A PROJECT REPORT ON

"FASHION RECOMMENDATION SYSTEM"

submitted in fulfilment of the requirements for the degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted By,

Name: Ritik Raj Kumar

Roll No.: 16500119022

Registration: 007437 OF 2019-20

Under the supervision of
AMRITA BHATTACHARYA
(Assistant Professor, Department of CSE, CIEM)

Calcutta Institute of Engineering and Management
(Affiliated to MAKAUT)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Kolkata, West Bengal, India-700040

DECLARATION

I hereby declare that the project work being presented in the project proposal entitled **"FASHION RECOMMENDATION SYSTEM"** in fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING** at **CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT** is carried out under the supervision of **AMRITA BHATTACHARYA.** The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 25/05/2023

Name of the Student: Ritik Raj Kumar

Signature of the Student: Ritik Raj Kumar.

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who make it possible and whose constant guidance and encouragement crown all the efforts with success. The acknowledgement transcends the reality of formality when I would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped me with the completion of my project work.

This project has been guided to its successful completion by constant inspiration and hearty cooperation from my Project Guide ***Amrita Bhattacharya****. I am grateful to her for her thoughtful advice, insight and detailed suggestions. She has constantly given me support and encouragement. I must mention that without her this project would have never seen the light of the day.*

It is a great pleasure to thank the head of the department, ***Mr. Abhijit Mitra*** *of the Computer Science & Engineering department, for his help and constructive criticisms.*

I am also indebted to all the faculty members of the department and to all our batchmates who made my stay a memorable experience. The laboratory of the department has provided an ideal work environment.

Regards,

Ritik Raj Kumar

CIEM

May 2023

CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT

Department of Computer Science & Engineering

CERTIFICATE OF APPROVAL

This is to certify that the work in preparing the project entitled **"FASHION RECOMMENDATION SYSTEM"** has been carried out by **Ritik Raj Kumar** under my guidance during the session 2023 and accepted in fulfilment of the requirement for the degree of Bachelor of Technology.

_____

(Signature of Project Guide)

_____

(Signature of HOD)

_____

(Signature of Principal)

CONTENTS

LIST OF FIGURES

Figures **Page No.**

LIST OF TABLES

Tables **Page No.**

ABSTRACT

In recent years, the textile and fashion industries have witnessed an enormous amount of growth in the fashion industry. On e-commerce platforms, where numerous choices are available, an efficient recommendation system is required to efficiently convey relevant product content or information to users. Artificial intelligence is becoming increasingly useful in creating personalised shopping experiences on e-commerce websites, user-specific advertisements, classifying objects, and detecting colours from images. Fashion is one of the most crucial industries in our world nowadays, style has become one of the primary ways in which people express their personality among others. The fashion industry has seen a significant shift towards personalisation and online shopping. In this context fashion recommendation systems have emerged as valuable tools to assist users in finding clothing items based on their interests.

Image-based Fashion Recommendation Systems (FRSs) have attracted a huge amount of attention from fast fashion retailers as they provide a personalised shopping experience to consumers. With technological advancements, this branch of artificial intelligence exhibits a tremendous amount of potential in image processing, parsing, classification, and segmentation. Humans are inevitably drawn towards something that is more attractive, this tendency of humans has led to the development of the fashion industry over the course of time. However, giving too many options of products on an e-commerce website has presented new challenges to customers in identifying their correct outfit choice.

Thus, in this project, we are creating a Fashion Recommendation System using Convolutional Neural Networks (CNNs) to recommend the most suitable products according to the user's input using a recommendation algorithm. The proposed system shows that it can process the user's clothes from the images, identify the type and colour of the outfit and finally recommend the most similar outfit to the users based on their choices. Overall the fashion recommendation system offers a personalised and efficient solution for users seeking fashion guidance. Through the integration of machine learning algorithms, the system aims to provide accurate and relevant fashion recommendations to its users.

Keywords: *Image-based Fashion Recommendation Systems (FRSs), Machine Learning, E-commerce, Convolutional Neural Networks (CNNs), ResNet, K-Nearest Neighbours (KNN)*

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION TO THE PROJECT

The fashion industry is one of the largest industries in the world[4]. One of the things that have remained constant throughout human civilization is humans covering their bodies with a piece of cloth. Initially, this cloth was worn as protection from the harsh climates in those ages. Later on, as we humans learnt to fend off the unforgiving climates, the cloth started to serve a different purpose. Clothing is a kind of symbol that represents people's internal perceptions through their outer appearance. Therefore, clothing is believed to be a nonverbal way of communicating and a major part of people's outer appearance. Fashion these days showcases the individuality of the person. Many things can be said about a person based on their fashion sense. It is the generation where everything is available online. But the confusion is what to choose and how to choose. There is a variety of outfits available in the market for various occasions. The internet has also provided various applications from where we can get suggestions online. We can exclusively choose the dress among the diverse variety.

