constructed to predict user interests in items, based on their past history of either buying or rating items. In order to achieve this goal, user feedback is leveraged, which may be manifested in the form of previously specified ratings (explicit feedback) or user activity (implicit feedback). Such feedbacks are used in conjunction with the attributes of the items in order to construct the training data. A learning model is constructed on this training data. This stage is often not very different from classification or regression modeling, depending on whether the feedback is categorical (e.g., binary act of selecting an item), or whether the feedback is numerical (e.g., ratings or buying frequency). The resulting model is referred to as the user profile because it conceptually relates user interests (ratings) to item attributes.

3. Filtering and recommendation: In this step, the learned model from the previous step is used to make recommendations on items for specific users. It is important for this step to be very efficient because the predictions need to be performed in real time [14].

2.1.2.3 Hybrid Filtering Model

The collaborative and content-based techniques have several limitations and drawbacks. To overcome these limitations, hybrid recommender systems are introduced. The hybrid systems combine the aforementioned techniques to enhance the advantages which are achieved. In the situations that there is no information about users or their ratings, the content-based part of the hybrid recommender system can be helpful to retrieve useful information to generate recommendations. On the other hand, when information about the contents associated with the items is not sufficient, the collaborative part of the hybrid recommender system can be supportive. Consequently, the cold start and data sparseness problems of recommender systems will be resolved [14].

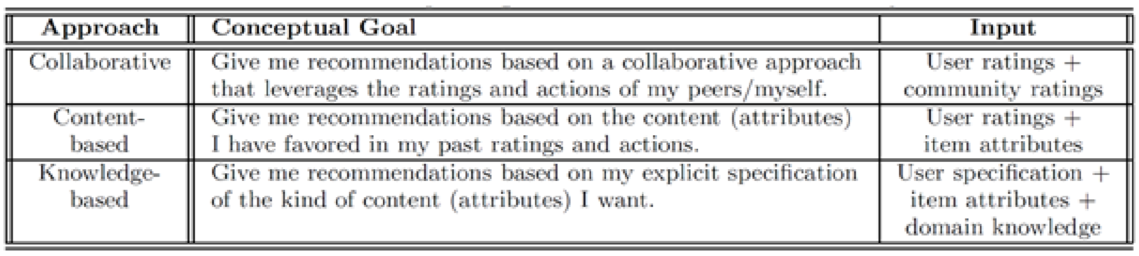
A number of the hybrid recommender systems operate based on the switching hybrid approach, which applies the content-based or collaborative recommendation technique depending on several criteria and available data. In these systems, when a new item is added to the system, the content-based recommendation technique is used to find similar and related items. Similarly, when a new user enters the system, a collaborative recommendation approach is applied to find similar users and find their interesting items to be recommended [16].

According to studies on recommender systems, especially the ones employed in discussion groups, it was realized that most of the recommender systems in this domain are based on the collaborative recommendation technique. According to the short lifetime of recommender systems associated with the discussion groups, the evaluation of such systems can be difficult and somehow tedious research activity. Castro-Herrera presents a hybrid recommender system that clusters similar contents based on their keywords by the TFIDF technique. The main goal of this system is to find similar and relevant users in forums based on their contributed posts. The collaborative filtering part of this system computes the similarity of users with the same interests who have contributed in common posts. The similarity criterion of contents in this research is based on the number of shared keywords, and also the weight and frequency of their happenings rather than their semantic similarities. This will produce two contents with the same keywords but with different concepts being placed in the same cluster [14]. This system suffers from a rating sparseness problem because it uses co-rated items to identify the neighbors in the collaborative filtering part of its suggested system.

2.1.2.4 Knowledge Based Recommendation Model

Knowledge-based recommender systems are particularly useful in the context of items that are not purchased very often. Examples include items such as real estate, automobiles, tourism requests, financial services, or expensive luxury goods. In such cases, sufficient ratings may not be available for the recommendation process. As the items are bought rarely, and with different types of detailed options, it is difficult to obtain a sufficient number of ratings for a specific instantiation of the item at hand. This problem is also encountered in the context of the cold-start problem when sufficient ratings are not available for the recommendation process. Furthermore, the nature of consumer preferences may evolve over time when dealing with such items. For example, the model of a car may evolve significantly over a few years, as a result of which the preferences may show a corresponding evolution. In other cases, it might be difficult to fully capture user interest with historical data such as ratings.

Table 1 The Conceptual goals of various recommender systems [16]



Approach	Conceptual Goal	Input
Collaborative	Give me recommendations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings + community ratings
Content-based	Give me recommendations based on the content (attributes) I have favored in my past ratings and actions.	User ratings + item attributes
Knowledge-based	Give me recommendations based on my explicit specification of the kind of content (attributes) I want.	User specification + item attributes + domain knowledge

The recommendation process is performed on the basis of similarities between customer requirements and item descriptions, or the use of constraints specifying user requirements. The process is facilitated with the use of knowledge bases, which contain data about rules and similarity functions to use during the retrieval process. In fact, the knowledge bases are so important to the effective functioning of these methods that the approach takes its name from this fact. The explicit specification of requirements results in greater control of users over the recommendation process. Knowledge-based systems are unique in that they allow the users to explicitly specify what they want. Knowledge-based recommender systems can be classified on the basis of the type of the interface (and corresponding knowledge) used to achieve the aforementioned goals:

1. Constraint-based recommender systems: In constraint-based systems, users typically specify requirements or constraints (e.g., lower or upper limits) on the item attributes. Domain-specific rules are used to match the user requirements to item attributes. These rules represent the domain-specific knowledge used by the system. Depending on the number and type of returned results, the user might have an opportunity to modify their original requirements. For example, they might relax some of their constraints when too few results are returned, or they might add more constraints. This search process is interactively repeated until the user arrives at her desired results.

2. Case-based recommender systems: In case-based recommender systems, specific cases are specified by the user as targets or anchor points. Similarity metrics are defined on the item attributes to retrieve similar items to these cases. The similarity metrics are often carefully defined in a domain-specific way. Therefore, the similarity metrics form the domain knowledge that is used in such systems. The returned results are often used as new target cases with some interactive modifications by the user. This interactive process is used to guide the user towards items of interest.

2.1.3 Recommender Systems-Performance Metrics

Evaluation metrics for recommender systems can be divided into 3 major classes. 1)Predictive accuracy metrics,

2) Classification accuracy metrics,

3) Rank accuracy metrics. We will discuss these classes

2.1.3.1 Predictive Accuracy Metrics

Predictive accuracy rating aims to close ratings estimated by the recommender. It commonly measures the evaluation of non-binary ratings. If we look into an accurate prediction of the ratings for all items, this metric fits for us. 3 main representatives are: mean absolute error (MAE), mean square error (MSE), normalized mean absolute error (NMAE), and root mean squared error (RMSE). MSE and RMSE use the squared deviations and thus emphasize larger errors in comparison to the MAE metric. NMAE normalizes the MAE metric to the range of the respective rating scale in order to make results comparable among recommenders with varying rating scales.

Recommender systems are more commonly used to display a limited list of top-ranked items or the set of all items that have been rated above a certain threshold. Many recommendation algorithms are able to provide more accurate statements about a limited set of items that the user either likes or dislikes. The estimations for many other items are rather inaccurate but often also significantly less important to users. [17]

2.1.3.2 Classification Accuracy Metrics

Table 2 Recommender System Evaluation Table

	Relevant	Irrelevant	Total
Recommended	tp	fp	tp + fp
Not Recommended	fn	tn	fn + tn
Total	tp + fn	fp + tn	N

Precision or true positive accuracy (also confidence in data mining) is calculated as the ratio of recommended items that are relevant to the total number of recommended items

$$\text{Precision} = \text{tpa} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{2.4}$$

Recall or true positive rate is calculated as the ratio of recommended items that are relevant to the total number of relevant items:

$$\text{Recall} = \text{tpr} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{2.5}$$

Fallout or false positive rate is calculated as the ratio of recommended items that are irrelevant to the total number of irrelevant items:

$$\text{Fallout} = \text{fpr} = \frac{\text{fp}}{\text{fp} + \text{tn}} \tag{2.6}$$

MissRate or false negative rate is calculated as the ratio of items not recommended but actually relevant to the total number of relevant items:

$$\text{Miss Rate} = \text{fnr} = \frac{\text{fn}}{\text{tp} + \text{fn}}$$

(2.7)

Inverse precision or true negative accuracy is calculated as the ratio of items not recommended that are irrelevant to the total number of recommender items:

$$\text{Inverse Precision} = \text{tna} = \frac{\text{tn}}{\text{fn} + \text{tn}} \tag{2.8}$$

Inverse Recall or true negative rate is calculated as the ratio of items not recommended that are irrelevant to the total number of irrelevant items:

$$\text{Inverse Recall} = \text{tnr} = \frac{\text{tn}}{\text{fp} + \text{tn}} = 1 - \text{fpr} \tag{2.9}$$

F1-measure try to combine precision and recall into a single score by calculating different types of means of both metrics:

$$\text{F1} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{2.10}$$

The arithmetic mean of all these precisions is AP:

$$\text{AP} = \frac{\sum_{r=1}^{N} P(r) * rel(r)}{\text{number of relevant documents (N)}} \tag{2.11}$$

MAP is the arithmetic mean of the average precisions of all users to get the final mean average precision:

$$\text{MAP} = \frac{\sum_{u=1}^{M} AP(u)}{\text{M}} \tag{2.12}$$

2.1.3.3 Rank Accuracy Metrics

Rank accuracy metric measures the ability of a recommender to predict the correct order of items concerning the user's preference. If the user presented with a long ordered list of items, this will be a good metric to evaluate the recommendation. The problem about these metric is that we are not provided with a full ranking of all items. In order to create a full ranking of the items, all preference values for the user have to be known. Since the user might express the same rating for several items the list will again contain groups of items that can appear in an arbitrary order. The largest problem is posed by items for which no user rating is known. These items could in fact hold an arbitrary place within the ranking [17].

Rank accuracy metrics are well suited for e-commerce applications if the metrics allow to compare partial rankings in a meaningful way.

Normalized Discounted Cumulative Gain (NDCG)

Discounted Cumulative Gain is the metric of measuring ranking quality. It is mostly used in information retrieval problems such as measuring the effectiveness of the search engine algorithm by ranking the articles it displays according to their relevance in terms of the search keyword.

- DCG accumulated at a particular rank position p is given by

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2(i+1)} \tag{2.13}$$

- Alternative formulation of DCG that places stronger emphasis on retrieving relevant documents is given by

$$DCG_\rho = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2.14}$$

Logarithmic scale for reduction provides a smooth reduction curve and hence is used. Comparing a search algorithms performance from one query to the next cannot be consistently achieved using DCG alone, so the cumulative gain at each position for a chosen value of p should be normalized across queries.

- Normalized DCG (nDCG) is given by

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{2.15}$$

- Where

$$IDCG_\rho = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2.16}$$

- Where $|REL|$ is the list of documents ordered by relevance in the corpus up to position p.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs), usually simply called neural networks (NNs) is a computing technology inspired by the information processing technique of the human brain. With ANNs, the way the simple biological nervous system works is imitated. In other words, it is digital modeling of biological neuron cells and the synaptic bond that these cells establish with each other. Neurons connect to each other in various ways to form networks. These networks have the capacity to learn, memorize and reveal the relationship between data. In other words, ANNs provide solutions to problems that normally require a person's natural abilities to think and observe. The main reason why a person can produce solutions to problems that require thinking and observing abilities is the ability of the human brain, and therefore, to learn by living or experimenting [18].

Advantages and features of artificial neural networks compared to other traditional computing methods used today can be listed as follows:

Parallelism: In most of the conventional computing methods, transactions are in a serial order. This arrangement particularly causes speed problems. Although the computer works much faster than the brain, the total speed of the brain is much higher than a computer. In artificial neural networks, operations are not linear and this is spread over the entire network. There is no time dependency between the same layers. This allows the whole system to work simultaneously and increases the speed very much. In this way, it is possible to solve nonlinear complex problems.

Learnability: Traditional computing systems operate within the scope of a specific algorithm and cannot renew weight or data itself. In this case, problems that are not fully defined cannot be solved either. It enables the determination of weights by using the samples entered into the system with artificial neural networks and this learning process can be renewed in each new study. In a study that will be suitable for the purpose, the fact that weights and connections cannot be given in advance creates a problem, while the artificial neural networks train themselves with examples and create the necessary data eliminates this problem.

Fault Tolerance: Removing any element in traditional methods makes it impossible to operate the system. The parallel structure in artificial neural networks enables the information of the network to spread to all connections. In this way, the inactivation of some connections or cells does not significantly affect the network's ability to produce the correct information, and however, the ability of networks to tolerate error compared to traditional methods.

Adaptability: The fact that weights can be reconfigured in artificial neural networks enables the artificial neural network trained to solve a specific problem to be retrained and adapted to different conditions according to the changes in the problem. This feature enabled artificial neural networks to be used effectively in areas such as sample recognition, signal processing, system diagnosis and control.

Generalization: After the artificial neural network training, the test samples that are not encountered during the training can also be evaluated and produced the desired responses. For example, it is possible to get the correct characters in the input of bad characters in character identification.

Local Information Processing: In artificial neural networks, rather than dealing with the whole problem, very complex and difficult problems can be solved thanks to the ability to deal with the parts and share the tasks.

Ease of Implementation: Using simple operations instead of complex functions provides ease of implementation.

Hardware and Speed: Artificial neural networks can be implemented with integrated circuit technology thanks to its parallel structure. This feature increases the fast information processing capability.

2.2.1 Structure Of Artificial Neural Networks

An artificial neural network consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons. Figure 2 shows a typical architecture, where lines connecting neurons are also shown. In the human brain, neurons communicate by sending signals to each other through complex connections. ANNs are based on the same principle in their attempts to simulate the learning process of the human brain using complex algorithms. Every connection has a weight attached which may have either a positive or a negative value associated with it. Positive weights activate the neuron while negative weights inhibit it.

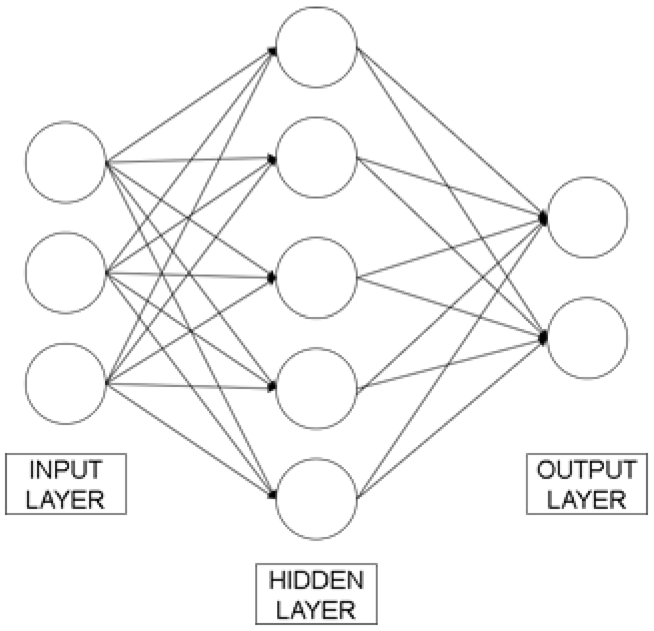


Figure 2 Architecture of a neural network

Just as biological neural networks have nerve cells, artificial neural networks also have artificial nerve cells. Artificial nerve cells are also called process elements in engineering science. This process has 5 basic elements [19]:

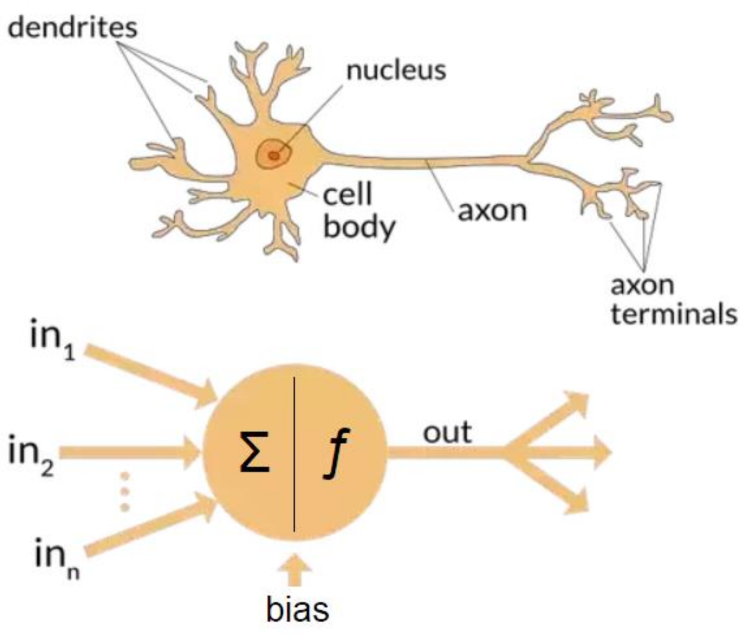


Figure 3 Structure of Biological Neuron and Artificial Neuron [20]

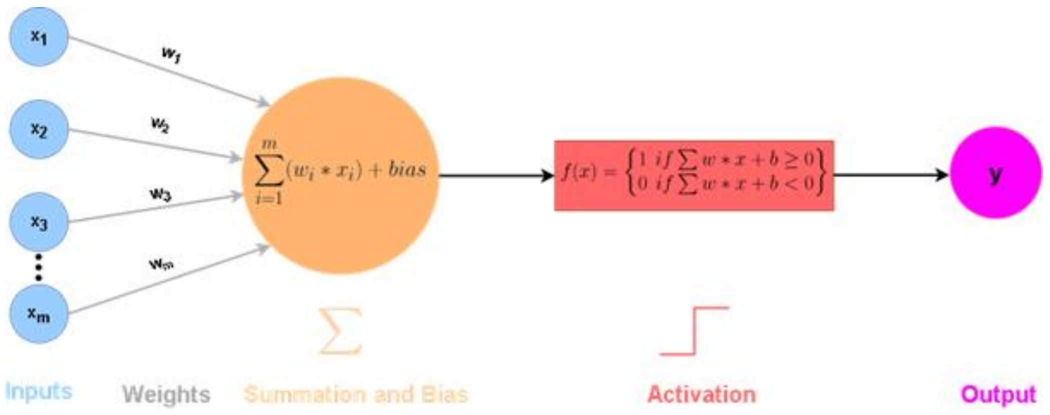

x_1
x_2
x_3
x_m
w_1
w_2
w_3
w_m
$\sum_{i=1}^{m}(w_i * x_i) + bias$
$f(x) = \begin{cases} 1 \ if \sum w * x + b \geq 0 \\ 0 \ if \sum w * x + b < 0 \end{cases}$
y
Inputs
Weights
Summation and Bias
Activation
Output

Figure 4 Structure of Artificial Neuron [21]

Inputs

Information coming from the outside world to an artificial cell. These are determined by the examples the network is asked to learn. In Figure b, inputs are shown as x_1, x_2, x_3 ...x_m which will be processed with weights.

Weights

Weights show the importance of the information coming to an artificial cell and its effect on the cell. The weight w_{i1} in the figure shows the effect of x_{i1} input on the cell. Whether the weights are big or small does not mean that they are important or not. A weight being zero may be the most important event for that network.

Summation Function (Concatenation Function)

This function calculates the net input to a cell. Various functions are used for this. The most common is the weighted sum. Here, each incoming information is collected by multiplying it by its own weight. Thus, the net input to the network is found. Some of the summation functions are given in the table below.

Table 3 Summation Function Examples [22]

Sum $Net = \sum_{i=1}^{N} x_i * w_i$	The weight values are multiplied by the inputs and the net input is calculated by adding the values found.
Product $Net = \Pi_{i=1}^{N} x_i{*}w_i$	The weight values are multiplied by the inputs and then the net input is calculated by multiplying the found values with each other.
Maximum $Net = Max(x_i{*}w_i)$	Out of n inputs, after weights are multiplied by the inputs, the biggest of them is considered the net input.
Minimum $Net = Min(x_i{*}w_i)$	Out of n inputs, after weights are multiplied by inputs, the smallest of them is accepted as net input.
Majority $Net = \sum_{i=1}^{N} Majority(x_i * w_i)$	After the weights are multiplied by the inputs out of n inputs, the number of positive and negative ones is found. The larger number is considered the net input of the cells.
Cumulative Total $Net = Net(old) + \sum_{i=1}^{N} x_i * w_i$	Information coming into the cell is mainly collected. The net input of the cell is calculated by adding the newly calculated input values to the previously received information in the cell.

The bias value allows the activation function to be shifted to the right or left. Learning does not take place when the sum of the input signals is 0. Bias neurons, whose output values are always 1, ensure that the input signals of the neurons are always non-zero. It speeds up learning and makes it difficult to stick to local optimum values. Also, bias determines the neuron's partner to respond.

Activation Function

This function determines the output the cell will produce for this input by processing the net input to the cell. The activation function is usually chosen as a nonlinear function. The nonlinearity feature of artificial neural networks, comes from the nonlinearity of activation functions. Another point to be considered when choosing the activation function is that the derivative of the function can be easily calculated. In the further part of our study, the commonly used functions of these functions are explained in more detail.

Output

It is the output value determined by the activation function. The output produced is sent to the outside world or to another cell. The cell can also send its output as input to itself. Although a process element has more than one output, it can only have one output.

2.2.2 Learning Process in Artificial Neural Networks

A learning process in the ANN context can be described as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. ANNs' ability to automatically learn from examples makes them attractive and exciting. Instead of following a set of rules specified by human experts, ANNs appear to learn underlying rules (like input-output relationships) from the given collection of representative examples [23]. This is one of the major advantages of neural networks over traditional expert systems.

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning networks are provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. The "Delta Rule" developed by Widrow-Hoff and the "Generalized Delta Rule" or "Feedback (backpropagation algorithm)" developed by Rumelhart and McClelland can be given as examples of supervised learning algorithms [24].

Unsupervised learning does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Hybrid learning combines supervised and unsupervised learning. Part of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning. This learning rule ART (Adaptive Resonance Theory) developed by Grossberg or SOM (Self Organizing Map) developed by Kohonen can be given as an example of unsupervised learning.

Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. Boltzmann Rule or Genetic Algorithms developed by Hinton and Sejnowski to solve optimization problems can be given as examples of reinforced learning.

2.2.2.1 Learning Algorithms

The sum of the squared differences between the output in the artificial neural network and the real label or target value gives us an error or cost function. This cost is calculated with the formula for square error (Square Error).

$$E_{total} = \sum \frac{1}{2}(target - output)^2 \tag{2.17}$$

We want the error function to be minimal. For this we have to iteratively converge towards the local minimum of this equation. Gradient descent, Newton, Quasi-Newton, Levenberg Marquardt optimization algorithms are used to get the local minimum of a function.

It is the most used gradient descent algorithm. This algorithm uses the first derivative to converge to the local minimum of the equation. Newton works with the second derivative and Quasi-Newton works with an algorithm similar to the second derivative. If you want to use the second derivative, this calculation is costly in terms of complexity. Its advantage is that it converges to the local minimum with little iteration.

In the gradient descent algorithm, the convergence direction is moved on a direction opposite to the gradient direction. The next new weight is found by multiplying the minimum point in that direction by the amount of a certain step or learning rate. This learning rule is called Delta Rule.

Gradient

A gradient is a slope whose angle we can measure. Like all slopes, it can be expressed as a relationship between two variables: “y over x”, or rise overrun. In this case, the y is the error produced by the neural network, and x is the parameter of the neural network. The parameter has a relationship to the error, and by changing the parameter, we can increase or decrease the error. So the gradient tells us the change we can expect in y with regard to x. To obtain this information, we must use differential calculus, which enables us to measure instantaneous rates of change, which in this case is the tangent of a changing slope expressed the relationship of the parameter to the neural network’s error. As the parameter changes, the error changes, and we want to move both variables in the direction of less error. What’s more, neural networks have parameters that process the input data sequentially, one after another. Therefore, backpropagation establishes the relationship between the neural network’s error and the parameters of the net’s last layer; then it establishes the relationship between the parameters of the neural net’s last layer those the parameters of the second-to-last layer, and so forth, in an application of the chain rule of calculus.

2.2.3 Activation Functions

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be (1) or (0), depending on the input. This is similar to the behavior of the linear perceptron in neural networks. However, only nonlinear activation functions allow such networks to compute non trivial problems using only a small number of nodes, and such activation functions are called nonlinearities. The purpose of the activation function is, besides introducing nonlinearity into the neural network, to bound the value of the neuron so that the neural network is not paralyzed by divergent [25].

2.2.3.1 Sigmoid

The sigmoid activation function, also called the logistic function, is traditionally a very popular activation function for neural networks. The input to the function is transformed into a value between 0.0 and 1.0. Inputs that are much larger than 1.0 are transformed to the value 1.0, similarly, values much smaller than 0.0 are snapped to 0.0. The shape of the function for all possible inputs is an S-shape from zero up through 0.5 to 1.0. For a long time, through the early 1990s, it was the default activation used on neural networks.

$$\sigma(u) = \frac{1}{1 + e^{(-u)}}$$

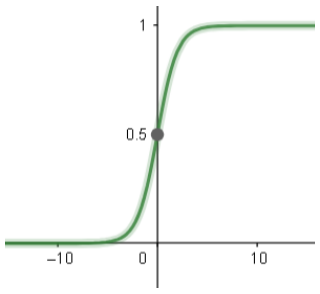


Figure 5 Graph of sigmoid function

2.2.3.2 Tanh

The hyperbolic tangent function, or tanh for short, is a similar-shaped nonlinear activation function that outputs values between -1.0 and 1.0. In the later 1990s and through the 2000s, the tanh function was preferred over the sigmoid activation function as models that used it were easier to train and often had a better predictive performance.

$$\sigma(z) = \frac{e^{z} - e^{(-z)}}{e^{(z)} + e^{(-z)}}$$

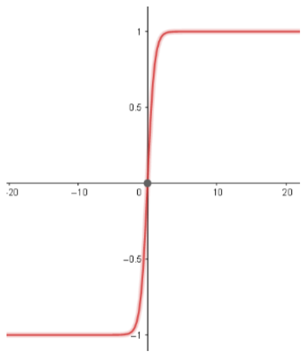


Figure 6 Graph of tanh function

2.2.3.3 ReLu

A general problem with both the sigmoid and tanh functions is that they saturate. This means that large values snap to 1.0 and small values snap to -1 or 0 for tanh and sigmoid respectively. Further, the functions are only really sensitive to changes around their mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh. The limited sensitivity and saturation of the function happen regardless of whether the summed activation from the node provided as input contains useful information or not. Once saturated, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performance of the model.

In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. A node or unit that implements this activation function is referred to as a rectified linear activation unit or ReLU for short. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks. The adoption of ReLU may easily be considered one of the few milestones in the deep learning revolution.

The rectified linear activation function is a simple calculation that returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less. The function is linear for values greater than zero, meaning it has a lot of the desirable properties of a linear activation function when training a neural network using backpropagation. Yet, it is a nonlinear function as negative values are always output as zero. [26]

$$\mathrm{ReLU}(z) = \begin{cases} z, z > 0 \\ 0, \text{otherwise} \end{cases}$$

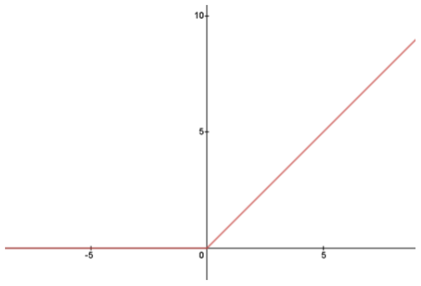


Figure 7 Graph of ReLU function

The ReLU does have some limitations. Key among the limitations of ReLU is the case where large weight updates can mean that the summed input to the activation function is always negative, regardless of the input to the network. This means that a node with this problem will forever output an activation value of 0.0.

2.2.3.4 Leaky ReLu

Some popular extensions to the ReLU relax the non-linear output of the function to allow small negative values in some way. The Leaky ReLU modifies the function to allow small negative values when the input is less than zero [26]. Thus, Leaky ReLu has brought a solution to the problem of ReLu which is giving the same value to all of the negative values, and it improves the problem from certain aspects.

$$(\mathrm{a} = 0.2)\ \mathrm{LeakyReLU}(\mathrm{z}) = \begin{cases} \mathrm{z}, \mathrm{z} > 0 \\ \mathrm{az}, \text{otherwise} \end{cases}$$

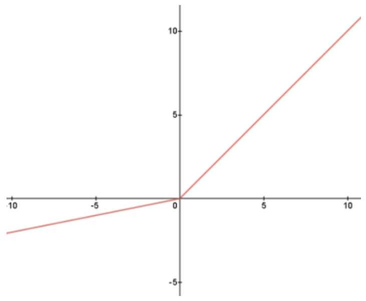


Figure 8 Graph of Leaky Relu

All methods propose different solutions and there is no optimum solution. Each method gives the best answer to a different kind of problem. Choosing the method only depends on the people.

2.2.3.5 Softmax

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class. In the last layer of an image classification network such as CNN (e.g. VGG16) used in ImageNet competitions, softmax is also applied.

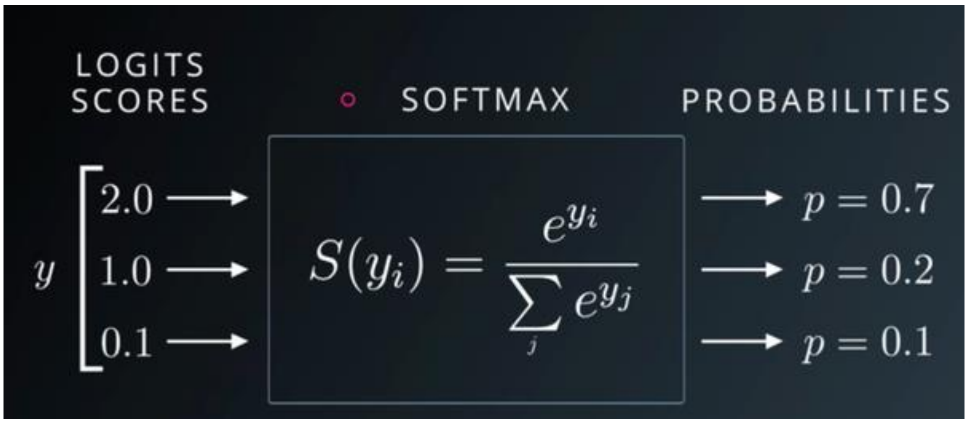


Figure 9 Softmax formula [27]

2.2.4 Structure Types of Artificial Neural Networks

Artificial neural network models can be examined in four groups as single layer perceptron, multi layer perceptron, feed forward artificial neural networks and feedback artificial neural networks. After mentioning these types, important architectures like RNN, LSTM, Bi-LSTM and CNN are explained later in our study.

2.2.4.1 Single Layer Perceptrons

Single layer networks consist of only input and output. In single layer perceptrons, the output function is linear and takes 1 or -1 values. If the output is 1, it is accepted to the first grade, if it is -1, it is accepted to the second grade [19].

In simple words, multiple input values feed up to the perceptron model, model executes with input values, and if the estimated value is the same as the required output, then the model performance is found out to be satisfied, therefore weights demand no changes. In fact, if the model doesn't meet the required result then few changes are made up in weights to minimize errors.

2.2.4.2 Multilayer Perceptrons

For many inputs, one neuron may not be enough. Multilayer concept comes into play when more than one neuron operating in parallel is needed. The input layer receives the incoming data and sends it to the intermediate layer. Incoming information is transferred to the next layer. The number of intermediate layers varies according to the problem, at least one, and is adjusted according to need. The exit of each layer becomes the entrance of the next layer. Each processing element, that is, the neuron, is connected to all neurons in the next layer. In addition, the number of neurons in the layer is determined according to the problem. The output layer determines the output of the network by processing data from previous layers. The output number of the system is equal to the number of elements in the output layer.

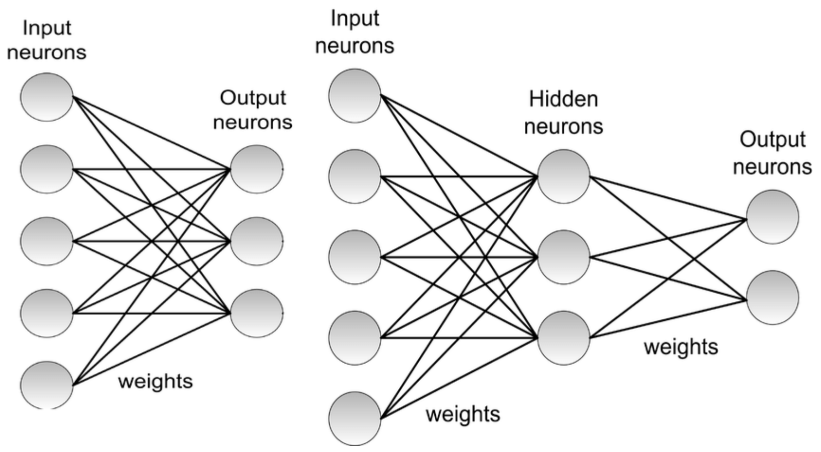


Figure 10 Structure of single and multilayer perceptrons

2.2.4.3 Feed-Forward Neural Networks

Feedforward neural networks allow for unidirectional signal flow. Also, feed-forward neural networks are often organized in layers. An example of a three-layer feed forward neural network is shown in Figure 12. These network entry nodes consist of two hidden layers and one output layer [28].

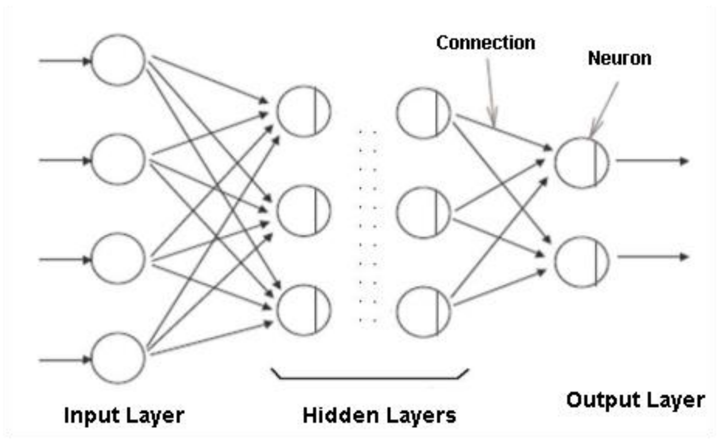


Figure 11 Structure of Feed Forward Neural Networks [19]

2.2.4.4 Feedback Neural Networks

Feedback Artificial Neural Networks, the output of at least one cell is given as input to itself or other cells, and the feedback is usually done through a delay element. Feedback can occur between cells in a layer as well as between cells between layers. With this structure, feedback ANN shows a nonlinear dynamic behavior. Therefore, feedback ANN structures with different structure and behavior can be obtained according to the way the feedback is made [29].

Feedback neural networks can be classified into two types: discrete feedback network and continuous feedback network. In both cases, when an initial input is imposed on the network, the transition process begins. The response of the neurons to the initial input is fed into the neurons as an updated input, and which, in turn, yields an updated response. The process continues until the response becomes unchanged anymore within a given accuracy. In other words the network produces the output as it reaches an equilibrium of the dynamical [30].

2.2.5 Applications Of Artificial Neural Networks

Artificial neural networks can provide solutions to problems that remain very complex for traditional techniques thanks to their learning ability. Artificial Neural Networks can be applied in many areas from financial issues to engineering and medical science, from manufacturing applications to fault detection and analysis.

Artificial neural networks are frequently used for prediction processes. The system evaluates the input according to previously learned sample data or the classification created by itself and predicts output value.

In the data filtering process, it provides the desired data according to the specified filters.

The Neural network can be used to identify a unique feature of the data and classify them into different categories without any prior knowledge of the data. This process is also called clustering.

In the data interpretation part, artificial neural networks analyze the inputs. Provides the interpretation of new events by using the information obtained from the samples collected about an event and created as a result of the training.

Artificial neural networks used for data association complement the missing information with the information learned. Completing a missing picture can be given as an example in this regard.

Pattern recognition is the study of how artificial neural networks can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns. At this point, image and natural language processing studies have increased in recent years.

2.2.6 History Of Artificial Neural Networks

In 1943, Warren McCulloch and Walter Pitts proposed a computational model (MP Model) for neural networks. In the model, the algorithm was implemented by considering the neuron as a functional logic device, and thus the theoretical research of the neural network model was started [31]. In the late 1940s, D. O. Hebb published ‘’The Organization of Behavior’’, in which he created a learning hypothesis based on the mechanism of neural plasticity that became known as Hebbian learning. In 1957 [32], Rosenblatt proposed the Perceptron model based on the M-P model. Perceptron model has the basic principle of modern neural network. Rosenblatt proved that two-layer sensors can classify inputs and also proposed an important research direction for three-layer sensors with hidden layer processing elements [33]. In 1959, B. Widrow and M.

Hoff proposed a neural network training method for adaptive linear element (Adaline) and Widrow-Hoff learning rules (also the least mean square deviation algorithm), and it was the first artificial neural to solve practically by applying it to the real project. Networking problems and encourages the application and development of neural network research. The ADALINE network model is a continuous-valued adaptive linear neuron network model that can be used for adaptive systems [34].

The first functional networks with many layers were published by Ivakhnenko and Lapa in 1965 [35], as the Group Method of Data Handling, The basics of continuous backpropagation were derived in the context of control theory by Kelley in 1960 and by Bryson in 1961, using principles of dynamic programming.

Minsky and Papert, published a book called " Perceptrons " in 1969 and pointed out that the function of simple linear perception is limited. It cannot solve the classification problem of two types of linear inseparable samples [36].

In 1972, Professor Kohonen T. proposed Self-Organizing Feature Map (SOM). Later neural networks were mainly based on the work of Kohonen T. SOM network is a kind of tutor learning network, mainly used for pattern recognition, speech recognition and classification problems [37].

In 1992, max-pooling was introduced to help with least-shift invariance and tolerance to deformation to aid 3D object recognition. Schmidhuber adopted a multi-level hierarchy of networks (1992) pre-trained one level at a time by unsupervised learning and fine-tuned by backpropagation [38].

Geoffrey Hinton et al. (2006) proposed learning a high-level representation using successive layers of binary or real-valued latent variables with a restricted Boltzmann machine to model each layer [39].

In 2012, Ng and Dean created a network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images. Unsupervised pre-training and increased computing power from GPUs and distributed computing allowed the use of larger networks, particularly in image and visual recognition problems, which became known as "deep learning" [40].

Ciresan and colleagues (2010) showed that despite the vanishing gradient problem, GPUs make backpropagation feasible for many-layered feedforward neural networks. Between 2009 and 2012, ANNs began winning prizes in ANN contests, approaching human level performance on various tasks, initially in pattern recognition and machine learning. For example, the bi-directional and multi-dimensional long short-term memory (LSTM) of Graves et al. won three competitions in connected handwriting recognition in 2009 without any prior knowledge about the three languages to be learned.

Ciresan and colleagues built the first pattern recognizers to achieve human-competitive/superhuman performance on benchmarks such as traffic sign recognition (IJCNN 2012) [41]

2.3 Introduction to Convolutional Neural Networks (CNN)

One of the most popular deep neural networks is the Convolutional Neural Network (CNN). CNN is a specialized kind of neural network for processing data that has a known grid-like topology. It takes this name from a mathematical linear operation between matrices called convolution. CNN has an excellent performance in machine learning problems; Especially fields dealing with image data, such as computer vision, and natural language processing (NLP).

CNN's are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNN's take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

The most beneficial aspect of CNNs is reducing the number of parameters in ANN. This achievement has prompted both researchers and developers to approach larger models in order to solve complex tasks, which was not possible with classic ANNs. The most important assumption about problems that are solved by CNN should not have features that are spatially dependent. Another important aspect of CNN is to obtain abstract features when input propagates toward the deeper layers. For example, in image classification, the edge might be detected in the first layers, and then the simpler shapes in the second layers, and then the higher-level features [26].

2.3.1 Architecture

CNN has multiple layers;

· The convolutional layer

· Non-linearity layer

· Pooling layer

· Fully Connected layer

2.3.1.1 The Convolutional Layer

When programming a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (input channels). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. The convolution operation brings a solution to the problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks are avoided.

2.3.1.2 Non-linearity layer

A neural network is comprised of layers of nodes and learns to map examples of inputs to outputs. For a given node, the inputs are multiplied by the weights in a node and summed together. This value is referred to as the summed activation of the node. The summed activation is then transformed via an activation function and defines the specific output or “activation” of the node.

The simplest activation function is referred to as the linear activation, where no transform is applied at all. A network comprised of only linear activation functions is very easy to train, but cannot learn complex mapping functions. Nonlinear activation functions are preferred as they allow the nodes to learn more complex structures in the data. Traditionally, two widely used nonlinear activation functions are the sigmoid and hyperbolic tangent activation functions.

2.3.1.3 Pooling layer

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. Pooling may compute a max or an average. Max-pooling is one of the most common types of pooling methods. It partitions the image to sub-region rectangles, and it only returns the maximum value of the inside of that sub-region. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

2.3.1.4 Fully Connected layer

The fully-connected layer is similar to the way that neurons are arranged in a traditional neural network. Therefore, each node in a fully-connected layer is directly connected to every node in both the previous and in the next layer. We can note that each of the nodes in the last frames in the pooling layer is connected as a vector to the first layer from the fully-connected layer. These are the most parameters used with the CNN within these layers, and take a long time in training [26].

The major drawback of a fully-connected layer is that it includes a lot of parameters that need complex computational in training examples. Therefore, we try to eliminate the number of nodes and connections. The removed nodes and connection can be satisfied by using the dropout technique.

2.3.2 CNN Architectures

Convolutional neural networks have been used extensively over the years. During these years, CNN architectures serving different purposes were created. In this section, we examined some of the important CNN architectures.

2.3.2.1 AlexNet

It is the first study in 2012 that made convolutional neural network models and deep learning become popular again. Developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Basically, it is very similar to the LeNet model since there are convolution and pooling layers that follow each other. ReLU (Rectified Linear Unit) is used as the activation function, and max-pooling is used in pooling layers. Approximately 60 million parameters are calculated. It is a breaking point in the image classification problem by providing a sudden increase in classification accuracy from 74.3% to 83.6% in the ImageNet ILSVRC competition.

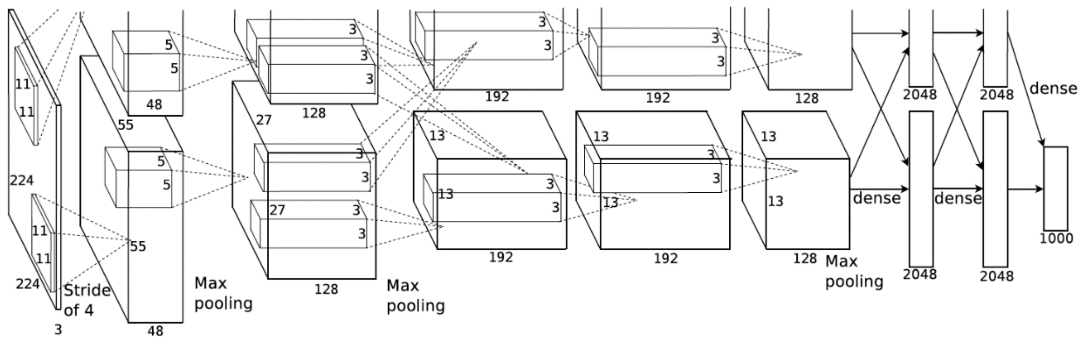


Figure 12 AlexNet Architecture

2.3.2.2 ResNet

ResNet, which has a different logic than its predecessors, where the network model is beginning to deepen; It is formed by adding the residual block to the model, which feeds residual values to the next layers. With this feature, ResNet ceases to be a classic model. In theory, it is thought that the performance will increase as the number of layers in the model increases. However, it has been experienced that this is not the case. Based on this, the ResNet model was created. Thus, when w [1 + 2] = 0, according to the new theory, a [1 + 2] = b [1 + 2]. This (vanishing gradient) is undesirable. However, the residual feed optimizes the new output equation for a [l] value from the two previous layers, even if the current weight is 0. It is trained faster.

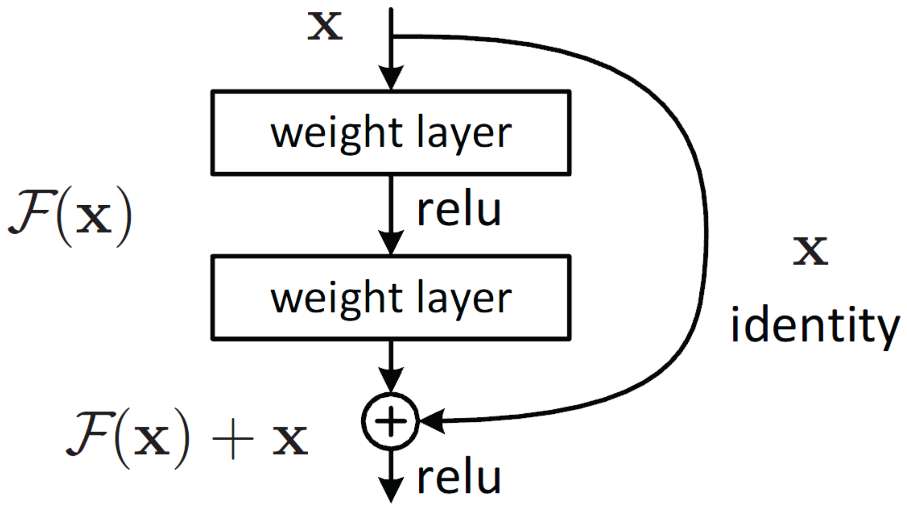

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 13 ResNet Model

2.3.2.3 Vgg-16

In the figure below, all the blue rectangles represent the convolution layers along with the non-linear activation function which is a rectified linear unit (or ReLU). As can be seen from the figure that there are 13 blue and 5 red rectangles i.e there are 13 convolution layers and 5 max-pooling layers. Along with these, there are 3 green rectangles representing 3 fully connected layers. So, the total number of layers having tunable parameters is 16 of which 13 is for convolution layers and 3 for fully connected layers, thus the name is given as VGG-16. At the output, we have a softmax layer having 1000 outputs per image category in the image-net dataset.

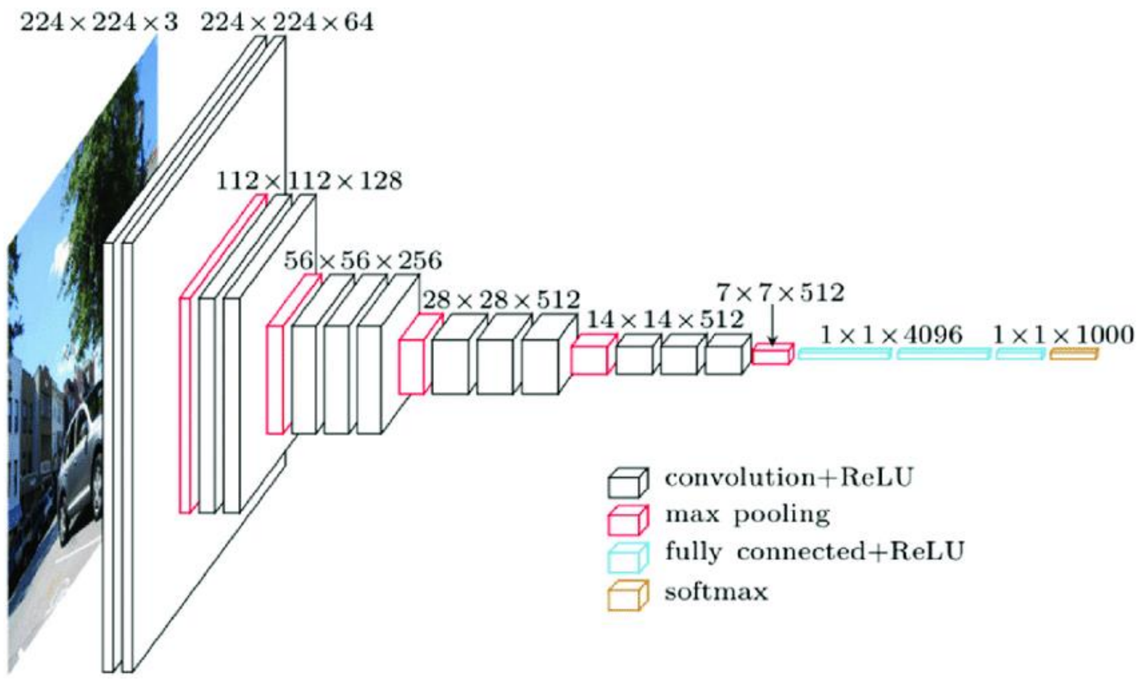


Figure 14 VGG-16 Architecture

2.3.2.4 GoogleNet

Google Net was proposed by research at Google (with the collaboration of various universities) in 2014 in the research paper titled "Going Deeper with Convolutions". This architecture was the winner at the ILSVRC 2014 image classification challenge. It has provided a significant decrease in error rate as compared to previous winners AlexNet (Winner of ILSVRC 2012) and ZF-Net (Winner of ILSVRC 2013) and significantly less error rate than VGG (2014 runner up). This architecture uses techniques such as 1×1 convolutions in the middle of the architecture and global average pooling.

The GoogLeNet architecture is very different from previous state-of-the-art architectures such as AlexNet and ZF-Net. It uses many different kinds of methods such as 1×1 convolution and global average pooling that enables it to create deeper architecture. The inception architecture uses 1×1 convolution in its architecture. These convolutions used to decrease the number of parameters (weights and biases) of the architecture. By reducing the parameters we also increase the depth of the architecture.

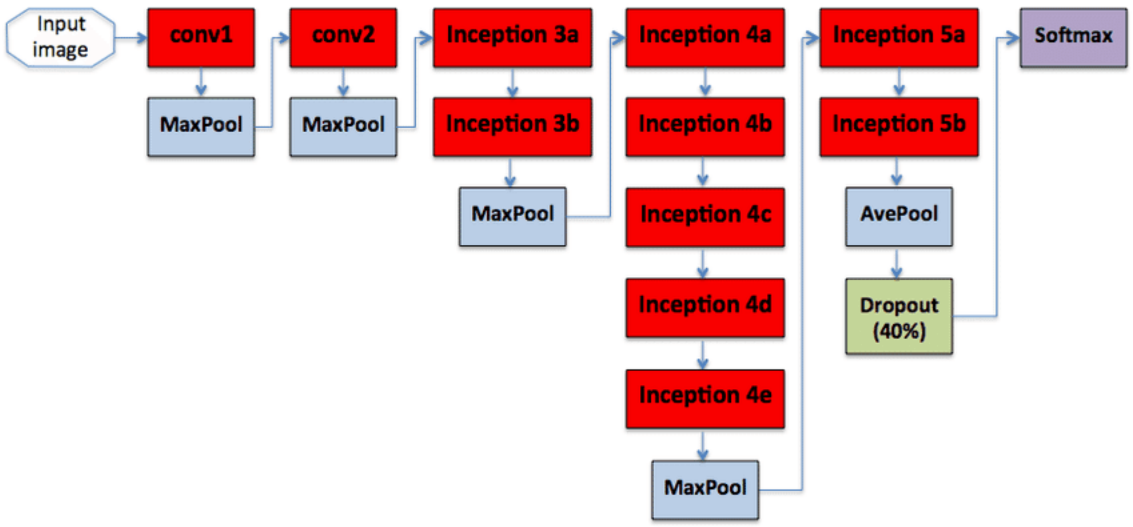


Figure 15 GoogleNet Architecture

2.3.2.5 MobileNet (Depthwise separable convolution)

MobileNet is an efficient and portable CNN architecture that is used in real-world applications. MobileNets primarily use depthwise separable convolutions in place of the standard convolutions used in earlier architectures to build lighter models. MobileNets introduce two new global hyperparameters (width multiplier and resolution multiplier) that allow model developers to trade off latency or accuracy for speed and low size depending on their requirements.