A recommendation system is referred to as a decision-making approach for users in a multidimensional information environment. It has also been defined as an e-commerce tool, which helps consumers search based on the knowledge that is related to a consumer's choices and preferences.

The proposed Fashion Recommendation System[8][9][10] is intended to be used by individual users to upload images of the clothes that our e-commerce site owns in what is called a digital wardrobe and from the database, the system recommends the most appropriate and similar clothes to the user's uploaded one. It is also helpful in recommending products based on user searches. It is a technology that takes an image file as an input query and returns results related to the image. The work of this project is based on combining deep learning models (*i.e., CNN*) to detect the similar type, colour, and design of the clothing in the given image.

The approach followed in this system is using Convolutional Neural Networks (CNNs)[3], a Deep Learning algorithm[1][5][6] that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The main aim of the project is to recommend the most appropriate fashion products for a given input based on the clothes existing in our wardrobe to relieve the user's burden of making decisions. Recommended clothes by the system are accomplished based on the type, colour, and design of the clothing that is uploaded in our system. The system is capable of handling basic clothing and other wearables types: Shirts, T-Shirt, Jeans, Tops, Skirts, Jackets, Shoes, Watches, Sunglasses, Saree, Tops, etc.

Here, the model that we have used is the ResNet[1] mostly to solve a complex problem, we stack some additional layers in the Deep Neural Networks which results in improved accuracy and performance[1][6]. The intuition behind adding more layers is that these layers progressively learn more complex features. *For example*, in the case of recognising images, the first layer may learn to detect edges, the second layer may learn to identify textures and similarly the third layer can learn to detect objects and so on. But it has been found that there is a maximum threshold for depth with the traditional Convolutional Neural Network model[3]. The resultant accuracy of the recommendation recommended by the system is above 90%.

1.2. AIM AND OBJECTIVES

1.2.1. Aim of the Project

The aim of our project named ***"Fashion Recommendation System"*** is to develop a robust recommendation system that can suggest fashion items to users based on their searches and inputs. With the rapid growth of Artificial Intelligence and its integration with various sectors, it has become a necessity for various organisations to keep up with current technologies and utilise this technology to enhance their customer's shopping experience[4].

Firstly, the system will help the users to find the exact item that they want or will suggest something similar to it in case of the non-availability of that particular item in our product database. And secondly, the system will recommend similar products based on users' searches.

1.2.2. Objectives of the Project

The main objectives of the project can be summarised in a few points given below:

- **Data Collection:** Gather a diverse dataset of different types of fashion items for both men and women.
- **Data Compression:** Scale down the images present in the dataset to a lower dimension before feeding it to the CNN model for the feature extraction process.
- **Data Preprocessing:** Some predefined processes are to be applied to the dataset for the preprocessing before sending it to the ResNet50 model.
- **Feature Selection:** Identify the most relevant features for fashion recommendation for every single product.
- **KNN Model Training:** Implement the KNN algorithm and train the model using the preprocessed dataset, considering appropriate distance metrics and tuning the value of K.
- **Similarity Calculation:** Apply the trained KNN model to compute the similarity between the user's input and fashion items in the dataset using the Euclidean distance function and, considering relevant features.
- **Recommendation Generation:** Generate personalised fashion recommendations by selecting the K-nearest neighbours[9][10] to the user's input based on similarity scores.
- **Evaluation:** Assess the performance of the recommendation system using metrics like accuracy, precision, recall, and user satisfaction through surveys or feedback analysis.
- **Optimization and Refinement:** Continuously improve the system by refining the feature selection, tuning hyperparameters, and incorporating user feedback to enhance the quality and relevance of recommendations.
- **Deployment:** Implement the fashion recommendation system in a

user-friendly interface, making it accessible to users for easier understanding of the working system.

1.3. SUPERVISED LEARNING

Supervised learning is a machine learning technique in which an algorithm learns from labelled training data to make predictions or decisions. In supervised learning, the dataset used for training consists of input features and corresponding target labels. The goal is to train a model that can generalise well to unseen data and accurately predict the target labels for new input instances.