The MobileNet model is based on depth-wise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size. [42]

Depth-wise separable convolution is made up of two layers: depthwise convolutions and pointwise convolutions. We use depthwise convolutions to apply a single filter per each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output of the depthwise layer. MobileNets use both batch norm and ReLU nonlinearities for both layers.

Depthwise convolution is extremely efficient relative to standard convolution. However it only filters input channels, it does not combine them to create new features. So an additional layer that computes a linear combination of the output of depthwise convolution via 1 × 1 convolution is needed in order to generate these new features. The combination of depthwise convolution and 1 × 1 (pointwise) convolution is called depth wise separable convolution.

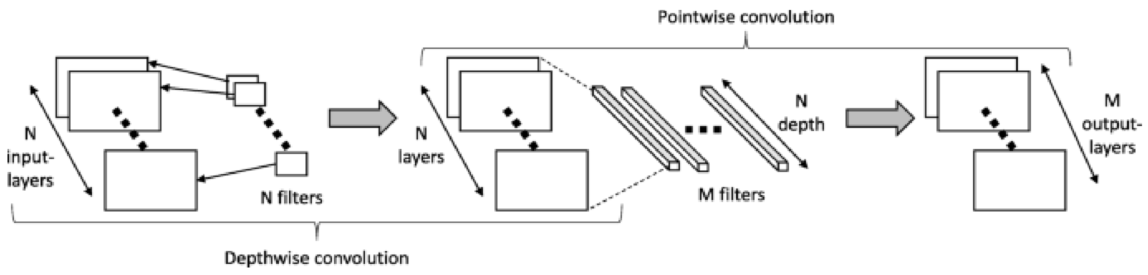


Figure 16 MobileNet Architecture

2.4 Performance Evaluation for Deep Learning

In addition to the performance criteria used in recommendation systems, there are some evaluation criteria specifically applied for Deep Learning. These often give better results and give us a different perspective. We will mention a few of them now.

2.4.1 ROC Curve (Receiver Operating Characteristic Curve)

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{2.18}$$

False Positive Rate (**FPR**) is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{2.19}$$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

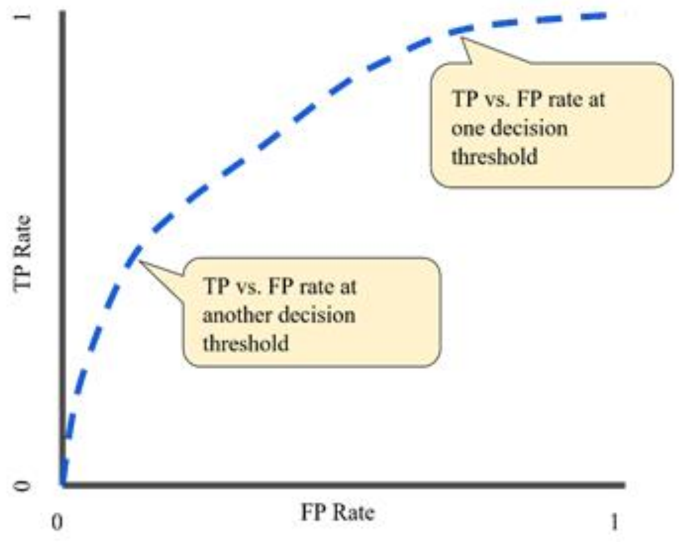


Figure 17 TP vs. FP rate at different classification thresholds

To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

2.4.2 AUC (Area Under the ROC Curve)

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

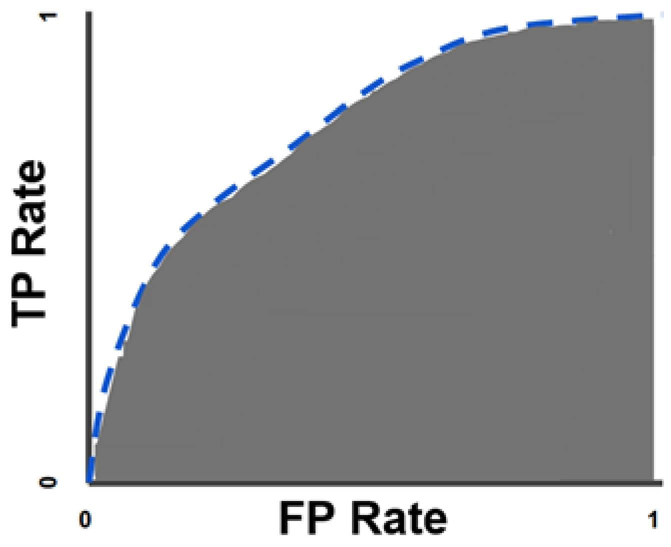


Figure 18 AUC (Area under the ROC Curve)

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

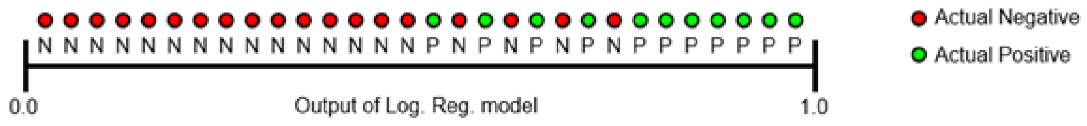

N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

Figure 19 Predictions ranked in ascending order of logistic regression score

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

- AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.

- AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

However, both these reasons come with caveats, which may limit the usefulness of AUC in certain use cases:

- **Scale invariance is not always desirable.** For example, sometimes we really do need well calibrated probability outputs, and AUC won't tell us about that.
- **Classification-threshold invariance is not always desirable.** In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.

In this section, we have made an overview of recommendation systems and artificial neural networks. We emphasized the types of these systems, the methods used in the systems and performance metrics. In the next section, we conducted a literature search for studies using these systems mentioned and interpreted the studies.

CHAPTER 3

LITERATURE REVIEW ON OUTFIT RECOMMENDATION SYSTEMS BASED ON COMPATIBILITY AND DEEP LEARNING

In this section, articles that can contribute to the current study, accessed with the keywords "fashion compatibility", "clothing matching" and "outfit suggestion systems", are examined.

3.1 Evaluation of the Studies in Regard of the Approaches Used

In one of the studies which takes advantage of visual and textual data [9], a multimodal framework which learns fashion compatibility by integrating both semantic and visual embeddings into a unified deep learning model. A multilayered LSTM (Long Short-Term Memory) is employed for discriminative semantic representation and a deep convolutional neural network (CNN) is used for visual embeddings. As a result, a fusion model, which transforms visual and semantic spaces into a style space, is constructed to combine semantic and visual information. Firstly, visual features of the images get extracted with deep Siamese CNNs, then RNNs with multilayered LSTM Module is constructed to obtain semantic features. After training RNN models, a hidden space is constructed which represents the semantic compatibility of fashion items. By linearly embedding textual and visual features, a style space which gives the total information of items by calculating their distance to the chosen (reference) item. In the evaluation chapter of the study, a new loss function which is a modified version of the triplet ranking loss algorithm is presented. Instead of using “bought together” data directly, a threshold value is chosen and based on the number of times pairs are bought together, the compatibility weight is set. As a result, by calculating the distances of reference item (which is selected by the user) to other fashion items, a distance list is obtained and sorted to recommend pairs with minimum distance to the user. The study also assures that the final model can directly be used to calculate the distance for pairs of fashion items offline.

The other study [7] only utilizes the visual data by using deep convolutional neural networks and mixtures of expert systems. The paper claims that it “relaxed the metricity assumption present in recent works by proposing more flexible notions of ‘relatedness’ while maintaining the same levels of speed and scalability”. By not depending on metric learning unlike other studies, it can make cross-category recommendations without any dependence on the explicit category information. Thus, by not limiting the clothes to categories, it prevents the deterioration of the relations between the products; It ensures that the compatibility of products in the same cluster is not ignored and eliminates the need for an external category knowledge. Firstly, visual features are obtained by passing the images through deep CNN. Then, with Mahalanobis Transformation, visual features are projected into N spaces. In this space, the reference item chosen by the user is located in the "anchor space"; Its relationship with the potential item is buried in these N spaces. In this way, the relationship of the reference product with the potential product is transferred to different spaces and examined from all aspects. With the Probabilistic Mixture of Experts method, the distance of each space to the reference space is calculated. The reason why the Mixture of Experts technique is used here is that this method ensures that each space focuses on its own task and is not affected by other spaces while calculating the loss. The result obtained when all the distances are added gives information about the compatibility of the product pair. The paper evaluates the presented recommendation system by using “viewed together” and “bought together” data from Amazon Dataset [43].

One of the studies which only exploits textual information [6] presents a recommendation system using Siamese LSTMs. It aims to find the complement of a product selected by the user using textual descriptors of products such as title sentences. It is pointed out that the current recommendation systems cannot offer two outfits that have never been seen together before, as they are based on product or user history, so these systems have to face the problem of cold start. In the system presented by the study, two products are given as input to two peers (Siamese) LSTMs which are trained with the same parameters and feature vectors are obtained. LSTM (Long Short-Term Memory) helps in storing the order of sequential words in product titles. Then, the relationship of these two products is calculated through the compatibility matrix presented by the study. The result that the matrix returns is only the information that if this pair is compatible or incompatible. Therefore, the "binary-cross entropy" error function is used while training the model. Therefore, the "binary-cross entropy" loss function is used while training the model. While the model is optimized, pairs that are compatible in space but distant from each other or incompatible but close to each other are punished. As a result, the presented model creates a candidate set containing k most compatible products with high probability of compatibility with the given outfit, and suggestions are presented to the user depending on the order of compatibility. At the end of the study, the success presented by comparing the model with the studies using visual or visual-textual data showed that textual information is as effective and important as visual information. "Relatedness" and "also bought" information in the Amazon Dataset were used to evaluate the success of the recommendation.

The framework presented in the study [8] is designed to complement an outfit of users of choice, using only categorized visual data. For this, firstly, feature transformation was applied to the visuals by using Siamese CNNs and the visuals were placed in "style space". While training the CNNs, the reference product and the product pair, categories and relationships it is compatible with are sampled. As mentioned in the study, using CNN is very effective in extracting visual features; Understands the relationship between images of the same object obtained from different locations or angles. However, in the article, it was mentioned that similar products inevitably converge during the training of CNNs, and this situation may cause false inferences (such as proposing two products in the same category) in the style space. In addition, the study aims to make suggestions for previously unseen categories. For all these reasons, the study offers three sampling techniques: Naïve Sampling, Strategic Sampling, and Holdout-categories Sampling. Training pairs were randomly selected with Naïve sampling; pressure was created not to bring pairs of products from the same categories with strategic sampling; With holdout-categories, feature transfer is provided by keeping data from categories not used in training in the test set. As a result, the system that is ready for use works as follows: First, the KMeans algorithm is used on the category where the product is selected by the user, and the items are clustered and the closest center point to the selected item is determined. Then, the five items closest to this center point are selected and recommendations are presented to the user.

Amazon.com's "also bought" and "bought together" data were used to measure the success of the recommendation system. However, as stated in this study, these data are based on Amazon's product-based recommendation system and user behavior; It cannot give precise information about the compatibility of the products. For this reason, apart from these data used, user study was conducted, and users were shown one product and two complements and asked which one they would prefer. User study has reached some determinations that may be useful in the present study: First, users consider functionality as well as compatibility when choosing complementary products. As an example, it has been shown in the study that the users choose incompatible long socks and boots instead of matching short socks and boots. Secondly, it has been observed that the users do not always find the products whose style is most similar. Thirdly, when users choose the complementary product, sometimes it is not the compatibility but which product they like more. The paper plans to focus on these observations and personalized recommendation systems in future studies on this topic.

The study [43], unlike most of the studies in the literature, aimed to take into account the annual fashion trend with compatibility, and by combining trend information obtained from experts, visual and visual contents, user behaviors and product descriptions, it revealed multidimensional information. The presented system is divided into two parts: convolutional neural networks, where the features of the product images and descriptions are extracted, and the hierarchical collocation model (HCM), which is established to embed these features into the style concept and to interpret them with high level semantic information. By presenting these two parts, the study tried to take into account "visual aesthetics" and "collocation experience", which are effective in users' decision to complete their clothes. The study has distinguished itself from other studies on this subject by not only considering the visuality, but also using features such as clothing style, color, texture and fabric in the recommendation system, making use of textual content and interpreting the features with high level semantic information.

Two Siamese CNN models are created for upper-body and lower-body inputs and trained to predict the clothing characteristics of the outputs. The outputs of these models for textual and visual data are converted into bag of words (BOW). After this stage, the two-stage HCM model is passed. In the first stage, outfit features are refined by adapting the Hybrid Topic Model (HT). The model learns a group of "style topics" that are the distribution of visual and semantic codewords in the dataset, and the clothes are treated like a product that is a mixture of style topics; These subjects are represented by a feature vector to represent each outfit. In the second step, Collocation Topic Model (CT) is adapted. Since each category is viewed as a pattern in this model, the patterns are weighted, and matching scores are calculated; Predictions are trained by Linear Regression method. The main reason for the application of these processes is to narrow the semantic gap between visual and textual features, to interpret the relationship between style elements and to create a proposal model that can make suggestions compatible with the fashion trend by embedding old information.

In order to implement the fashion trend into the suggestion system, the study presents its own strategy with the "fashion tracker" module. First, expert advice phrases on sites such as Vogue.com and Yoka.com are transformed into feature words, and the pairs of clothes that were previously compatible in the system are compared. Couples that overlap with expert opinions are considered directly compatible, while couples that do not in any way overlap with the characteristics of the recommendation are considered incompatible. If an outfit within the couple matches the expert opinion, recalculation is made until the appropriate complement for that outfit is found. Thus, the trend is caught. The matching scores are calculated by multiplying the couples that meet the desired condition with the h parameter known as "fashion sensitivity", and the couples that do not meet it by 1 / h. The higher this value, the greater the weight of the trends in the system (h was determined as 1.5 in the study). This approach is advantageous as it does not rely on additional information such as user buying behavior or photos of clothing.

Another study [44] brings a different approach to the general feature extraction techniques for the image data. The clothing images are passed through the ResNet50

[45] model, and the feature matrices in the layers "conv2_x", "conv3_x", "conv4_x" and "conv5_x" are extracted. Then, the properties were reduced as vectors and got rid of from spatial information such as location with the Global Average Pooling (GAP) method. Total compatibility scores were obtained by comparing the clothes vectors whose features were extracted with the vectors at their level with the Multi-Layered Comparison Network (MCN) presented by the study. Backpropagation gradients were used to estimate how much each pair of clothing feature vectors affected the mismatch in the MCN. However, while calculating the compatibility score, the feature vectors were subjected to element-wise multiplication with a learnable mask and reflected to a new space with the ReLU activation function. Then, the total compatibility scores were calculated by taking the cosine distances of the features. Linear regression model was trained using these scores. In addition, the textual descriptions of the images were brought to the BOW format, by transferring the visual features to the same space, ensuring that the descriptions matching the visual were as close to each other in space as possible.

As a result, the article can perform two tasks: To calculate the compatibility scores of the given clothes or to find the complementary product for these clothes. While finding the complementary product, the model calculated the compatibility scores of the combinations using the four options given in the experiment and recommended the item of clothing with the highest score.

In addition to the mentioned study, [46] also contributed to the system proposed by the study. Here, with the pre-trained GoogleNet's InceptionV3 [47] model, 512-D feature vectors of the visuals have been extracted and passed through the Bidirectional LSTM (Bi-LSTM). Bi-LSTMs are made able to predict the complement of a given outfit by taking compatible and incompatible dress sequences. In this respect, it can be said that the study treats the clothes as if they are words that follow one after another in a sentence. Apart from this, just like in [44], textual explanations of the clothes are also used. First, the clothes filtered according to the frequency of occurrence in the descriptions were turned into a dictionary, One Hot Encoding method was applied and the words were converted to Bag of Words (BOW) format; The dictionary created in the study has 2757 words. Then, the visual and textual features were multiplied by weights and transferred to the 512-dimension common space, and it was aimed to have the matching visual-explanations close to each other depending on the cosine distance. The system presented in the study can perform the tasks of estimating the fit score, filling-in-the-blank and generating clothes.

One of the recent studies differ [10] from others in the way of modelling compatibility between items. It criticizes other studies trying to model compatibility by projecting clothes into a single space alone, as it argues that this leads to a false assumption that similar products are compatible or the other way round. However, this approach also leads to the wrong notion that there is a transition between clothes. For example, if a hat is compatible with a t-shirt and a shoe is compatible with that shirt, it does not mean that the hat is compatible with that shoe, but the models mentioned may make the mistake of not paying attention to this.

As a result, the study identified a common space for the similarities of clothes and subspaces for the compatibility of pairs of clothes. First of all, the clothes, whose characteristics are extracted through CNNs and reflected to the similarity space, have been trained to be close in space with their textual expressions. The vectors of the texts in the space were formed as a result of the product titles being reduced to 6000 dimensions with the PCA technique and then encoded into the HGLMM Fisher vector. While considering the clothes in terms of compatibility, spaces consisting of the double combinations of outfits were created and their proximity/distances in this space were calculated. For this, while the model was being trained, product triplets consisting of a reference (anchor), a positive (compatible) and a negative (incompatible) clothings were given as input, and when calculating the compability scores, compatibility was represented by 1 and incompatibility as 0. The loss functions were created separately for similarity, compatibility, space sparsity and L2 regularization, and then the total loss was calculated.

The data used in the study, on the other hand, was obtained from the products in Polyvore, a social trade site where many outfits are shared. While separating the clothes for training and testing, the graph segmentation algorithm was used to prevent a product seen in training from being seen in the test.

At the end of the study, the model can assign a compatibility score for the clothing sets and even complete a missing piece in a given combination by choosing one of the options (Fill in the blank).

3.2 Evaluation of the Literature Review

Most of the studies on complementary clothing recommendation systems rely on visual data, as the aesthetic perception in fashion is based on visuality. However, some studies [6] [10] have shown that textual data are also quite effective in recommendation systems. Also, combining outfits is complicated, as it requires a lot of parameters; Therefore, studies generally focus on offering complementary clothing to a given (reference) clothing.

The main shortcomings of these studies are that they have almost no real applications, the variety of the datasets used is low and the success criteria are based on fixed assumptions. Many studies, as mentioned before, use image datasets from Amazon [48]. Although these datasets are large, the systems to be created may have difficulties in grasping the general aesthetics because they consist of data from a single source. However, the success rates achieved are controversial, as some studies deem "buy together" or "look together" data sufficient for compatibility.

As a result, it can be said that there has been a lot of work on clothing recommendation systems, especially in recent years. Although these studies presented their solutions in different ways, the purpose of all of them tried to catch the aesthetic perception and the concept of compatibility in fashion. It is also seen that academic studies have achieved different results depending on their way of handling the problem; This has shown us how new this field is in fact and contains a lot of topics that need to be discovered.

In this part of the study, various studies selected from the literature are discussed and summarized from many different perspectives. In addition to this, the deficiencies identified during the literature review, which are the motivation source for this study, are also emphasized in this section. In the next part of the study, the conceptual design and steps of the recommendation system will be explained, as well as the techniques used.

CHAPTER 4

CONCEPTUAL DESIGN OF THE PROPOSED SYSTEM

In this section, the conceptual design of the model we developed for the complementary clothing recommendation system will be discussed. The modeling understanding that we have included for the solution of the problem has been created by taking examples from [10] [46] studies.

As seen in the literature research, basically, whether two clothes are compatible or not can be explained by two variables: visual and textual data. Considering the visual data set is expressed as X and the textual data set is expressed as T; $(x_i^{(m)}, x_j^{(n)}, x_k^{(n)})$ represents the triple item sets : $x_i^{(m)}$ being considered to be the anchor item, $x_j^{(n)}$ and $x_k^{(n)}$ the positive and negative complementary items respectively. Assuming we have a taxonomy of I types, when the data is projected into general space, we obtain two feature vectors:

$$f\left(x_i^{(m)}; \theta\right) = y_i^{(m)}, y_i^{(m)} \epsilon\, \mathbb{R}^d, m \,\epsilon\, I \tag{4.1}$$

$$h\left(t_i^{(m)}; \beta\right) = v_i^{(m)}, v_i^{(m)} \epsilon\, \mathbb{R}^d, m \,\epsilon\, I \tag{4.2}$$

$y_i^{(m)}$ represents the visual embedding of the i^{th} item from the m^{th} category, while $v_i^{(m)}$ represents the item's textual feature vector. θ and β are included as parameters of the functions. However, it should be noted that complementary products of similar products from the same category may not be the same; So there is no transitivity. Therefore, the visual feature vectors obtained in the general space should be moved to a subspace with the complementary products they pass together so that the concepts of similarity and compatibility can be handled separately. The embeddings obtained as a result of projection, with $P^{(.)\rightarrow(.,.)}$ being the projecting function:

$$P^{(m)\to(m,n)}\left(y_i, w^{(m,n)}\right) = y_i^{(m)} \odot w^{(m,n)} = z_i^{(m)}, m, n \epsilon I \qquad (4.3)$$

Here, m is the category of the current item and n represents the category of another item. Feature vectors are subject to element-based multiplication with the $w^{(m,n)}$ mask in order to be reflected in the space of these two categories. The $w^{(m,n)}$ mask is a vector that is not fixed and can be learned over time. However, the same mask values are used when transferring products from two different categories to the shared sub-space:

$$P^{(m)\to(m,n)} = P^{(n)\to(m,n)} = diag(w^{(mn)}) \qquad (4.4)$$

Mask W can be expressed as $W \in \mathbb{R}^{p\times d}$, the number of different types of clothing pairs p, and the size of the embeddings d. The compatibility of the clothes with each other is calculated with the pairwise distance of the embeddings in the type space. It is assumed that the greater this distance, the more incongruous the clothes, and the closer they are, the more compatible they are. The compatibility between two clothes from categories m and n is expressed as

$$d_{ij}^{(mn)} = \left\| z_i^{(m)} - z_J^{(n)} \right\|_2^2 = \left\| y_i^{(m)} \odot w^{(m,n)} + y_j^{(n)} \odot w^{(m,n)} \right\|_2^2 \qquad (4.5)$$

The resulting compatibility loss is a modified version of Triplet Margin Loss:

$$\mathcal{L}_{comp}\left(y_i^{(m)}, y_j^{(n)}, y_k^{(n)}; \theta\right) = max\left(0, \left(d_{ij}^{(mn)} - d_{ik}^{(mn)}\right) + \mu\right) \qquad (4.6)$$

μ, representing the margin loss.

In the general space, the similarities of the clothes are calculated and compared. The resulting similarity loss is expressed as

$$\mathcal{L}_{sim} = \lambda_1 \mathcal{L}\left(y_j^{(n)}, y_k^{(n)}, y_i^{(m)}\right) + \lambda_2 \mathcal{L}\left(v_j^{(n)}, v_k^{(n)}, v_i^{(m)}\right) \qquad (4.7)$$

λ_{1-2} are scalar parameters which are fixed, and they determine the weight of the loss of image-image similarity and text-text similarity. The items from the same category are assumed as similar and vice versa. $\mathcal{L}$ function is the Triplet Margin Loss function which is

$$\ell(i,j,k) = max\big(0, (d_{ij} - d_{ik}) + \mu\big) \tag{4.8}$$

and distance d is calculated using pairwise distance function

$$d_{ij} = \|y_i - y_j\|_2^2 \tag{4.9}$$

There is also visual-semantic loss which is the calculation of the distance between image and text of the same item and different items. The visual-semantic loss of i^{th} item is defined as:

$$\mathcal{L}_{vse_i} = \mathcal{L}\left(y_i^{(m)}, v_i^{(m)}, v_j^{(n)}\right) + \mathcal{L}\left(y_i^{(m)}, v_i^{(m)}, v_k^{(n)}\right) \tag{4.10}$$

$$\mathcal{L}_{vse} = \sum_{t=i}^{S} \mathcal{L}_{vse_t}, S = \{i,j,k\} \tag{4.11}$$

Consequently, the total loss of the model is presented as

$$\mathcal{L}\big(X, T, P^{(.)\to(.,.)}, \lambda, \theta, \beta\,\big) = \mathcal{L}_{comp} + \lambda_3\mathcal{L}_{vse} + \lambda_4\mathcal{L}_{\ell_1} + \lambda_5\mathcal{L}_{\ell_2} \tag{4.12}$$

λ_{3-5} are scalar parameters; $\mathcal{L}_{\ell_1}$ and $\mathcal{L}_{\ell_2}$ are loss of ℓ_1 and ℓ_2 regularization respectively. ℓ_1 regularization is used to encourage type sub-spaces to become more sparse which means the data in dual type space can accumulate on a particular region. However, ℓ_2 regularization is used to obtain image embedding y_i.

The model is required to offer a complementary clothing product such as fill-in-the-blank. The complement of the i^{th} outfit from the m^{th} category can be calculated as:

$$argmin\left(\left[\left\|z_i^{(m,n_1)} - z_1^{(m,n_1)}\right\|_2^2, \left\|z_i^{(m,n_2)} - z_2^{(m,n_2)}\right\|_2^2, \dots, \left\|z_i^{(m,n_p)} - z_p^{(m,n_p)}\right\|_2^2\right]\right) \tag{4.13}$$

$$z_i^{n_k} = y_i^{n_k} \odot w_i^{(m,n_k)} \tag{4.14}$$

$$Z^{(m,n)} \in \mathbb{R}^{p\times d}, Z^{(m,n)} = \left\{z_1^{(mn_1)}, z_2^{(m,n_2)}, \ldots, z_p^{(m,n_p)}\right\} \tag{4.15}$$

Here, with $n = \{n_1, n_2, \ldots, n_p\}, n \in I \ and \ m \notin n, n$ represents the clothing categories and p represents the number of clothing compared.

In this section, the conceptual design of the proposed complementary clothing recommendation system is explained in detail. For "Implementation" section, analysis and processing of the data required for the system; The technical methods and parameters determined during the preparation of the system models will be included.

CHAPTER 5

IMPLEMENTATION OF THE CONCEPTUAL SYSTEM DESIGNED

5.1 Data Used in the System

In this part of the research, we will give detailed explanation about how we analyzed the data set we took as a sample and arrange it for our own study.

We determined the categories of clothing data that our study that we need from the Polyvore dataset [10], which contains 205,959 clothing data. We have divided the clothing categories we need in the clothing set into two parts as top and bottom clothes categories. We can classify top and bottom clothes in themselves as follows:

Table 4 Top Clothes of Data

Category ID	Top Clothes	Number
11	Tops	9149
17	Blouses	4956
15	Tunics	386
19	Sweaters	5750
21	T-Shirts	4893
343	Men's Tank Tops	12
252	Activewear Tops	98
272	Men's Shirts	14
273	Men's Sweaters	25
275	Men's T-Shirts	148
286	Men's Activewear Tops	6
309	Activewear Tank Tops	129
342	Men's Polos	11
4454	Men's Casual Shirts	102
4496	Hoodies	722
341	Men's Dress Shirts	17
	TOTAL TOPS	26418

Table 5 Bottom Clothes of Data

Category ID	Bottom Clothes	Number
7	Skirts	778
8	Mini Skirts	2429
9	Knee Length Skirts	5534
10	Long Skirts	749
27	Jeans	1565
28	Pants	4141
29	Shorts	3963
237	Skinny Jeans	3094
238	Bootcut Jeans	174
239	Wide Leg Jeans	126
240	Boyfriend Jeans	750
241	Leggings	914
251	Tights	318
253	Activewear Pants	633
254	Activewear Skirts	17
255	Activewear Shorts	169
278	Men's Jeans	137
279	Men's Pants	2
280	Men's Shorts	27
287	Men's Activewear Pants	19
288	Men's Activewear Shorts	18
310	Straight Leg Jeans	342
4452	Flared Jeans	312
4454	Men's Casual Pants	102
4459	Men's Dress Pants	13
332	Capri & Cropped Pants	1136
	TOTAL BOTTOMS	27462

After categorizing the clothes set we have as top and bottom clothes, we transferred the clothes in the determined categories to a new file with the help of Python programming language using JSON and NumPy libraries. At the end of the transaction we had 26418 top and 27462 bottom clothes.

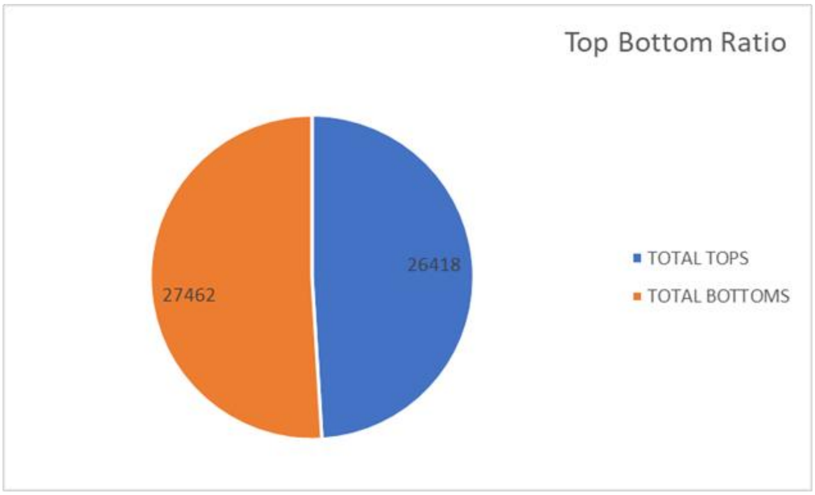


Figure 20 Top Bottom Ratio

The number inequality between the top and bottom clothes showed us that we have used some products more than once in the combinations we used.

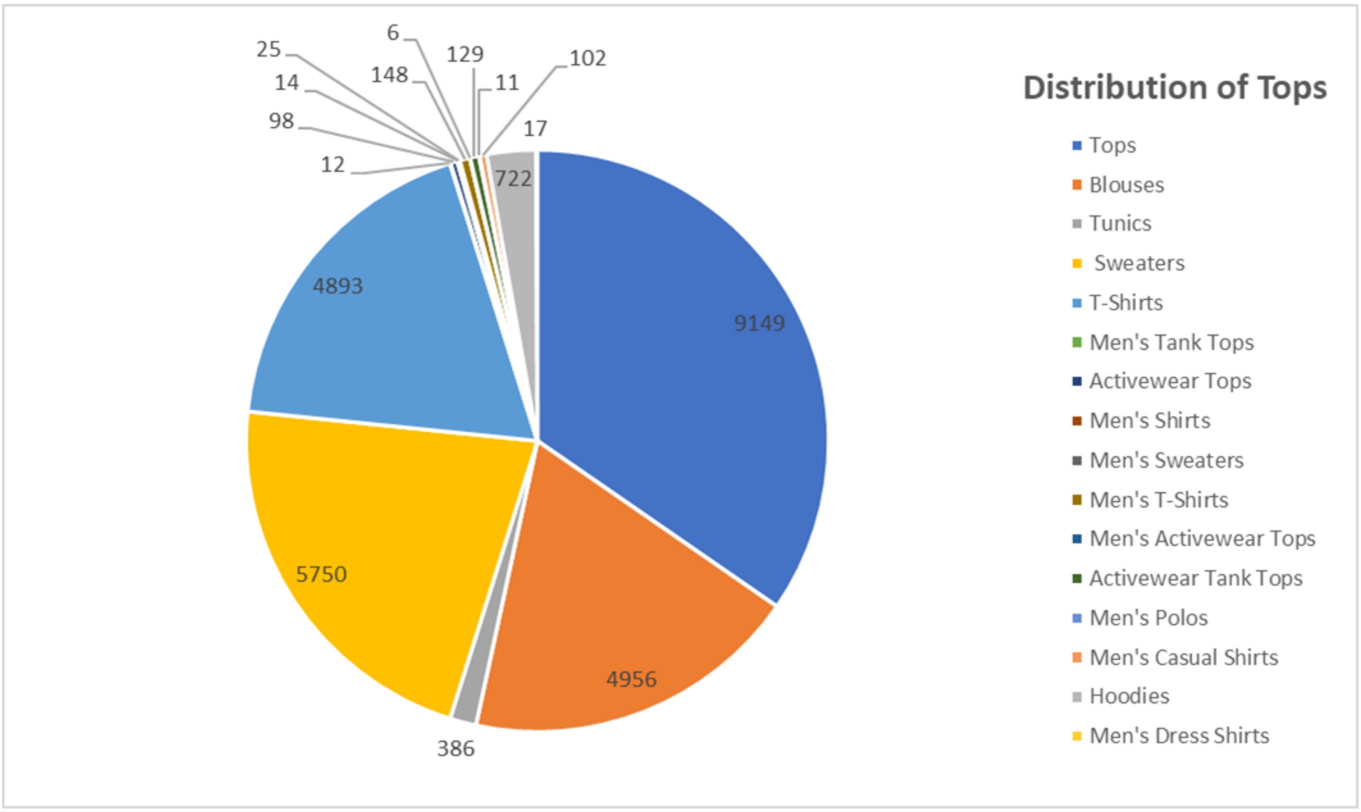


Figure 21 Distribution of Tops

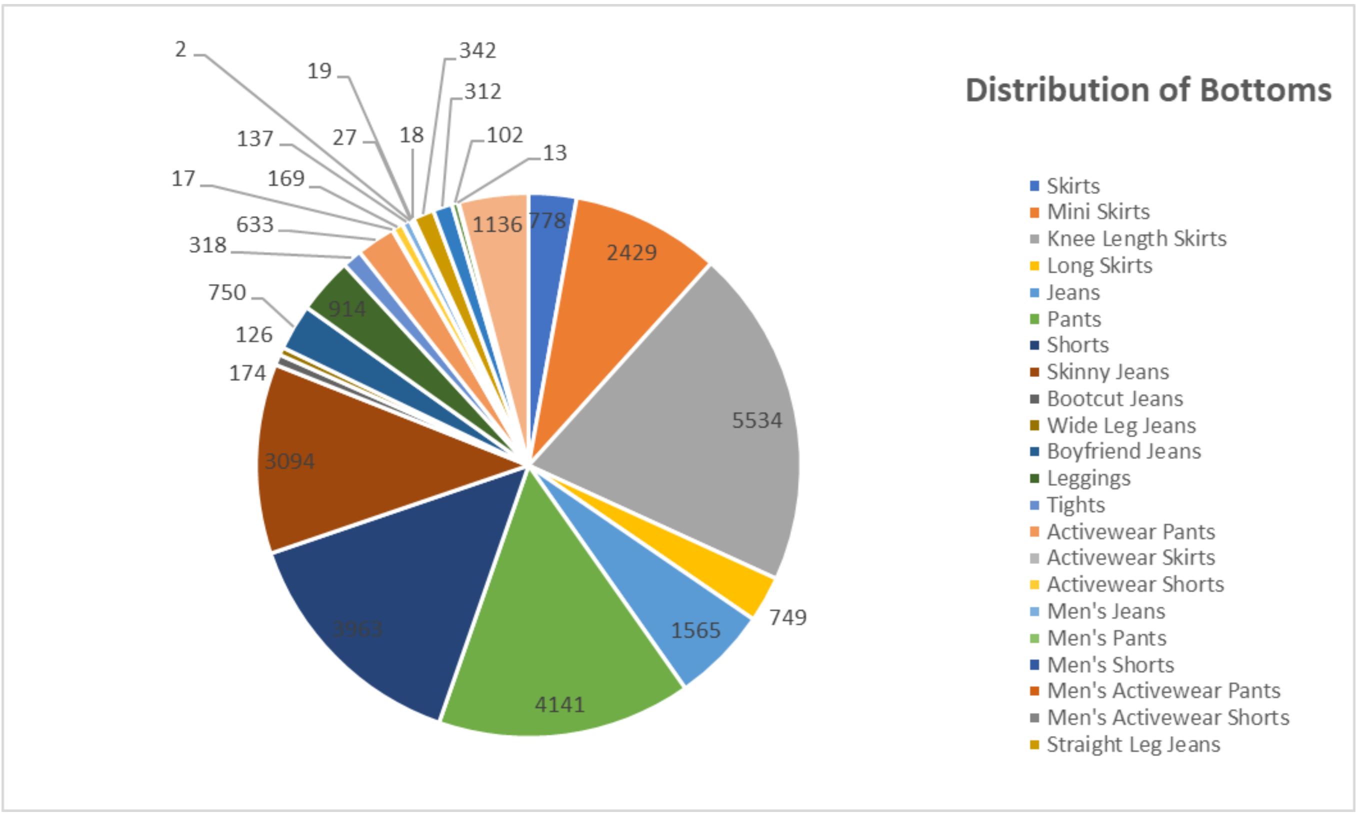


Figure 22 Distribution of Bottoms

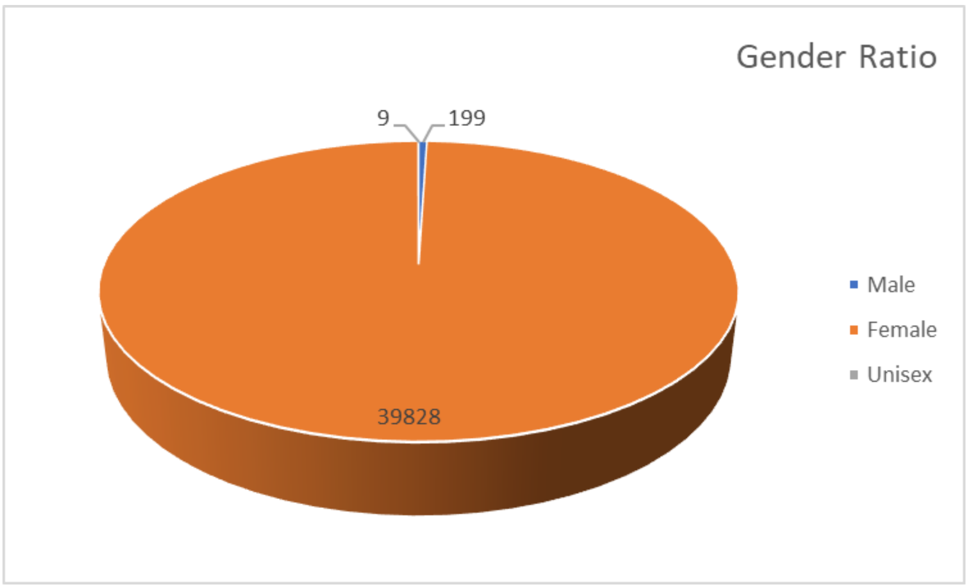


Figure 23 Gender Ratio

Table 6 Gender Distribution of Clothes

Gender	Numbers
Male	199
Female	39828
Unisex	9

Since the data contains a lot of women's clothing, it may cause the model to better grasp the female fashion sense. In addition to this, it may be possible to create a more genderless and valid fashion sense for the model by increasing the men's and unisex clothing data, as a situation that can be improved.

After dividing our clothes into the desired categories, we examined the combinations used in the study and turned the ready-made combination pairs containing at least one top and bottom into one top and one outfit with indexes 1 and 2, and we obtained our own clothing combination files. In this process, we have obtained the combinations we need by comparing the top and bottom clothing file that we have separated with desired categories so that it will be useful for us, with the combination file currently used in the data sample. We have used the same process for the test, train, and validation files in order to make them useful for our model.

Table 7 Number of Clothes Before Arrangement

Files	Pairs
Train	53306
Test	10000
Validation	5000

Table 8 Number of Clothes After Arrangement

Files	Pairs
Train	25685
Test	4691
Validation	2348

The clothing data in our hands have changed as shown in the table, as we have adapted the clothing dataset to only the top and bottom clothing categories. In addition, we made sure to have a photo of every outfit information we retrieved from the dataset, and deleted the data that had no photos.

If we need to examine the types of data in these files more closely, they can be presented as follows:

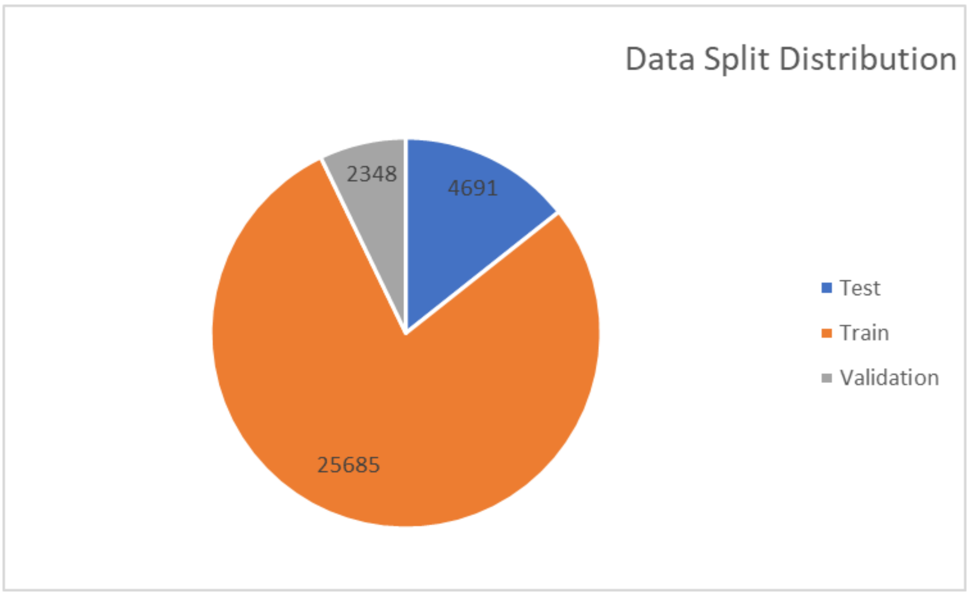


Figure 24 Data Split Distribution

The color of data is distributed as in the graph:

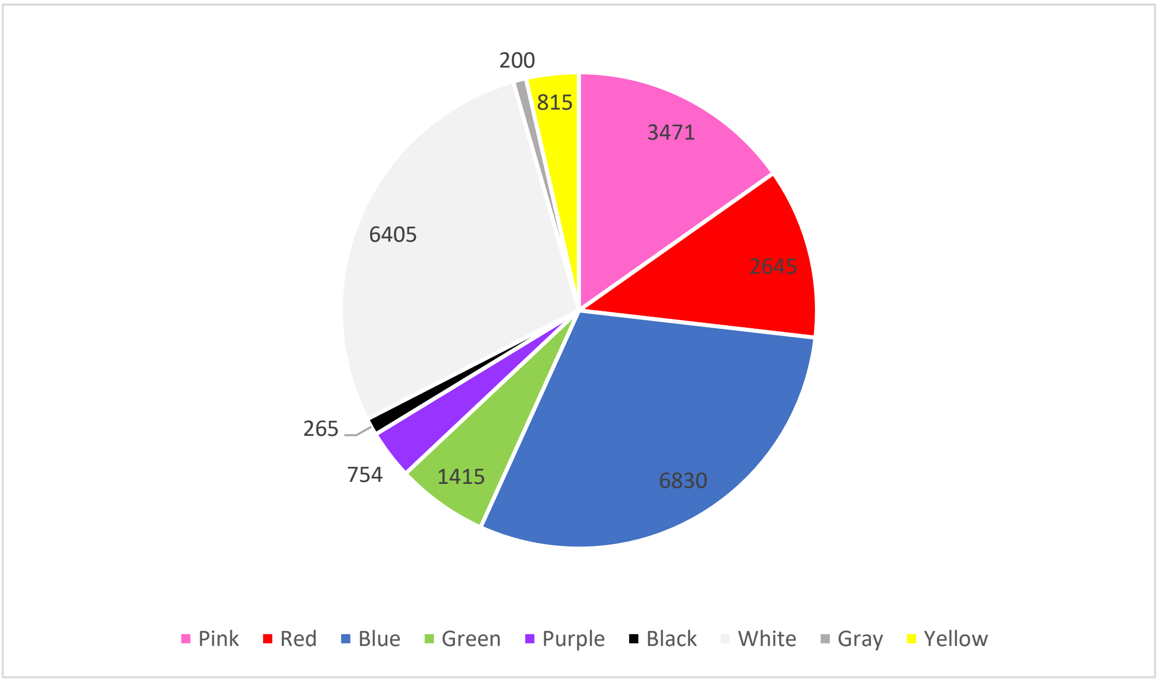


Figure 25 Color Data of Clothes

The data in our test, train and validation file is distributed as in the graph:

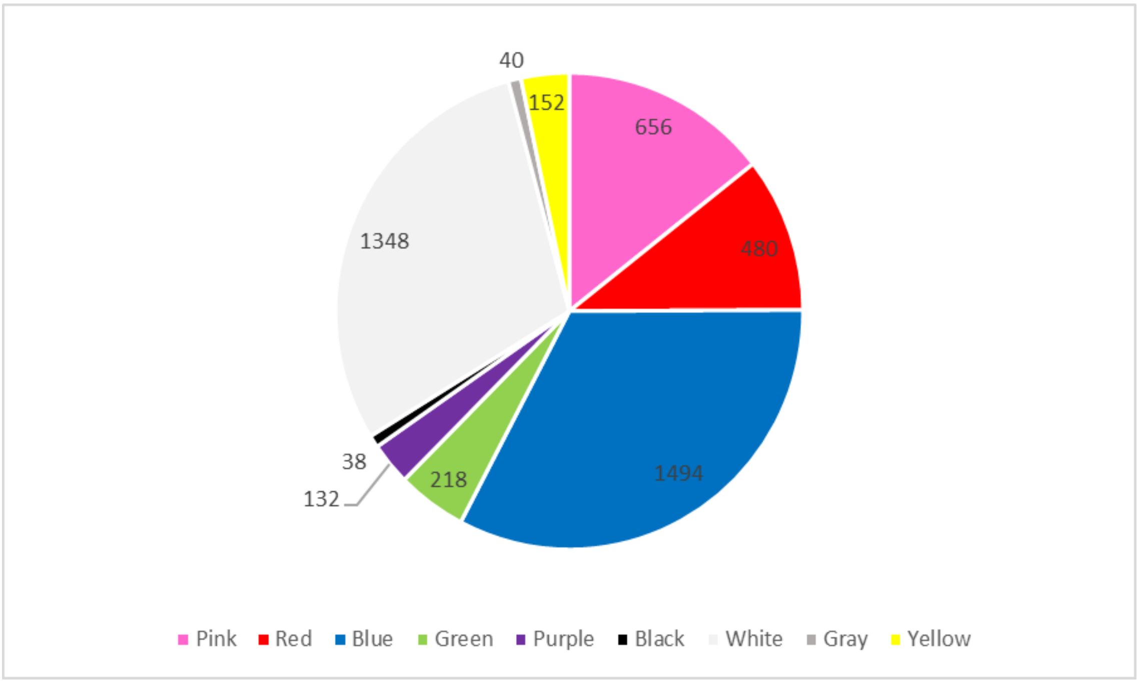


Figure 27 Color Data of Test File

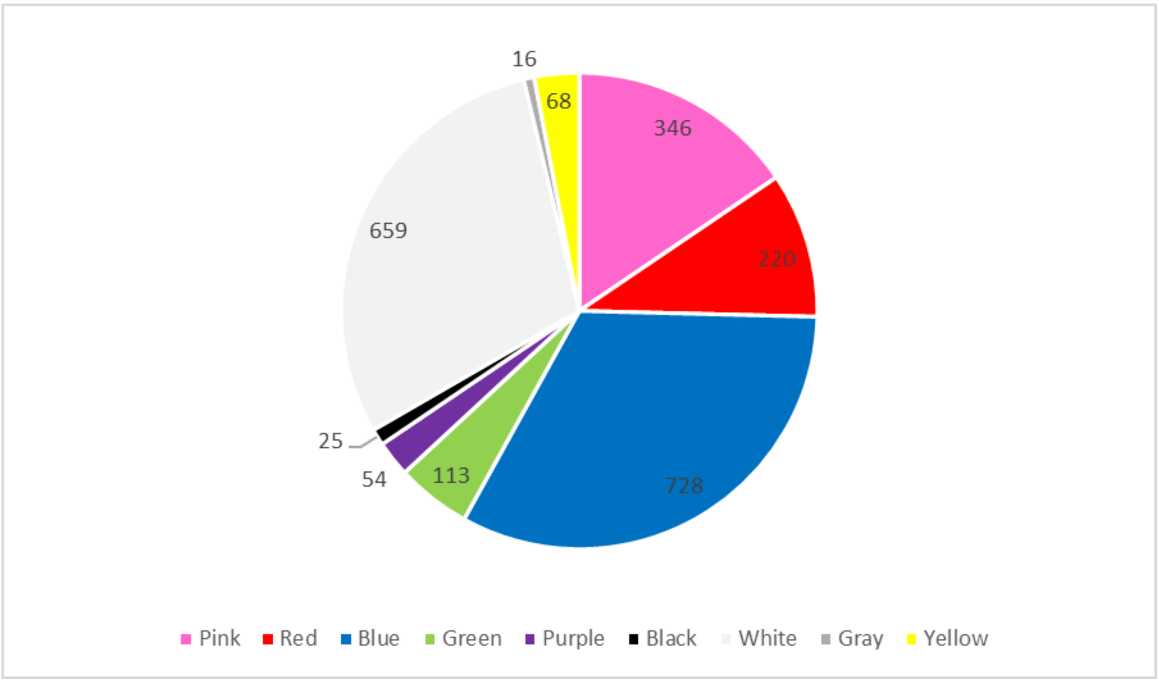


Figure 26 Color Data of Train File

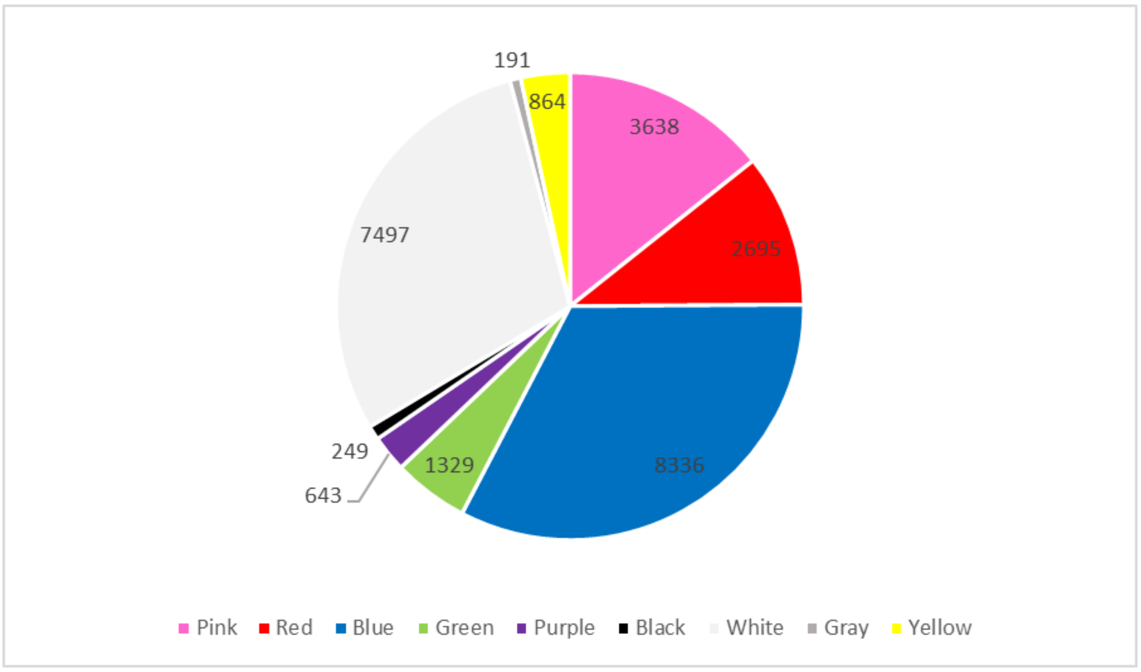


Figure 28 Color Data of Validation File

In order to understand the perception of fashion and compatibility and to make the desired suggestions, the brands that are taken as an example in the data are also of great importance. Therefore, the brand distributions of the photographs we use are distributed as shown in the tables.

Table 9 Brand Diversity of our Test Data (Tops)

TOPS			
Type	**Number**	**Type**	**Number**
Gucci sweaters	10	Topshop t-shirts	18
Chloé blouses	10	Chicwish tops	19
Diane Von Furstenberg sweaters	10	Acne Studios sweaters	20
Chicwish blouses	10	Topshop shorts	20
Burberry sweaters	10	MANGO sweaters	21
Topshop blouses	10	MANGO blouses	21
J.Crew tops	10	WearAll tops	23
Chloé sweaters	11	Alice + Olivia tops	24
WithChic sweaters	11	H&M tops	26
Dolce&Gabbana sweaters	11	MANGO tops	29
Proenza Schouler sweaters	11	Tank Tops	30
Chicwish sweaters	11	River Island tops	32
Hollister Co. Tops	12	Boohoo tops	34
Zimmermann tops	12	Topshop tops	60
Gucci blouses	12	Hoodies	69
Dolce&Gabbana tops	12	Sweatshirts	123
Alexander McQueen blouses	13	Sweatshirts & Hoodies	192
J.Crew sweaters	13	T-Shirts	636
H&M blouses	14	Blouses	746
Alexander Wang t-shirts	16	Sweaters	796
Men's Shirts	16	Tops	3641
TIBI tops	16	Clothing	4132
Men's Clothing	17	Women's Fashion	7644
Tunics	17		

Table 10 Brand Diversity of our Test Data (Bottoms)

BOTTOMS			
Type	**Number**	**Type**	**Number**
Marni skirts	10	J.Crew jeans	20
Dsquared2 jeans	10	Topshop mini skirts	21
MANGO skirts	10	River Island shorts	21
Valentino shorts	10	River Island mini skirts	22
Miss Selfridge jeans	10	Citizens of Humanity jeans	23
J.Crew shorts	10	AG Adriano Goldschmied jeans	24
Emilio Pucci pants	11	rag & bone jeans	24
Frame shorts	11	Levi's jeans	24
Roland Mouret skirts	11	Bootcut Jeans	25
Balmain jeans	12	7 For All Mankind jeans	28
H&M shorts	12	Chicwish skirts	29
Alexander Wang jeans	12	Current/Elliott jeans	31
MANGO shorts	12	Paige Denim jeans	33
True Religion jeans	12	J Brand jeans	36
Activewear Shorts	12	Flared Jeans	45
Hollister Co. Jeans	13	Frame jeans	46
Dolce&Gabbana skirts	13	Activewear Pants	52
Yves Saint Laurent jeans	13	Straight Leg Jeans	57
Men's Jeans	14	Topshop jeans	81
MSGM skirts	14	Long Skirts	88
Abercrombie & Fitch jeans	14	Leggings	108
River Island skirts	14	Capri & Cropped Pants	126
STELLA McCARTNEY jeans	14	Boyfriend Jeans	176
Miss Selfridge skirts	15	Mini Skirts	370
Wide Leg Jeans	15	Shorts	578
Hollister Co. Shorts	15	Skinny Jeans	607
River Island jeans	16	Knee Length Skirts	693
Levi's shorts	16	Pants	773
Topshop skirts	17	Jeans	1217
Topshop pants	17	Skirts	1249

Table 11 Brand Diversity of our Train Data (Tops)

TOPS			
Type	**Number**	**Type**	**Number**
Sans Souci tops	50	J.Crew tops	88
Burberry sweaters	50	Topshop t-shirts	90
MSGM tops	51	Tunics	99
Glamorous tops	52	WearAll top	100
Proenza Schouler sweaters	52	T By Alexander Wang tops	108
Hollister Co. Tops	55	MANGO t-shirts	114
The Row sweaters	56	MANGO blouses	121
WithChic blouses	58	MANGO sweaters	125
Valentino sweaters	61	H&M tops	128
H&M sweaters	64	MANGO tops	128
Men's Shirts	66	Miss Selfridge tops	130
WithChic t-shirts	68	Tank Tops	140
Chloé sweaters	70	Alice + Olivia tops	156
J.Crew sweaters	71	Boohoo tops	158
Monki t-shirts	73	River Island tops	166
T By Alexander Wang t-shirts	74	Topshop tops	251
Gucci sweaters	74	Hoodies	400
TIBI tops	75	Activewear	407
Dolce&Gabbana tops	83	Sweatshirts	795
Men's Jeans	83	Sweatshirts & Hoodies	1195
Chloé blouses	84	T-Shirts	3533
Topshop sweaters	84	Blouses	4230
H&M blouses	85	Sweaters	4722
Chicwish tops	85	Tops	20637
Acne Studios sweaters	87		

Table 12 Brand Diversity of our Train Data (Bottoms)

BOTTOMS			
Type	**Number**	**Type**	**Number**
STELLA McCARTNEY pants	50	River Island jeans	116
Cheap Monday jeans	52	Men's Clothing	116
American Eagle Outfitters jeans	53	7 For All Mankind jeans	122
True Religion jeans	53	rag & bone jeans	122
MSGM skirts	55	Levi's jeans	122
rag & bone shorts	56	Citizens of Humanity jeans	124
River Island mini skirts	57	Topshop shorts	131
Boohoo shorts	57	Topshop skirts	136
Balmain mini skirts	58	AG Adriano Goldschmied jeans	138
WithChic skirts	59	Chicwish skirts	163
Roland Mouret skirts	60	Paige Denim jeans	170
Topshop pants	61	Current/Elliott jeans	185
H&M shorts	63	Men's Fashion	186
Activewear Shorts	65	J Brand jeans	192
River Island shorts	68	Flared Jeans	240
Miss Selfridge skirts	70	Activewear Pants	277
Valentino skirts	71	Frame jeans	282
Balmain jeans	81	Straight Leg Jeans	291
MANGO jeans	82	Long Skirts	414
Dolce&Gabbana skirts	85	Topshop jeans	435
Levi's shorts	86	Leggings	606
Gucci skirts	86	Capri & Cropped Pants	677
Abercrombie & Fitch jeans	87	Boyfriend Jeans	1093
Bootcut Jeans	89	Mini Skirts	2035
Hollister Co. Jeans	93	Skinny Jeans	3228
Yves Saint Laurent jeans	93	Shorts	3307
STELLA McCARTNEY jeans	96	Knee Length Skirts	3963
Topshop mini skirts	97	Pants	4117
Wide Leg Jeans	97	Jeans	6609
J.Crew jeans	104	Skirts	6938
River Island skirts	109		

Table 13 Brand Diversity of our Validation Data (Tops)

TOPS			
Type	**Number**	**Type**	**Number**
H&M tops	10	Activewear	32
Miss Selfridge tops	10	Hoodies	33
Men's Clothing	11	Sweatshirts	74
Tank Tops	11	Sweatshirts & Hoodies	107
Acne Studios sweaters	11	T-Shirts	314
Men's Fashion	13	Blouses	379
River Island tops	13	Sweaters	400
Tunics	14	Tops	1796
MANGO sweaters	15	Clothing	2083
Boohoo tops	16	Women's Fashion	3792
Topshop tops	32		

Table 14 Brand Diversity of our Validation Data (Bottoms)

BOTTOMS			
Type	**Number**	**Type**	**Number**
MANGO jeans	10	Current/Elliott jeans	23
Citizens of Humanity jeans	10	Activewear Pants	25
Dolce&Gabbana skirts	10	Long Skirts	32
7 For All Mankind jeans	12	Topshop jeans	44
AG Adriano Goldschmied jeans	13	Leggings	46
Topshop shorts	13	Capri & Cropped Pants	64
Chicwish skirts	13	Boyfriend Jeans	107
rag & bone jeans	14	Mini Skirts	178
River Island jeans	15	Shorts	291
Levi's jeans	16	Skinny Jeans	305
Paige Denim jeans	17	Knee Length Skirts	333
J Brand jeans	18	Pants	401
Flared Jeans	19	Skirts	586
Straight Leg Jeans	20	Jeans	612
Frame jeans	21		

The textual data is taken as the titles of the clothes, if not available, their URLs. These texts have been encoded into 6000 dimensional vectors with the HGLMM Fisher vector encoding method, which we will discuss in detail later.

Before the processed visual data is given as input to the model, it is subjected to scaling, cropping and normalization processes. Since the ResNet-18 model, which will take the feature vector of the features of the image, takes 224x224 images with three channels as input, the images are scaled accordingly. In addition, the images were normalized during model training in order not to encounter exploding/vanishing gradient problems and to avoid excessive use of the processor power of the model.

5.2 Implementation of Proposed System

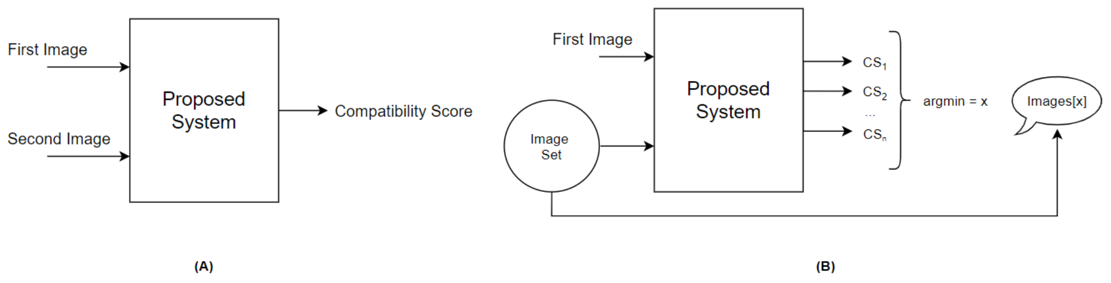


Figure 29 Blackbox Design of the Model

The system can basically generate scores for the compatibility between two outfits or suggest a complement to an outfit. For the compatibility score calculation, the distance between two images is calculated by taking their embeddings in the shared type space. For this distance, "pairwise distance" was chosen as the metric. In this way, multidimensional data is reduced to a single dimension. For the complementary product recommendation, a certain outfit and candidate product set are given to the system and the product that gives the least distance between them is selected and presented to the user. The general training principal of the model is given in Figure 33.

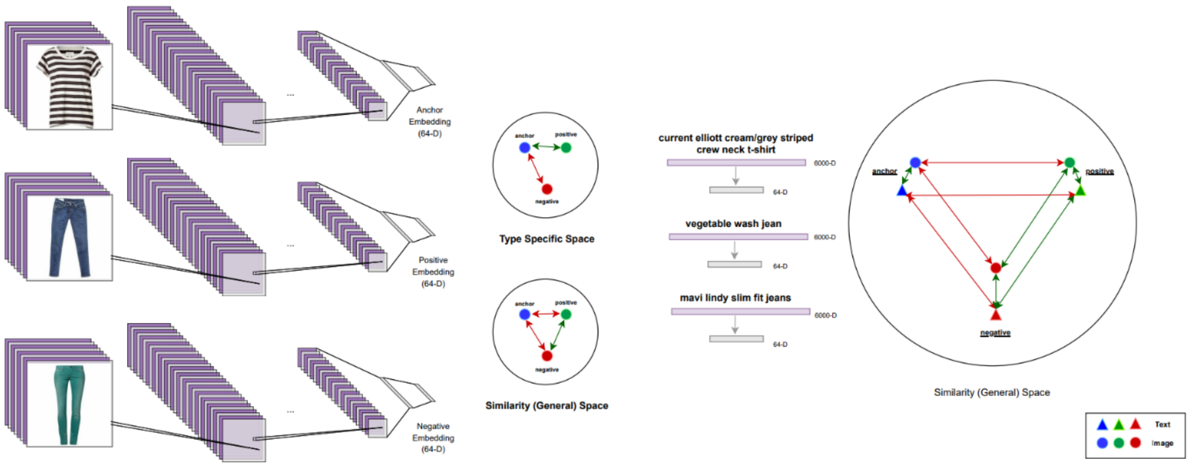

Anchor Embedding (64-D)
Positive Embedding (64-D)
Negative Embedding (64-D)
anchor
positive
negative
Type Specific Space
Similarity (General) Space
current elliott cream/grey striped crew neck t-shirt
6000-D
64-D
vegetable wash jean
mavi lindy slim fit jeans
Similarity (General) Space
Text
Image

Figure 30 The design principle of the presented model

The ResNet-18 model, previously trained with the ImageNet dataset, was used for the extraction of visual features. Here, it is aimed to increase the success of the model by applying to models that can extract the basic visual features of the pictures by using "transfer learning". However, the ResNet-18 model was preferred due to its multi-layered structure and its success in computer vision projects. In the extraction of textual features, HGLMM (Hybrid Gaussian-Laplacian Mixture Models) Fisher vector encoding [49] method. This encoding method was preferred because of its success in image tagging and RNN image search via sentence. The 6000-D vectors obtained as a result of this encoding method were converted into 64-D vectors by passing through the fully connected layer, thus reducing them to the shared general space with visual embeddings.

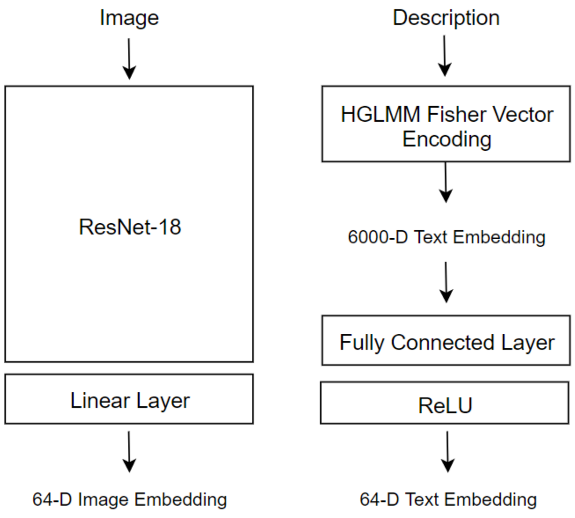
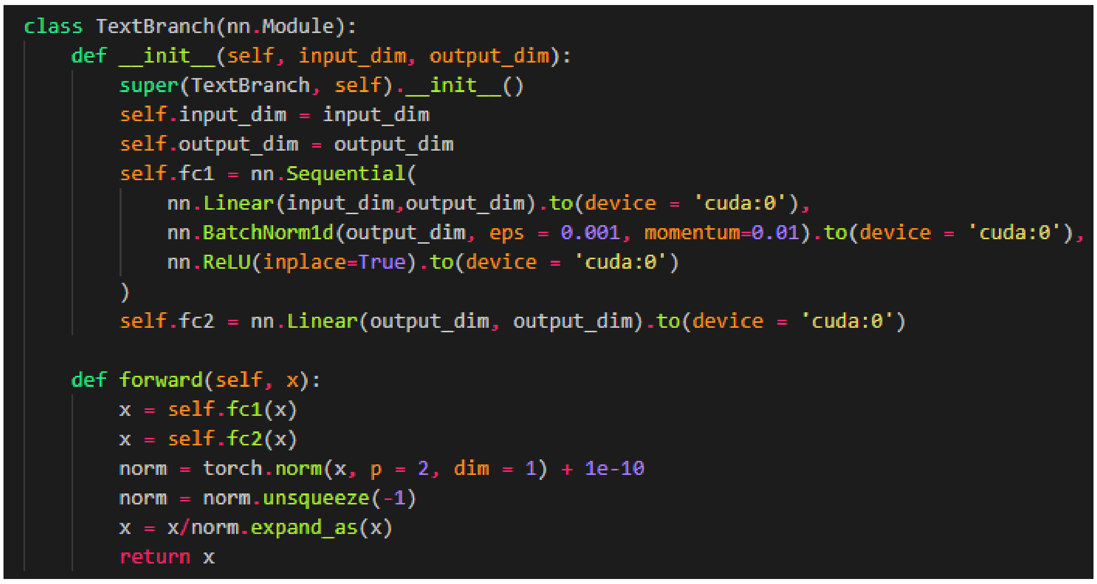


Figure 31 Reduction of visual and textual features to 64-D

Visual Semantic Loss and Similarity Loss are calculated in the space where these 64-D vectors are located. As seen in Figure 34, products in the same category in the general space are considered similar, while products from different categories are considered different. However, since the textual descriptions and visual features of the products should be close, the model is optimized so that the distance between the vectors is minimal.



```
class TextBranch(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(TextBranch, self).__init__()
        self.input_dim = input_dim
        self.output_dim = output_dim
        self.fc1 = nn.Sequential(
            nn.Linear(input_dim,output_dim).to(device = 'cuda:0'),
            nn.BatchNorm1d(output_dim, eps = 0.001, momentum=0.01).to(device = 'cuda:0'),
            nn.ReLU(inplace=True).to(device = 'cuda:0')
        )
        self.fc2 = nn.Linear(output_dim, output_dim).to(device = 'cuda:0')

    def forward(self, x):
        x = self.fc1(x)
        x = self.fc2(x)
        norm = torch.norm(x, p = 2, dim = 1) + 1e-10
        norm = norm.unsqueeze(-1)
        x = x/norm.expand_as(x)
        return x
```

Figure 32 TextBranch Class

Figure 35 contains the codes of the TextBranch class, which reduces 6000-D textual descriptions to 64-D. Here, as mentioned, embeddings are reduced by passing through fully connected layers where batch normalization and ReLU (Rectified Linear Unit) functions are located. However, since the direction and dimensions of the matrix are important rather than the size of the matrix, and it is not desired to encounter problems such as exploding gradient in the deep neural network, the matrix is normalized in the L2 norm.

As mentioned before, there are some points to be aware of: similar products may not be compatible with each other, the complementary product of two similar products may be different, and the feature vectors obtained with CNN may be calculated so that similar images are close to each other. For this reason, compatibility is not considered on existing similarity space, and to compensate for this, subspaces in which dual types are specially examined, are created.

In order to reflect the vectors to the subspaces, a linear mask layer that can be developed with training was created and the existing vectors were multiplied on an element-wise basis. By calculating the pairwise distances of the final vectors obtained, the error is calculated depending on whether they are compatible or not. Figure 36 shows the design explanation of this projection process.

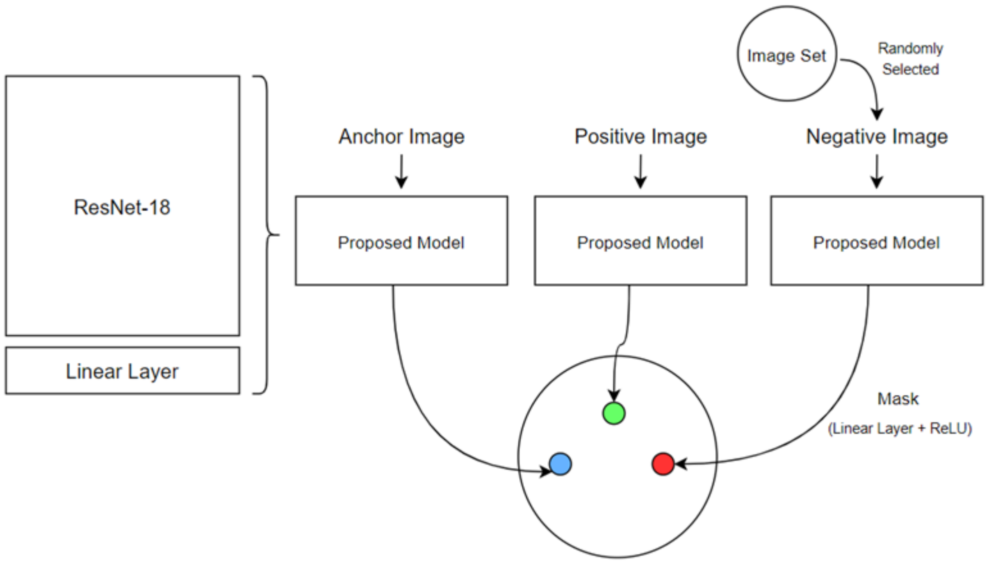
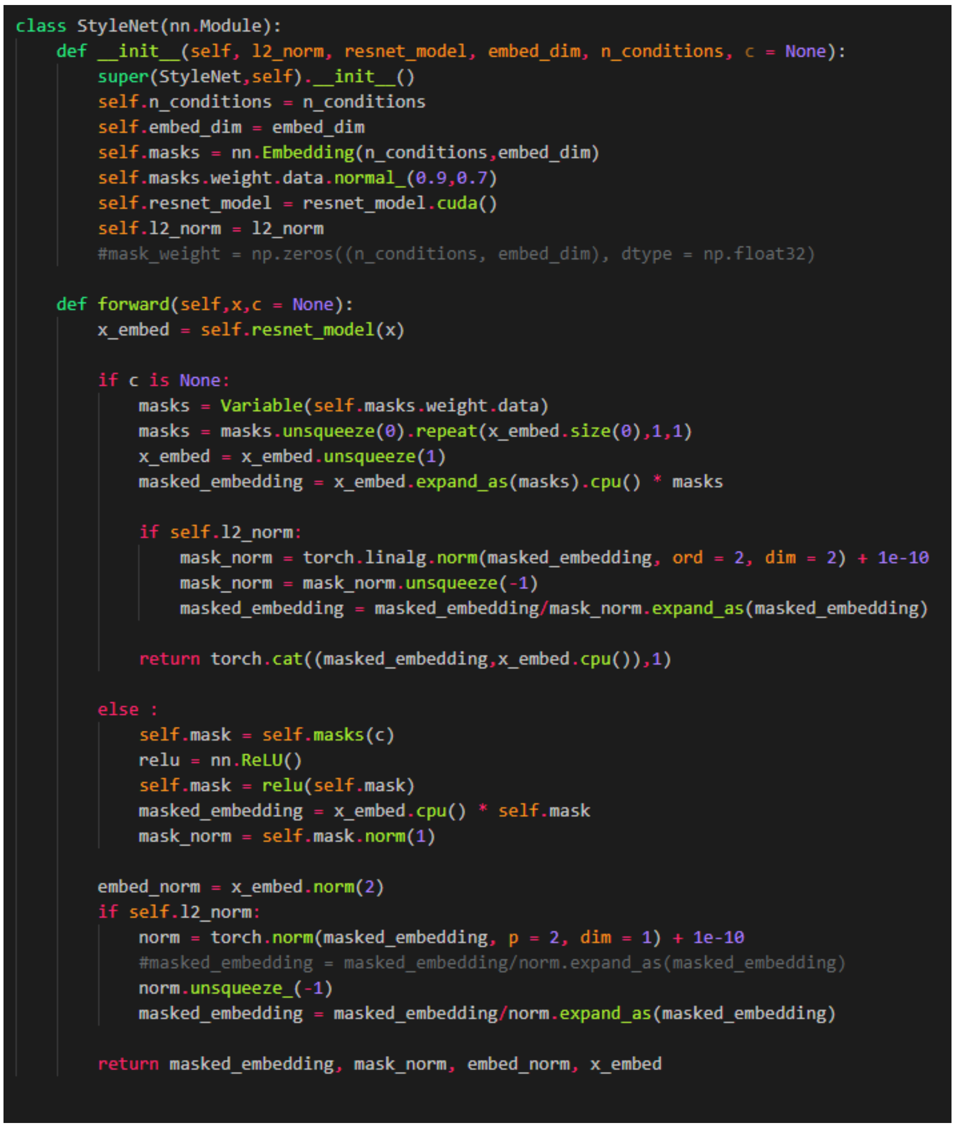


Figure 33 Projection of images into the compatibility space

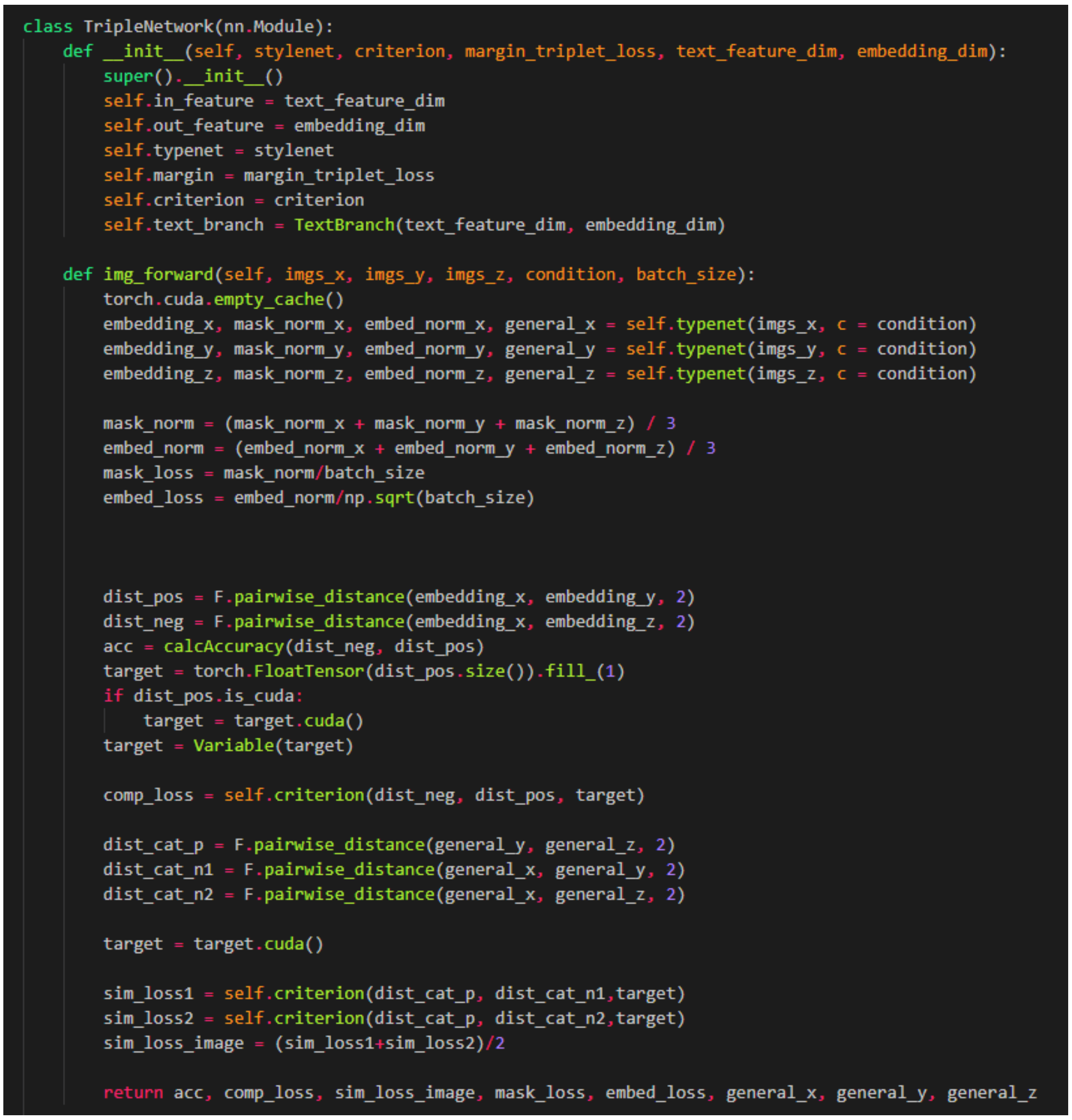
In Figure 37, there is the "StyleNet" class, which is used to reflect visual embeddings to subspace. A mask is created for the 64-dimensional visual vector x and the sub-binary type spaces it will reflect, and its values are fitted to the normal distribution with a mean of 0.9 and a standard deviation of 0.7. The fetching of all or one of the reflected embeddings is determined by the variable c. The resulting projected new embedding/embeddings return with their original vectors.



```python
class StyleNet(nn.Module):
    def __init__(self, l2_norm, resnet_model, embed_dim, n_conditions, c = None):
        super(StyleNet,self).__init__()
        self.n_conditions = n_conditions
        self.embed_dim = embed_dim
        self.masks = nn.Embedding(n_conditions,embed_dim)
        self.masks.weight.data.normal_(0.9,0.7)
        self.resnet_model = resnet_model.cuda()
        self.l2_norm = l2_norm
        #mask_weight = np.zeros((n_conditions, embed_dim), dtype = np.float32)

    def forward(self,x,c = None):
        x_embed = self.resnet_model(x)

        if c is None:
            masks = Variable(self.masks.weight.data)
            masks = masks.unsqueeze(0).repeat(x_embed.size(0),1,1)
            x_embed = x_embed.unsqueeze(1)
            masked_embedding = x_embed.expand_as(masks).cpu() * masks

            if self.l2_norm:
                mask_norm = torch.linalg.norm(masked_embedding, ord = 2, dim = 2) + 1e-10
                mask_norm = mask_norm.unsqueeze(-1)
                masked_embedding = masked_embedding/mask_norm.expand_as(masked_embedding)

            return torch.cat((masked_embedding,x_embed.cpu()),1)

        else :
            self.mask = self.masks(c)
            relu = nn.ReLU()
            self.mask = relu(self.mask)
            masked_embedding = x_embed.cpu() * self.mask
            mask_norm = self.mask.norm(1)

        embed_norm = x_embed.norm(2)
        if self.l2_norm:
            norm = torch.norm(masked_embedding, p = 2, dim = 1) + 1e-10
            #masked_embedding = masked_embedding/norm.expand_as(masked_embedding)
            norm.unsqueeze_(-1)
            masked_embedding = masked_embedding/norm.expand_as(masked_embedding)

        return masked_embedding, mask_norm, embed_norm, x_embed
```

Figure 34 StyleNet Class

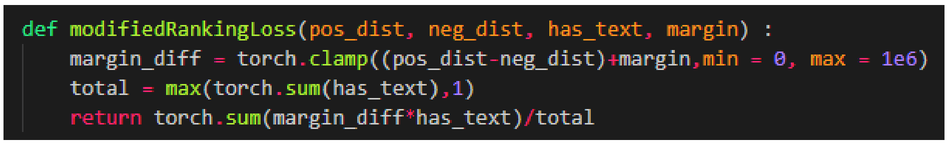
As seen in Figure 38, all embeddings obtained (text embedding, visual embedding and embeddings projected in the type space) are classified into the "TripleNetwork" class, and similarity and compatibility errors of the images are calculated in the img_forward function.



```python
class TripleNetwork(nn.Module):
    def __init__(self, stylenet, criterion, margin_triplet_loss, text_feature_dim, embedding_dim):
        super().__init__()
        self.in_feature = text_feature_dim
        self.out_feature = embedding_dim
        self.typenet = stylenet
        self.margin = margin_triplet_loss
        self.criterion = criterion
        self.text_branch = TextBranch(text_feature_dim, embedding_dim)

    def img_forward(self, imgs_x, imgs_y, imgs_z, condition, batch_size):
        torch.cuda.empty_cache()
        embedding_x, mask_norm_x, embed_norm_x, general_x = self.typenet(imgs_x, c = condition)
        embedding_y, mask_norm_y, embed_norm_y, general_y = self.typenet(imgs_y, c = condition)
        embedding_z, mask_norm_z, embed_norm_z, general_z = self.typenet(imgs_z, c = condition)

        mask_norm = (mask_norm_x + mask_norm_y + mask_norm_z) / 3
        embed_norm = (embed_norm_x + embed_norm_y + embed_norm_z) / 3
        mask_loss = mask_norm/batch_size
        embed_loss = embed_norm/np.sqrt(batch_size)



        dist_pos = F.pairwise_distance(embedding_x, embedding_y, 2)
        dist_neg = F.pairwise_distance(embedding_x, embedding_z, 2)
        acc = calcAccuracy(dist_neg, dist_pos)
        target = torch.FloatTensor(dist_pos.size()).fill_(1)
        if dist_pos.is_cuda:
            target = target.cuda()
        target = Variable(target)

        comp_loss = self.criterion(dist_neg, dist_pos, target)

        dist_cat_p = F.pairwise_distance(general_y, general_z, 2)
        dist_cat_n1 = F.pairwise_distance(general_x, general_y, 2)
        dist_cat_n2 = F.pairwise_distance(general_x, general_z, 2)

        target = target.cuda()

        sim_loss1 = self.criterion(dist_cat_p, dist_cat_n1,target)
        sim_loss2 = self.criterion(dist_cat_p, dist_cat_n2,target)
        sim_loss_image = (sim_loss1+sim_loss2)/2

        return acc, comp_loss, sim_loss_image, mask_loss, embed_loss, general_x, general_y, general_z
```

Figure 35 TripleNetwork Class

While calculating the errors, the modifiedRankingLoss function in Figure 39 is called. This function is a modified version of the error function known as Margin Ranking Loss based on existing embeddings.



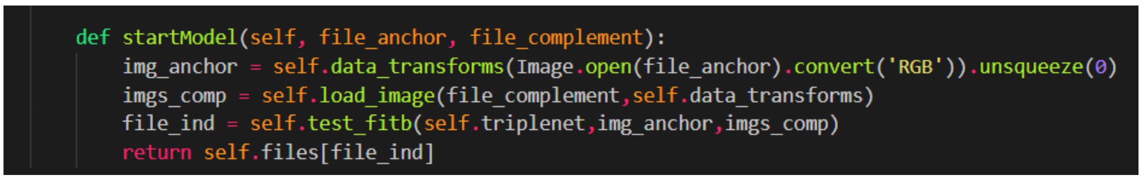
```python
def modifiedRankingLoss(pos_dist, neg_dist, has_text, margin) :
    margin_diff = torch.clamp((pos_dist-neg_dist)+margin,min = 0, max = 1e6)
    total = max(torch.sum(has_text),1)
    return torch.sum(margin_diff*has_text)/total
```

Figure 36 Modified Ranking Loss Function

Basically, the aforementioned model can calculate the compatibility error for the clothing data given in bulk, and it can also calculate the error by performing the task of choosing the best among the multiple-choice options offered for a given product, also called fill-in-the-blank.

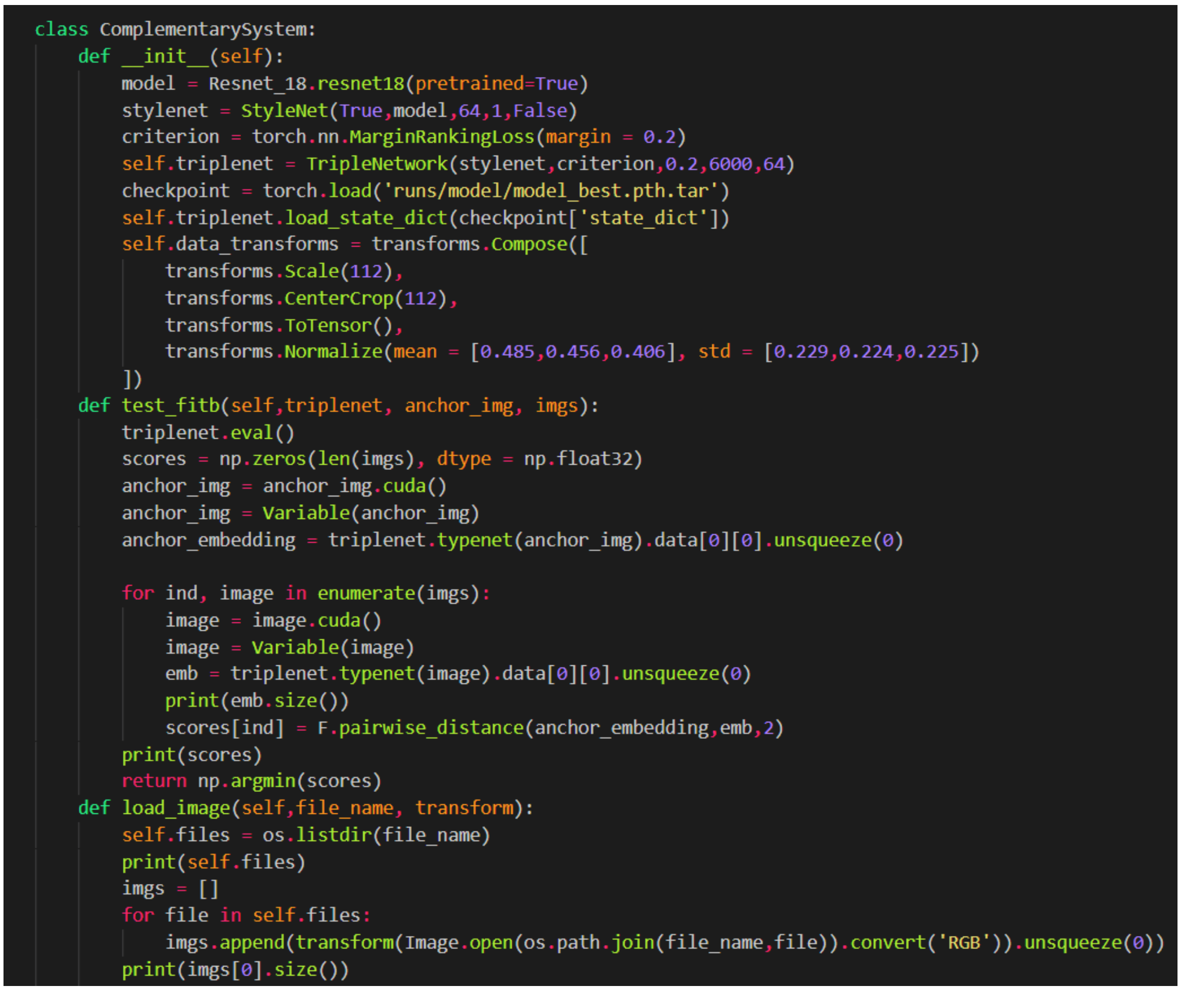
In order to adapt the application to real life and the use of users, another layer has been created on the model, and the system has been designed to offer complementary clothing for a reference outfit among several candidate clothing images. Below is the program code of the added layer program code of the added layer.

With this system, the images of the reference outfit and the candidate outfits are taken in the RGB channel and undergo transformations such as scaling, cropping and normalization. Then, the obtained images are given to the pre-trained, highest-accuracy complementary clothing recommendation model, and their embeddings in the compatibility space are taken. By calculating the distances of the embeddings obtained in the compatibility space, the image in the folder that gives the minimum distance is presented to the user. Figure 41 shows the startModel function that must be called to run the modules.



```
def startModel(self, file_anchor, file_complement):
    img_anchor = self.data_transforms(Image.open(file_anchor).convert('RGB')).unsqueeze(0)
    imgs_comp = self.load_image(file_complement,self.data_transforms)
    file_ind = self.test_fitb(self.triplenet,img_anchor,imgs_comp)
    return self.files[file_ind]
```

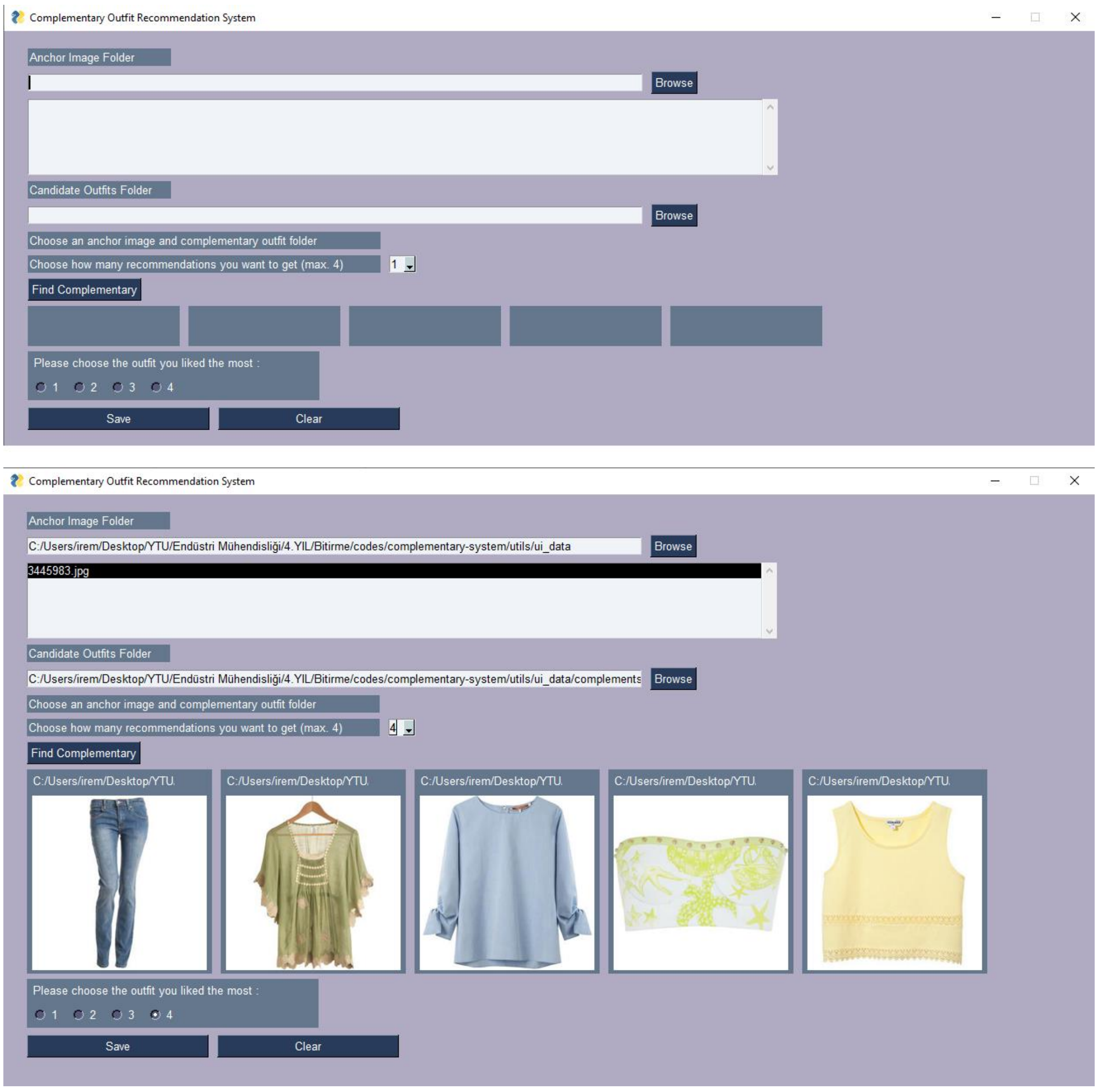
Figure 37 startModel function created for the user to use the model



```python
class ComplementarySystem:
    def __init__(self):
        model = Resnet_18.resnet18(pretrained=True)
        stylenet = StyleNet(True,model,64,1,False)
        criterion = torch.nn.MarginRankingLoss(margin = 0.2)
        self.triplenet = TripleNetwork(stylenet,criterion,0.2,6000,64)
        checkpoint = torch.load('runs/model/model_best.pth.tar')
        self.triplenet.load_state_dict(checkpoint['state_dict'])
        self.data_transforms = transforms.Compose([
            transforms.Scale(112),
            transforms.CenterCrop(112),
            transforms.ToTensor(),
            transforms.Normalize(mean = [0.485,0.456,0.406], std = [0.229,0.224,0.225])
        ])
    def test_fitb(self,triplenet, anchor_img, imgs):
        triplenet.eval()
        scores = np.zeros(len(imgs), dtype = np.float32)
        anchor_img = anchor_img.cuda()
        anchor_img = Variable(anchor_img)
        anchor_embedding = triplenet.typenet(anchor_img).data[0][0].unsqueeze(0)

        for ind, image in enumerate(imgs):
            image = image.cuda()
            image = Variable(image)
            emb = triplenet.typenet(image).data[0][0].unsqueeze(0)
            print(emb.size())
            scores[ind] = F.pairwise_distance(anchor_embedding,emb,2)
        print(scores)
        return np.argmin(scores)
    def load_image(self,file_name, transform):
        self.files = os.listdir(file_name)
        print(self.files)
        imgs = []
        for file in self.files:
            imgs.append(transform(Image.open(os.path.join(file_name,file)).convert('RGB')).unsqueeze(0))
        print(imgs[0].size())
```

Figure 38 Fill-in-the-blank accuracy calculation function

The user interface created with the created middleware and the PySimpleGUI library is shown in Figure 42. First, the user selects the folder of the image of the outfit he/she wants to reference, then selects the file and uploads it to the interface. Then he/she selects the folder with the candidate products and selects how many recommendations he/she wants to get; then, clicks the "Find Complementary" button. The model working in the background returns the image of the complementary outfit she finds best in a few seconds. The user can save this product in his folder and use the system again to get suggestions for other outfits.



Complementary Outfit Recommendation System
Anchor Image Folder
Browse
Candidate Outfits Folder
Browse
Find Complementary
Please choose the outfit you liked the most :
1
2
3
4
Save
Clear

Anchor Image Folder: C:/Users/irem/Desktop/YTU/Endüstri Mühendisliği/4.YIL/Bitirme/codes/complementary-system/utils/ui_data

3445983.jpg

Candidate Outfits Folder: C:/Users/irem/Desktop/YTU/Endüstri Mühendisliği/4.YIL/Bitirme/codes/complementary-system/utils/ui_data/complements

Choose an anchor image and complementary outfit folder

Choose how many recommendations you want to get (max. 4): 1

Choose how many recommendations you want to get (max. 4): 4

C:/Users/irem/Desktop/YTU.

C:/Users/irem/Desktop/YTU.

C:/Users/irem/Desktop/YTU.

C:/Users/irem/Desktop/YTU.

C:/Users/irem/Desktop/YTU.

Please choose the outfit you liked the most : 1 2 3 [✓ 4]

Figure 39 The user interface of the proposed model

In this section, the analysis of visual and textual data, which has a great impact on the success of the model, is included, and then how the proposed conceptual model is programmed during the transition phase is mentioned. In the next section, the metrics of this model and the success of the application in various dimensions will be discussed.

5.3 Evaluation of the Proposed System

The performance of the system, as well as the importance of the design and implementation of the system, should not be overlooked. In this title, firstly, the performance metrics we determined for the system were briefly explained, then the results obtained from these metrics were included, and finally the system was evaluated in general and some comments were made.

5.3.1 Introduction of Performance Metrics

The performance metrics we have determined are listed below with their explanations.

Area Under the Curve (AUC): AUC is an important metric that is often considered in deep learning. It shows how successful the deep learning model is, making predictions/classifications away from randomness. AUC close to 1 indicates that the model works successfully, while a value of 0.5 indicates that it has made predictions/classifications without any basis.

Accuracy: Accuracy, like AUC, is a frequently used metric in the field of machine learning. It is calculated based on the closeness of the prediction/classification to the actual result. In our study, accuracy is found by the difference in the distance between compatible and incompatible clothing pairs in the compatibility space. Where x is the reference outfit, y is the matching outfit, z is the mismatched outfit, and d_{xz} and d_{xz} are the difference in the embeddings of these outfit images in the style space, accuracy can be expressed by

$$accuracy = d_{xz} - d_{xy} \tag{5.1}$$

FITB Accuracy: FITB Accuracy is about the fill-in-the-blank function that we have mentioned in the study. It is calculated by the distances in space of an item set consisting of 4 clothes, containing a reference item and its complement. Then, it is checked whether the product giving the least distance is the correct answer. FITB accuracy is calculated by dividing the number of correct predictions made in this way by the total number of predictions.

General Loss: The general loss consists of the loss made in the compatibility calculation, the loss made while projecting the visual features into the compatibility space, and the mask loss which is the performance of reflecting the visuals into the compatibility space. These errors are covered in detail in the theoretical conceptual design section.

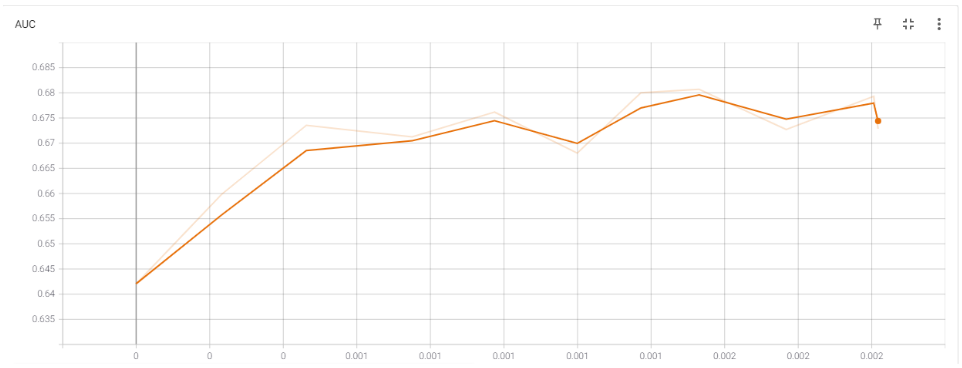
Response Time: In the interface created for the model to be used, the user will be affected by the system's response speed. Therefore, both the initial setup time of the model when opening the application and the time of the application to find the complementary product among a certain number of candidate products were determined as performance metrics. This metric has come to the fore as our study is application-oriented, compared to other studies.

5.3.2 Model Performance

Before the model is tested with bulk data, it must first be tested with various parameters, appropriate parameters must be selected depending on certain criteria, and then the model established with these parameters must be trained and tested in detail; The parameters discussed in this study are learning rate and optimizer.

Provided that other parameters remain constant, different options for learning rate and optimizer were tried, and the one with the best performance was selected. For this, the models with learning parameters 0.0001, 0.00005 and 0.001, respectively, were tested for 10 epochs and the results were compared.

The change in AUC (Area Under the Curve) values, which is one of our performance metrics, are shown in Figure 43 for the 0.0001 and 0.001 learning parameters during 10 epochs.



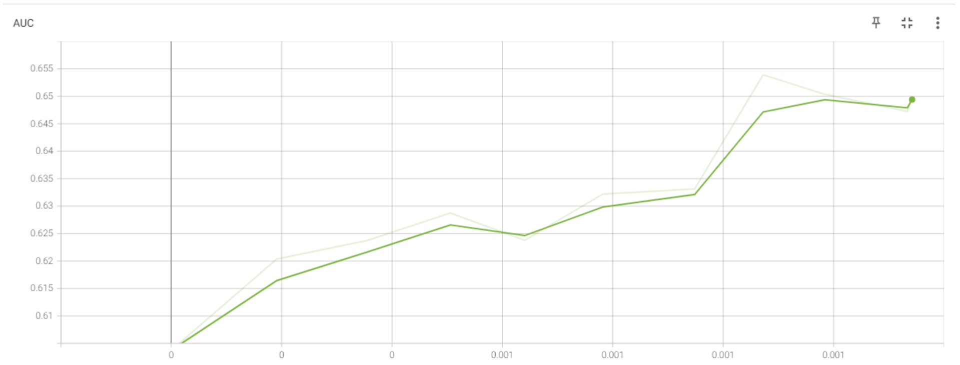
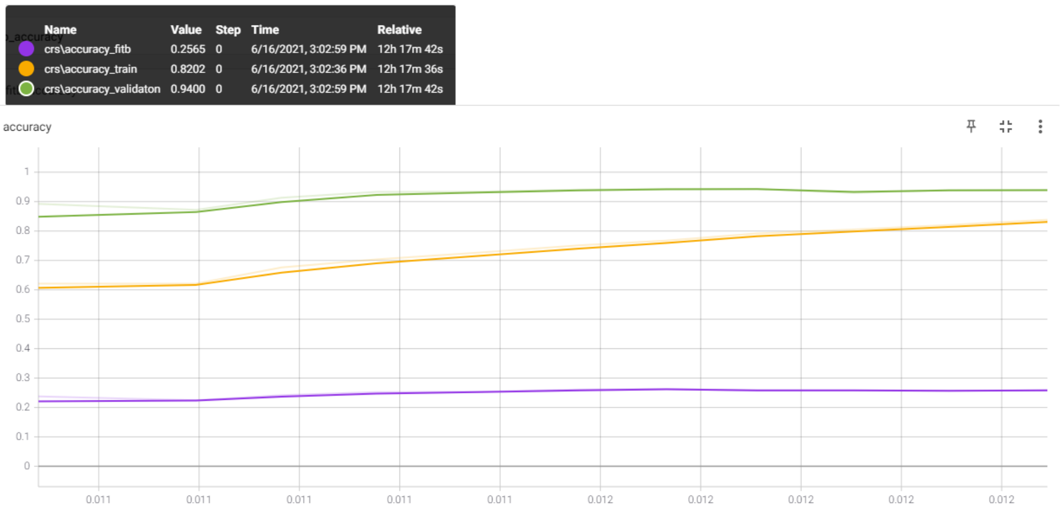


Figure 40: AUC value for different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

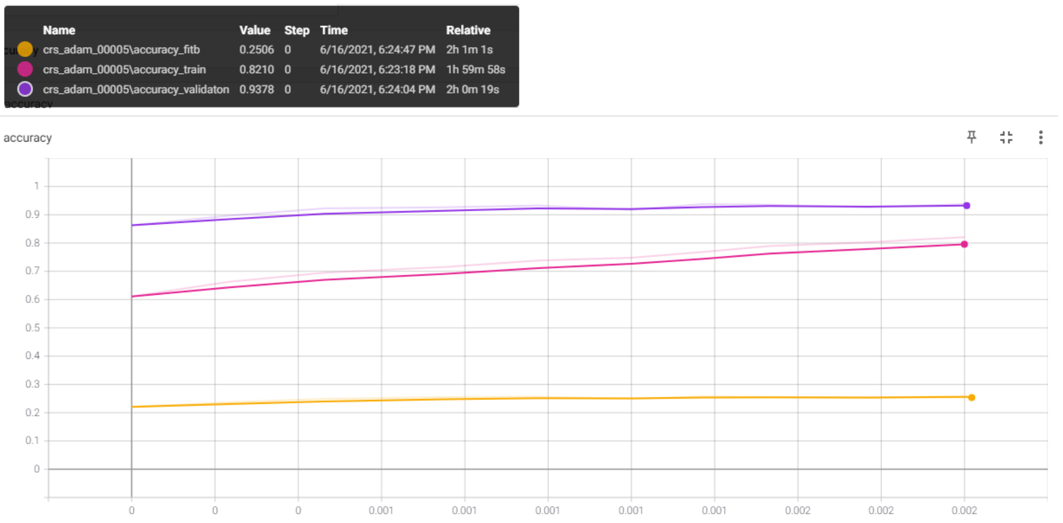
AUC, as we mentioned earlier, is a basic metric that shows how accurately the model can classify. Here, it can be seen that the first model achieves a better result after 10 epochs, although the second model appears to improve its performance over time.

The chart below shows the values of the performance metrics obtained for each learning parameter.



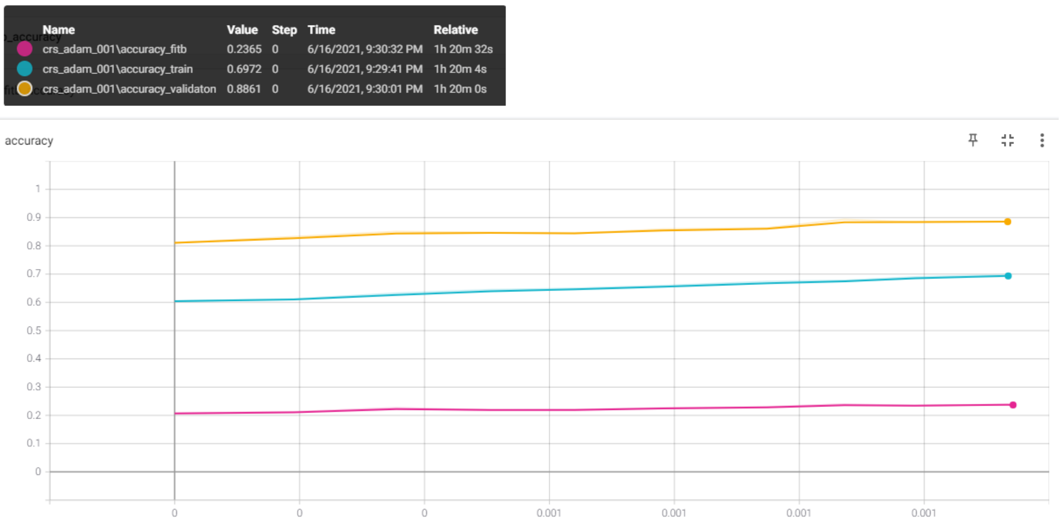
	Name	Value	Step	Time	Relative
●	crs\accuracy_fitb	0.2565	0	6/16/2021, 3:02:59 PM	12h 17m 42s
●	crs\accuracy_train	0.8202	0	6/16/2021, 3:02:36 PM	12h 17m 36s
●	crs\accuracy_validaton	0.9400	0	6/16/2021, 3:02:59 PM	12h 17m 42s

(A)



	Name	Value	Step	Time	Relative
●	crs_adam_00005\accuracy_fitb	0.2506	0	6/16/2021, 6:24:47 PM	2h 1m 1s
●	crs_adam_00005\accuracy_train	0.8210	0	6/16/2021, 6:23:18 PM	1h 59m 58s
●	crs_adam_00005\accuracy_validaton	0.9378	0	6/16/2021, 6:24:04 PM	2h 0m 19s

(B)



	Name	Value	Step	Time	Relative
●	crs_adam_001\accuracy_fitb	0.2365	0	6/16/2021, 9:30:32 PM	1h 20m 32s
●	crs_adam_001\accuracy_train	0.6972	0	6/16/2021, 9:29:41 PM	1h 20m 4s
●	crs_adam_001\accuracy_validaton	0.8861	0	6/16/2021, 9:30:01 PM	1h 20m 0s

(C)

Figure 41 Accuracy for different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

Here, it can be seen that the train and validation accuracies of the first and second models are getting closer, while the distance between the accuracy of the third model is generally maintained. However, the common point of each model is that “validation accuracy” has the highest value and “fitb accuracy” has the lowest value. This should be considered normal because the validation dataset is smaller than the train and test datasets, and the constant exposure of the model to this little data made it more successful in validation data. The reason why "fitb accuracy" remains low is due to the nature of the fill-in-the-blank task. Finding the right one among the four products for the model is more difficult than calculating a compatibility ratio. However, we can say that this accuracy has also improved, albeit very slightly.

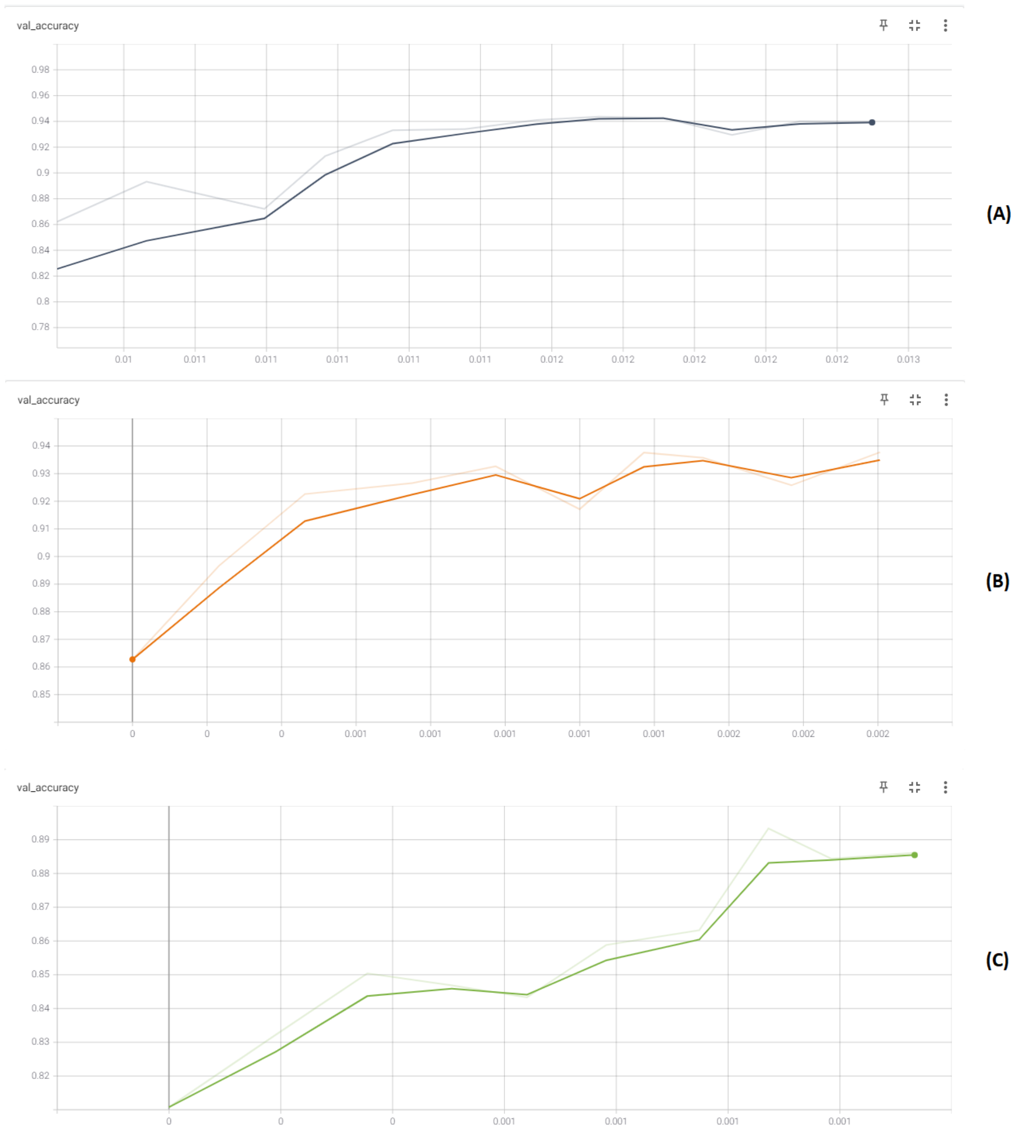


Figure 42 Validation accuracy for different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

Accuracy values were examined in more detail and validation accuracy and fill-in-the-blank accuracy values were compared. Below is the representation of the validation accuracy values for the three models.

When the validation accuracy is examined, we can say that all three models show positive progress. However, considering the validation accuracy values, it is seen that the first two models show the best values.

The accuracy rates of the models in the fill-in-the-blank task are shown in Figure 46. Here, again, the first and second models achieved higher accuracy than the third model. However, the rate of increase in the accuracy of the third model over time is higher than the others. This is because its learning rate is greater than others; It takes larger steps during optimization than other models, which causes values to change faster.

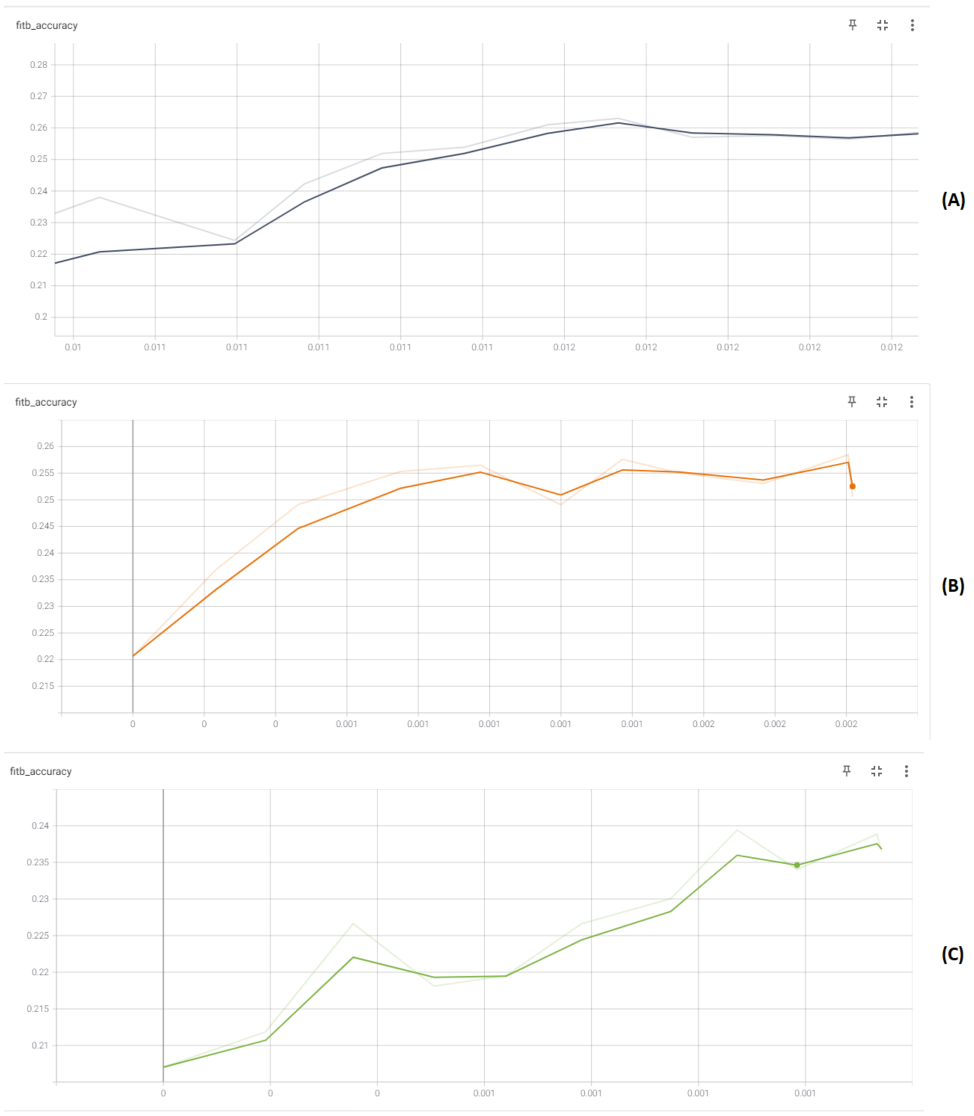


Figure 43 Fill-in-the-blank accuracy for different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

The training loss calculated for the three models are given in Figure 47. As can be seen, while there is a more balanced decrease in the first two models, there are unbalanced decreases and increases in the third model. This is because, as mentioned earlier, the learning rate remains large for the model. The model, which tries to find the local minimum point, has a large step size value due to the learning rate value and cannot converge to this point.

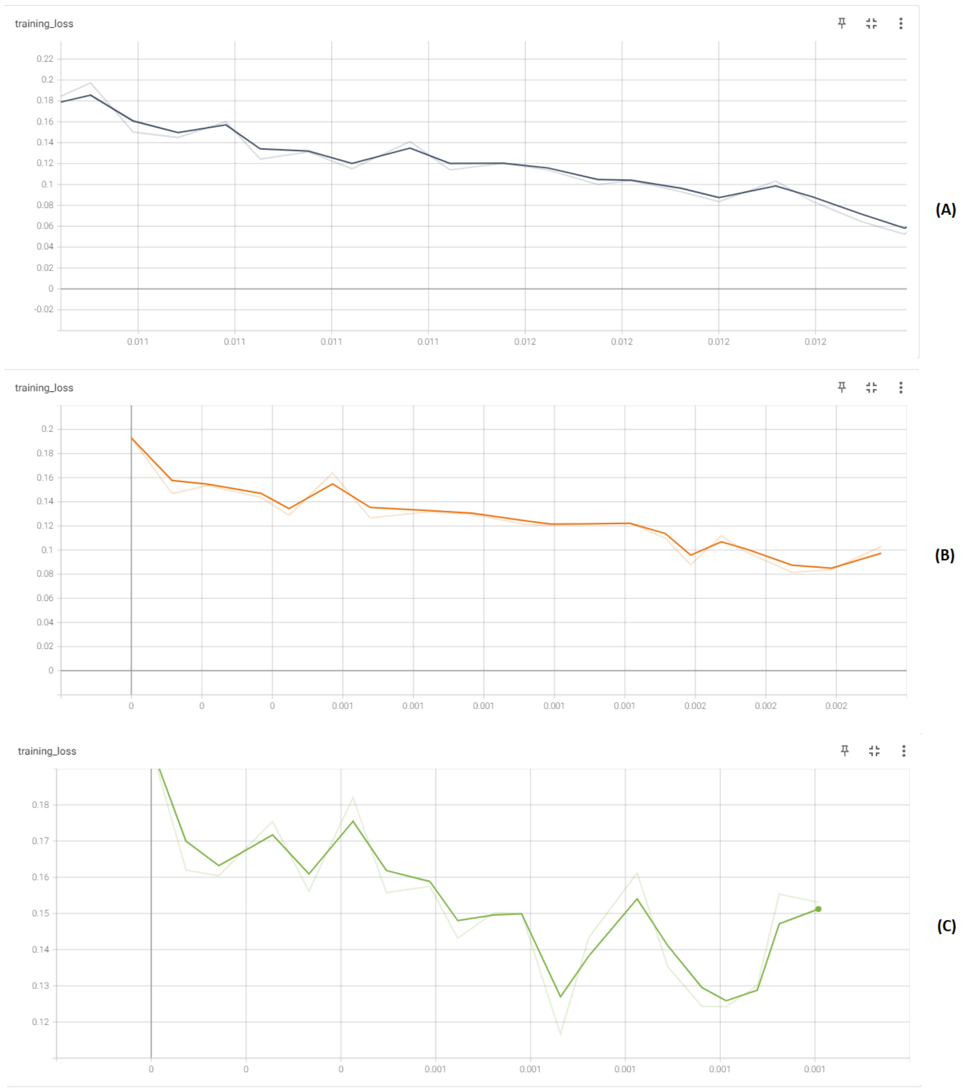
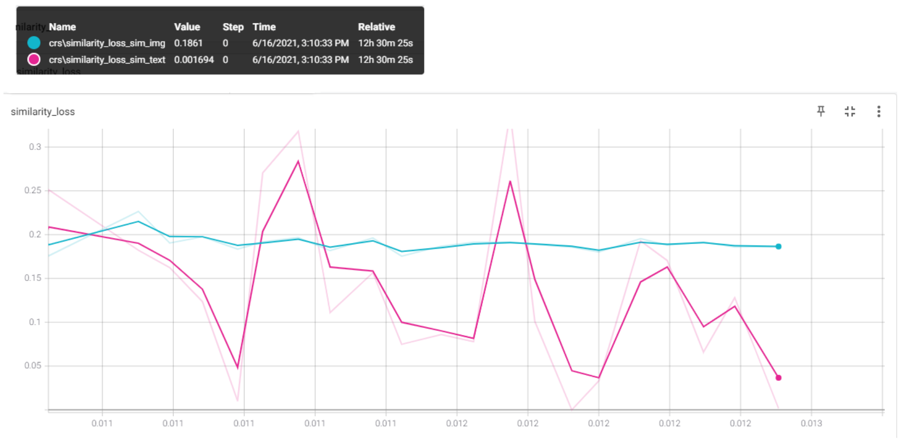


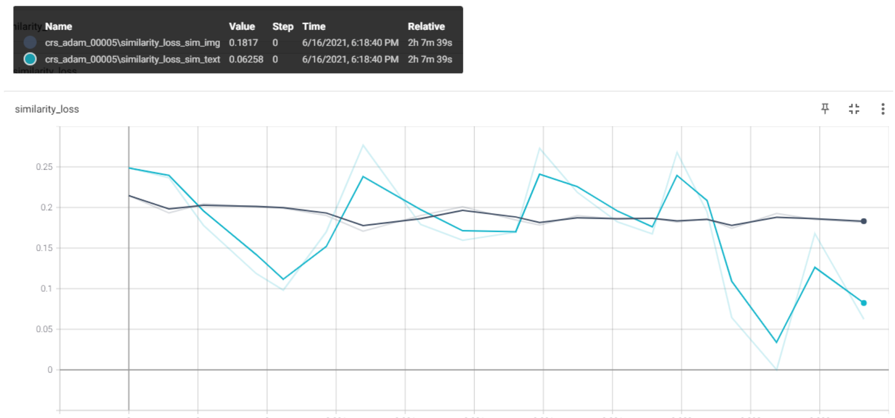
Figure 44 Training Loss for three different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

Although the purpose of the system is to be able to assign the correct compatibility value to the clothes, it is also important that the system can grasp the similarities between the clothes; The fact that the similarity can be well understood by the model shows that the visual and textual features of the outfit are also descriptive. Therefore, below are graphs with visual and textual similarity error values calculated in training the data.



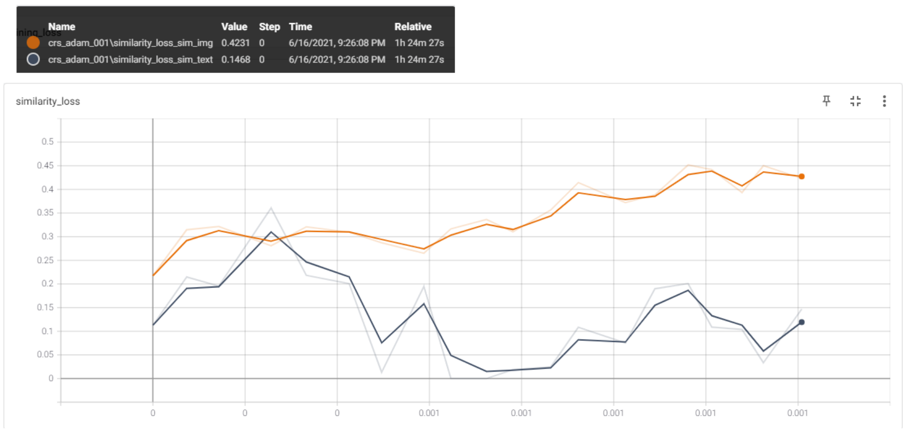
	Name	Value	Step	Time	Relative
●	crs\similarity_loss_sim_img	0.1861	0	6/16/2021, 3:10:33 PM	12h 30m 25s
●	crs\similarity_loss_sim_text	0.001694	0	6/16/2021, 3:10:33 PM	12h 30m 25s

(A)



	Name	Value	Step	Time	Relative
●	crs_adam_00005\similarity_loss_sim_img	0.1817	0	6/16/2021, 6:18:40 PM	2h 7m 39s
●	crs_adam_00005\similarity_loss_sim_text	0.06258	0	6/16/2021, 6:18:40 PM	2h 7m 39s

(B)



	Name	Value	Step	Time	Relative
●	crs_adam_001\similarity_loss_sim_img	0.4231	0	6/16/2021, 9:26:08 PM	1h 24m 27s
○	crs_adam_001\similarity_loss_sim_text	0.1468	0	6/16/2021, 9:26:08 PM	1h 24m 27s

(C)

Figure 45 Similarity Loss for different learning ratios (A) 0.0001 (B) 0.00005 (C) 0.001

When we look at the charts in general, the first thing to notice is that the visual similarity error of the third model is increasing gradually. The reason for this may be the deviation of the model from the minimum point depending on the size of the learning rate.

When the first two models are compared, it can be said that the visual similarity error of both models is quite close and constant. In textual similarity, we can say that although both models have many ups and downs, the second model shows a more balanced change.

As a result, the values obtained at 10 epochs are shown in the tables below.

Table 16: General Loss, Accuracy and Fill-in-the-blank Accuracy values for three different learning ratios

	Learning Rate		
Epochs	**0.00005**	**0.0001**	**0.001**
1	0.1506	0.1692	0.1762
2	0.1415	0.1463	0.1678
3	0.1301	0.1334	0.1623
4	0.1220	0.1256	0.1568
5	0.1154	0.1140	0.1544
6	0.1154	0.1079	0.1532
7	0.1074	0.0990	0.1518
8	0.1011	0.0914	0.1408
9	0.0948	0.0852	0.1448
10	0.0865	0.0768	0.1380

	Learning Rate		
Epochs	**0.00005**	**0.0001**	**0.001**
1	66.13%	60.68%	59.56%
2	68.47%	67.59%	60.98%
3	71.48%	70.82%	63.14%
4	73.42%	72.42%	64.62%
5	73.42%	75.51%	64.94%
6	75.05%	76.84%	66.10%
7	77.07%	79.16%	66.45%
8	78.59%	80.87%	67.60%
9	80.17%	82.12%	68.94%
10	82.28%	84.14%	69.93%

	Learning Rate		
Epochs	**0.00005**	**0.0001**	**0.001**
1	22.10	22.40	20.70
2	23.70	24.20	21.20
3	24.90	25.20	22.70
4	25.60	25.40	21.80
5	24.90	26.10	22.00
6	25.80	26.30	22.60
7	25.50	25.70	23.00
8	25.30	25.70	23.90
9	25.80	25.60	23.40
10	25.10	25.90	23.90

Table 15: Best accuracy values for three different learning ratios

	Learning Rate		
	0.00005	**0.0001**	**0.001**
Best Accuracy	0.937763	0.943601	0.893329

Considering all table values, the model with a learning rate of 0.001 was not preferred because it had the lowest accuracy and highest loss value, and among the models with a learning rate of 0.0001 and 0.00005, a value of 0.0001 was chosen, which gives the highest accuracy and fill-in-the-blank accuracy and the lowest error value.

After selecting the learning rate, Adam and SGD (Stochastic Gradient Descent) options were tried as optimizers. The error, accuracy and fill-in-the-blank accuracy values obtained here are given in the tables below.

Table 17: General Loss, Accuracy and Fill-in-the-blank Accuracy values for two optimizers

Epochs	Adam	SGD
1	0.1692	0.1887
2	0.1463	0.1885
3	0.1334	0.1873
4	0.1256	0.1877
5	0.1140	0.1852
6	0.1079	0.1864
7	0.0990	0.1872
8	0.0914	0.1870
9	0.0852	0.1861
10	0.0768	0.1861

Epochs	Adam	SGD
1	60.68%	55.40%
2	67.59%	55.46%
3	70.82%	56.06%
4	72.42%	55.38%
5	75.51%	55.64%
6	76.84%	56.16%
7	79.16%	56.13%
8	80.87%	55.80%
9	82.12%	56.19%
10	84.14%	56.59%

Epochs	Adam	SGD
1	22.40	17.10
2	24.20	17.20
3	25.20	17.10
4	25.40	17.20
5	26.10	17.00
6	26.30	17.20
7	25.70	17.30
8	25.70	17.30
9	25.60	17.20
10	25.90	17.60

When the performance metrics were evaluated, Adam was determined as the most accurate optimizer option. The reason for Adam's higher performance than SGD can be attributed to the fact that Adam is more stable locally than SGD.

With these parameters, the batch size was specified as 128 depending on hardware constraints and performance, and the model was subjected to 40 epoch training. Obtained metric values are shown in the graphs below.

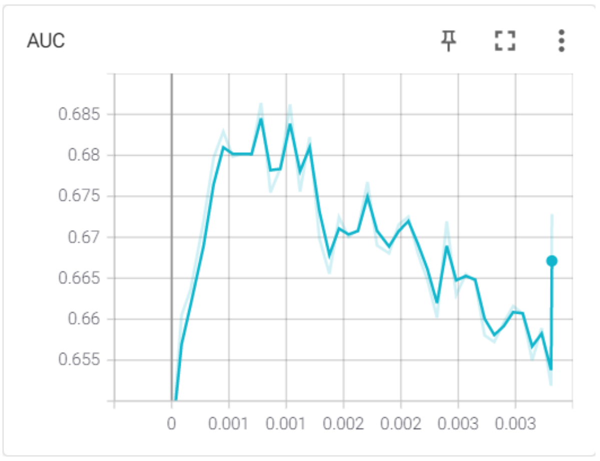


Figure 46: Change in AUC value in 40 epochs

While the AUC (Area Under the Curve) value of the model initially increased, it decreased over time. The reason for this is that the size of the dataset is not sufficient and the model is overfit.

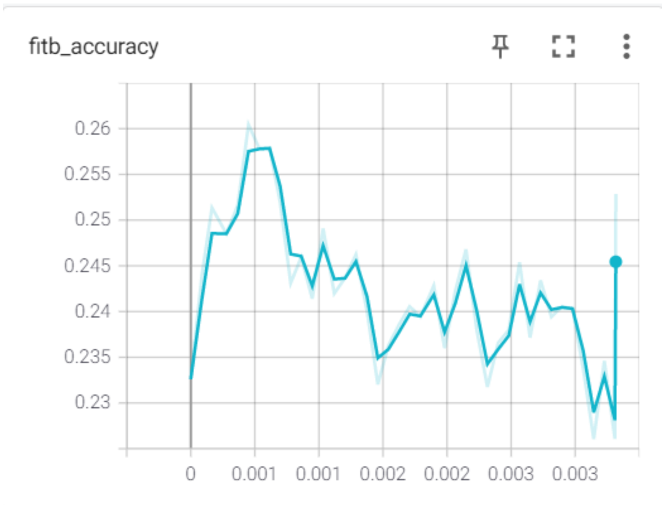
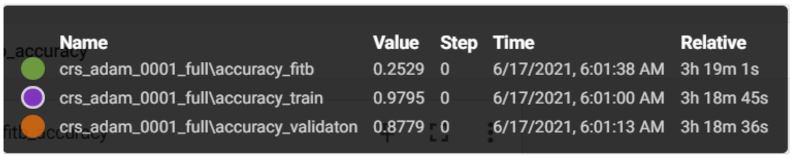


Figure 47: Change in FITB accuacy in 40 epochs

The accuracy values obtained during the training period of the model are shown in Figure 51.



	Name	Value	Step	Time	Relative
🟢	crs_adam_0001_full\accuracy_fitb	0.2529	0	6/17/2021, 6:01:38 AM	3h 19m 1s
🟣	crs_adam_0001_full\accuracy_train	0.9795	0	6/17/2021, 6:01:00 AM	3h 18m 45s
🟠	crs_adam_0001_full\accuracy_validaton	0.8779	0	6/17/2021, 6:01:13 AM	3h 18m 36s

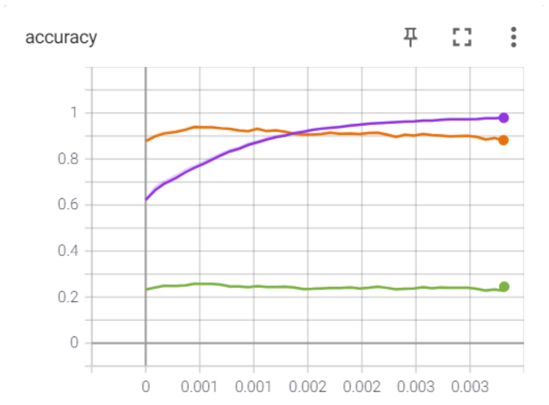


Figure 48 FITB, train and validation accuracy in 40 epochs

As can be seen, the training accuracy value exceeded the validation accuracy value over time. Considering the number of epochs and the larger training dataset, it can be said that this is expected. Fill-in-the-blank accuracy, on the other hand, does not seem to have experienced a large decrease or increase, although the epoch number is high. This is due to the difficulty of the fill-in-the-blank task as mentioned earlier in the study. Accuracy values were examined in more detail, and the changes of validation accuracy and fill-in-the-blank accuracy values over time were shown on the graph, respectively.

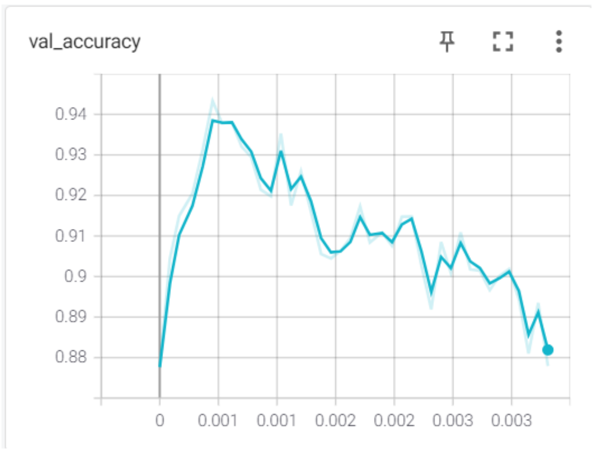


Figure 49 Validation accuracy in 40 epochs

Figure 52 shows the variation of the validation accuracy of the model over time. Although the value increased in the early stages, it started to decrease after about the 6th-7th epoch. This shows that the model may face the risk of overfit with too little data and too many epochs.

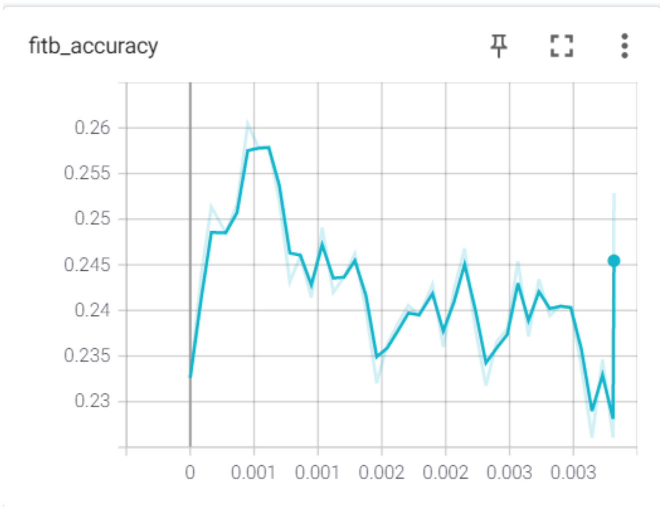


Figure 50 FITB accuracy in 40 epochs

The change in the fill-in-the-blank accuracy value was similar to the validation accuracy value. However, the range of variation of the value is not as large as the other metric. Therefore, it can be said that this value usually takes values between 0.23 and 0.26.

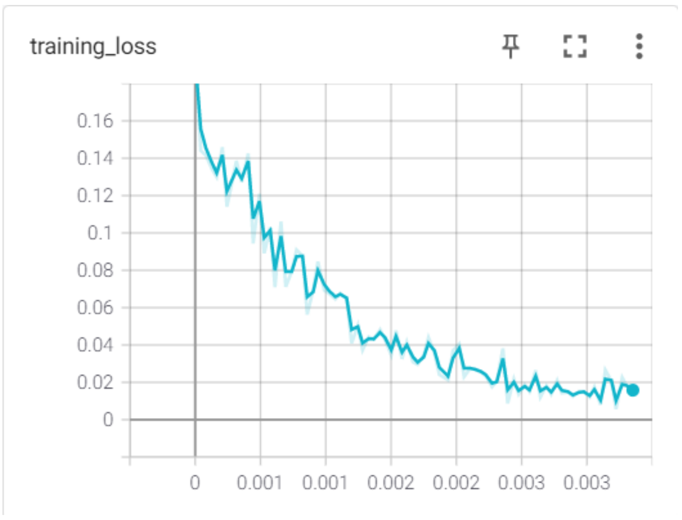
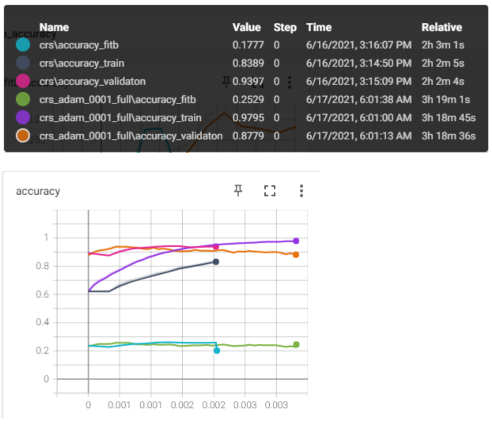


Figure 51 Training loss in 40 epochs

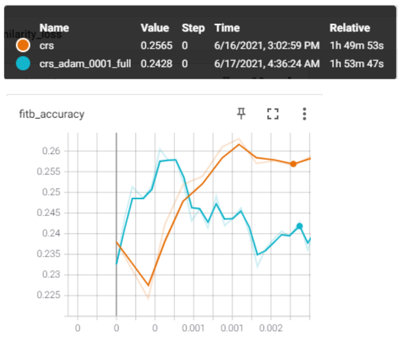
Training loss generally showed a decreasing trend during training. The fact that the training accuracy value has increased is compatible with this result. However, the positive change of these values does not indicate that the model is not in an overfit state. This can be observed in the testing and validation stages, with data that the model has not seen before.

In order to see whether the model is adversely affected by the increase in the epoch number, the results obtained in this section are compared with another model with the same parameters, which was run for 10 epochs. The graphs below show the change in accuracy, fill-in-the-blank accuracy and validation accuracy of the two models.



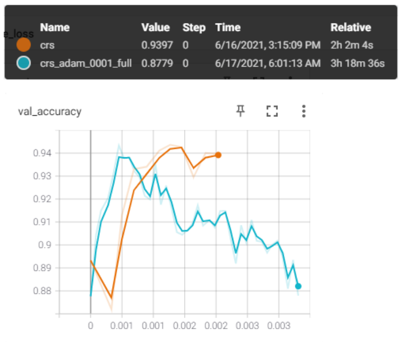
	Name	Value	Step	Time	Relative
●	crs\accuracy_fitb	0.1777	0	6/16/2021, 3:16:07 PM	2h 3m 1s
●	crs\accuracy_train	0.8389	0	6/16/2021, 3:14:50 PM	2h 2m 5s
●	crs\accuracy_validaton	0.9397	0	6/16/2021, 3:15:09 PM	2h 2m 4s
●	crs_adam_0001_full\accuracy_fitb	0.2529	0	6/17/2021, 6:01:38 AM	3h 19m 1s
●	crs_adam_0001_full\accuracy_train	0.9795	0	6/17/2021, 6:01:00 AM	3h 18m 45s
●	crs_adam_0001_full\accuracy_validaton	0.8779	0	6/17/2021, 6:01:13 AM	3h 18m 36s

(A)



	Name	Value	Step	Time	Relative
●	crs	0.2565	0	6/16/2021, 3:02:59 PM	1h 49m 53s
●	crs_adam_0001_full	0.2428	0	6/17/2021, 4:36:24 AM	1h 53m 47s

(B)



	Name	Value	Step	Time	Relative
●	crs	0.9397	0	6/16/2021, 3:15:09 PM	2h 2m 4s
●	crs_adam_0001_full	0.8779	0	6/17/2021, 6:01:13 AM	3h 18m 36s

(C)

Figure 52 Comparison of performance metrics of models trained with 10 epochs and 40 epochs. (A) Accuracy, (B) FITB Accuracy, (C) Validation Accuracy

It can be said that the model trained with 10 epochs gives better validation and fill-in-the blank accuracy values than the model trained with 40 epochs. The reason for this is that the model trained with little data starts to memorize the data it uses after a point and its generalization ability decreases. This overfit situation can be proved by a continuous increase in the training accuracy value but a decrease in the other values.

As a result of all these inferences, the model trained with 10 epochs with better generalization ability was chosen as the final model of the study. The parameters selected in the model are given in Table 18.

Table 18 Selected model parametes

Batch size	128
Epochs	10
Learning rate	0.0001
Margin	0.2
Optimizer	Adam

Here, the Margin value is the margin parameter used in the MarginRankingLoss function.

In the application developed for the user interface, the setup time of the model was determined as approximately 3.63 seconds, and the response time of the model according to the number of candidate products is shown in Table 19.

Table 19 The response time of the implemented model, depending on the number of candidate complementary outfits

# Candidates	Response Time (seconds)
5	0.454394820
25	0.515641212
50	0.971265554
75	1.432305336
100	1.999879122
200	5.950330973
500	6.532698631

Although the response time of the model gets longer with the increasing number of candidate products, it is considered as an expectable time for the user.

The responses of the model to the different candidate outfits is shown in Figure 56.

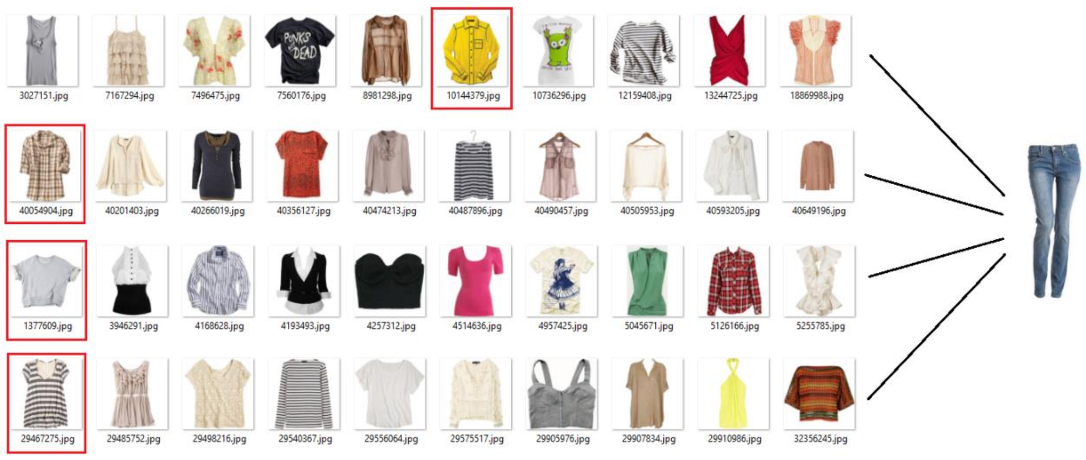


Figure 53 Different recommendations that the model gave for the reference outfit

In the next section, the performance of the model and the interpretation of the evaluation section will be discussed.

5.4 General Overview of the Model Performance

In this section, potential input data for the model we have conceptually designed are analyzed and various inferences are made. Then, the programming of the model and how it is presented to the users through an appropriate interface are mentioned; Finally, before the model was trained, it was tested with various parameters, and the parameters that could give the optimum value were determined among the results they gave depending on the performance metrics we determined. By assigning these determined parameters to the model, the performance values shown for larger harnesses were obtained and the system was evaluated from every aspect. It was observed that the model trained in high harnesses faced the risk of overfit and the model trained with 10 epochs was selected. The performance of the model over 10 epochs is given in the table below.

Table 20 Performance of the model, trained with 10 epochs

Epochs	General Loss	Average Accuracy	FITB Accuracy
1	0.1692	60.68%	22.40
2	0.1463	67.59%	24.20
3	0.1334	70.82%	25.20
4	0.1256	72.42%	25.40
5	0.1140	75.51%	26.10
6	0.1079	76.84%	26.30
7	0.0990	79.16%	25.70
8	0.0914	80.87%	25.70
9	0.0852	82.12%	25.60
10	0.0768	84.14%	25.90

However, the model was recorded during the training phase, where it gave the highest validation accuracy value of 94.36%, and the best generalization ability it showed in the training was used. In addition, the error value also showed a steady decrease over time.

However, like every model tested, the fill-in-the-blank accuracy values of the model we chose remained low. The main two reasons for this are that it is more difficult for the model to perform this task successfully compared to other tasks, and the dataset used is not large enough.

The responsiveness of the user interface is considered acceptable with current data.

In the next section, comments were made about the results of the study, the components that could be developed in the system for future studies and the opportunities before the study were mentioned.

CHAPTER 6

CONCLUSION & FUTURE WORKS

In this study, we designed and implemented a complementary clothing recommendation system by using visual and textual data with artificial neural networks. We aimed to provide a better understanding of the relationship between clothing pairs by giving importance to clothing types and considering the concepts of similarity and compatibility in different ways.

At the end of our study, we observed that with the model we designed, the compatibility and incompatibility between the clothes can be modeled depending on a certain context. With the interface we have created, users will be able to receive fashionable recommandations by selecting clothing images and candidate outfits.

In future studies, it is planned to increase the data size for the model to work with higher fill-in-the-blank accuracy and AUC (Area Under the Curve) value, and to investigate methods that will reduce the computational power of the model. In addition, the scope of the model can be expanded by adding men's clothing to the current dataset, which mainly includes women's clothing, and including data from different communities.

REFERENCES

[1] ecommerceDB. https://ecommercedb.com/en/markets/tr/all

[2] M. &. Company, The State of Fashion, 2020.

[3] J. P. a. Y. Y. J. Han, Mining frequent patterns without candidate generation, *ACM SIGMOD Record,* cilt vol. 29, no. 2, pp. pp. 1-12, 2000.

[4] A. P. L. H. U. a. D. M. P. A. I. Schein, Methods and metrics for cold-start recommendations,» *in Proc. 25nd Annu. Int. SIGIR Conf pp. 253-260*, 2002.

[5] Y. J. J. L. H. Zhang, A Triple Wing Harmonium Model for Movie Recommendation, *IEEE Transactions on Industrial Informatics, volume* 12(1), pp. 231-239, 2016.

[6] W. H. L. L. a. T. W. S. C. H. Zhang, «Learning to Match Clothing From Textual Feature-Based Compatible Relationships, *IEEE Transactions on Industrial Informatics,* cilt 16, no. 11, pp. pp. 6750-6759, Nov. 2020.

[7] C. P. a. J. M. R. He, «Learning Compatibility Across Categories for Heterogeneous Item Recommendation, *IEEE 16th International Conference on Data Mining (ICDM)*, Barcelona, 2016.

[8] B. K. A. Veit, Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences, CoRR, 2015.

[9] J.-Y. H. X. W. B. Z. Q. P. Guang-Lu Sun, Learning Fashion Compatibility across Categories with Deep Multimodal Neural Networks, Neurocomputing, 2019.

[10] M. I. Vasileva, B. A. Plummer, K. Dusad, S. Rajpal, R. Kumar ve D. Forsyth, «Learning Type-Aware Embeddings for Fashion Compatibility, *ECCV*, 2018.

[11] Y. L. Lin, S. D. Tran ve S. L. Davis, Fashion Outfit Complementary Item Retrieval, 2019.

[12] K. Falk, Practical Recommender Systems (1st ed.),Manning Publications, (2019).

[13] E. Yıldız, MÜŞTERİ SEGMENTASYONU ODAKLI BÜTÜNLEŞİK BİR ÜRÜN ÖNERİ SİSTEMİ: MODA PERAKENDESİ SEKTÖRÜNDE BİR UYGULAMA, (2018).

[14] A. A. &. E. M. Kardan, *A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups*, Information Sciences, (2013), pp. 219, 93–110.

[15] Z. &. A. .. M. S. Duzen, An Approach To Hybrid Personalized Recommender Systems, 2016.

[16] C. C. Aggarwal, Recommender Systems: The Textbook (1st ed. 2016 ed.), Springer, (2016).

[17] M. &. L. Thiele, Setting Goals and Choosing Metrics for Recommender System Evaluations, 2017.

[18] Ç. Elmas, Yapay Zeka Uygulamaları, Yapay Sinir Ağları – Bulanık Mantık– Genetik Algoritma, Ankara: Seçkin Yayinevi, (2012).

[19] E. Öztemel, Yapay Sinir Ağları, İstanbul: Papatya Yayıncılık, (2003).

[20] [Çevrimiçi]. Available: https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network.

[21] R. F. &. Y. F. &. A. M. K. A. &. H. J. Turkson, Artificial neural network applications in the calibration of spark-ignition engines: An overview., *Engineering Science and Technology, an International Journal,* cilt 19, no. (3), p. 1346–1359, 2016.

[22] İ. Çayıroğlu, İleri Algoritma Analizi-5 Yapay Sinir Ağları, 2015.

[23] B. Ataseven, Yapay Sinir Ağları İle Öngörü Modellemesi, *Öneri Dergisi , volume* 10, no. (39).

[24] D. E. &. H. G. E. M. J. L. Rumelhart, A General Framework for Parallel Distributed Processing, Stanford University, 1886.

[25] S. C. Wang, Interdisciplinary computing in java programming, *Artif Neural Network*, 2003, pp. Part II,pp.81-100.

[26] S. &. M. T. A. .. &. A.-Z. S. Albawi, Understanding of a Convolutional Neural Network, 2017.

[27] A. G. &. W. W. &. Z. M. &. C. B. &. K. D. &. W. T. &. A. M. &. A. H. Howard, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision, 2017.

[28] L. R. &. G. T. W. &. W. B. Weatherford, Neural network forecasting for airlines: A comparative analysis. Journal of Revenue and Pricing Management, volume 1, no. (4), p. 319–331, 2003.

[29] E. Kabalcı, Artificial neural networks. Course Notes, 2015.

[30] X.-S. Zhang, Feedback Neural Networks,*Neural Networks in Optimization*, p. 137–175.

[31] W. S. &. P. W. McCulloch, A logical calculus of the ideas immanent in nervous activity, *The Bulletin of Mathematical Biophysics, volume* 5, no. (4), p. 115–133, 1943.

[32] D. O. Hebb, The organization of behavior; a neuropsychological theory., Wiley , 1949.

[33] R. F, Principles of Neurodynamics, New York: Spartan books, 1962.

[34] &. M. H. J. Widrow, Adaptive Switching Circuits, *IRE WESCON Convention Record,* pp. 4:96-104, 1960.

[35] A. G. a. L. V. G. Ivakhnenko, Cybernetic Predicting Devices, CCM Information Corporation, 1965.

[36] M. &. P. S. A. Minsky, Perceptrons: An Introduction to Computational Geometry, 1969.

[37] T. Kohonen, The self-organizing map, *IEEE,* cilt 78, no. 9, pp. pp. 1464-1480, Sept. 1990.

[38] J. Schmidhuber, Learning to control fast-weight memories: An alternative to dynamic recurrent networks. Neural Computation,volume 4, no. (1), p. 131–139, 1992.

[39] G. &. O. S. &. T. Y.-W. Hinton, A Fast Learning Algorithm for Deep Belief Nets. Neural computation.18. 1527-54.

[40] J. N. A. Y. Dean, Large Scale Distributed Deep Networks, Google Inc, Mountain View, CA , 2012.

[41] U. M. a. J. S. D. Ciregan, Multi-column deep neural networks for image classification pp. 3642-3649,*IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI,*, 2012.

[42] A. G. &. W. W. &. Z. M. &. C. B. &. K. D. &. W. T. &. A. M. &. A. H. Howard, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision, (2017).

[43] C. T. Q. S. a. A. v. d. H. J. McAuley, Image-based recommendations on style and substitutes,*Proceedings of the 38st annual international ACM SIGIR conference*, 2015.

[44] X. D. W. Z. a. L. Z. Zhengzhong Zhou, Fashion Sensitive Clothing Recommendation Using Hierarchical Collocation Model. In Association for Computing Machinery, New York, NY, USA, 2018.

[45] X. &. W. B. &. Y. Y. &. Z. Y. Wang, Outfit Compatibility Prediction and Diagnosis with Multi-Layered Comparison Network., 2019.

[46] X. Z. S. R. a. J. S. Kaiming He, «Deep residual learning for image recognition,» *In Proceedings of the IEEE conference on computer vision and pattern recognition 770–778.*, 2016.

[47] X. &. W. Z. &. J. Y.-G. &. D. L. Han, Learning Fashion Compatibility with Bidirectional LSTMs, 2017.

[48] V. V. S. I. J. S. a. Z. W. Christian Szegedy, Rethinking the Inception Architecture for Computer Vision, 2015.

[49] B. Klein, G. Lev, G. Sadeh ve L. Wolf, Fisher Vectors Derived from Hybrid Gaussian-Laplacian Mixture Models, 2014.

RESUME

PERSONAL INFORMATIONS

Name	: İrem ATILGAN
Birth Date and Place	: 18.09.1999 MALTEPE
Foreign Language	: English
E-mail	: irematilgan99@gmail.com

EDUCATION

Degree	**Field**	**School / University**	**Graduation Year**
Bachelor's degree	Industrial Engineering	Yildiz Technical University	2021(Estimated)
High school	High School	Istanbul Anatolian High School	2017

JOB EXPERIENCE

Yıl	**Firma/Kurum**	**Görevi**
2020	Yapı Kredi Technology	Customer Database Management Intern

PERSONAL INFORMATIONS

Name	: Gökhan Köse
Birth Date and Place	: 06.01.1998 ÜMRANİYE
Foreign Language	: English
E-mail	: gokhanksee@gmail.com

EDUCATION

Degree	**Field**	**School / University**	**Graduation Year**
Bachelor's degree	Industrial Engineering	Yildiz Technical University	2021
High school	High School	Maltepe College	2016

JOB EXPERIENCE

Yıl	**Firma/Kurum**	**Görevi**
2021	Dogus Automotive	Human Recources Intern
2020	Biselam	Marketing Intern
2019	Samm Technology	IT Intern
2018	Ulus Production	Production Intern