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

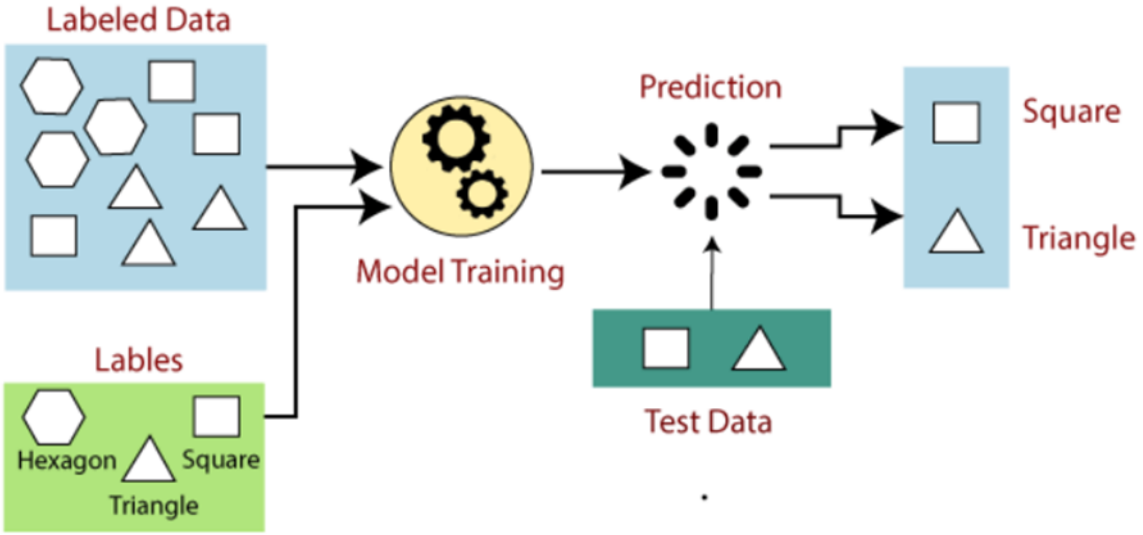


Figure 1.1.: Working of Supervised Learning Model

Steps Involved in Supervised Learning:

- First, determine the type of training dataset.
- Collect/gather the labelled training data.
- Split the training dataset into the training ***dataset, test dataset, and validation dataset***.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as brute force, k-d tree, ball tree etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of Supervised Machine Learning Algorithms:

Supervised learning can be further divided into two types of problems:

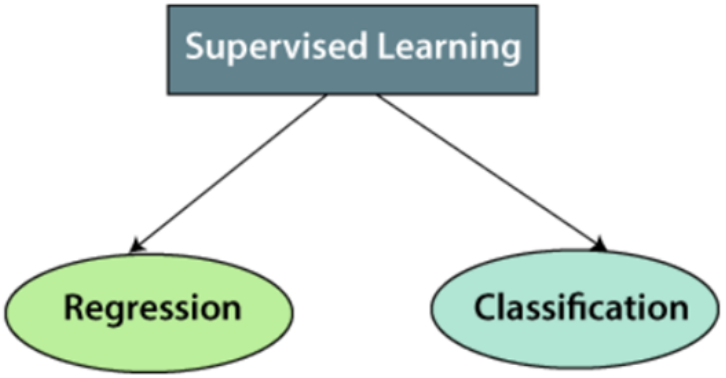


Figure 1.2.: Types of Supervised Learning

1.3.1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.

1.3.2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes as Yes-No, Male-Female, True-false, etc.

1.4. DEEP LEARNING

Deep learning is a subfield of machine learning that focuses on training artificial neural networks to learn and make predictions or decisions[7][11]. It is inspired by the structure and function of the human brain, with the goal of creating models that can automatically learn representations of data through multiple layers of abstraction.

Deep learning models, often called *deep neural networks*[11], are composed of interconnected layers of artificial neurons, also known as nodes or units[1][6]. Each layer receives input from the previous layer and applies mathematical operations to transform the input. The output of each layer serves as the input for the next layer[5], allowing the network to learn hierarchical representations of the data.

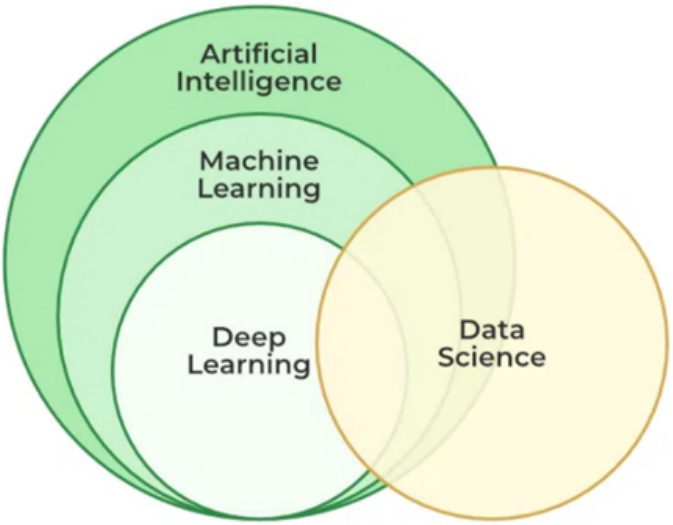


Figure 1.3.: Relationship between AI, ML, DL and DS

Specific fields in which deep learning is currently being used include the following:

- **Customer Experience (CX)**: Deep learning models are already being used for chatbots. And, as it continues to mature, deep learning is expected to be implemented in various businesses to improve CX and increase customer satisfaction.
- **Text Generation:** Machines are being taught the grammar and style of a piece of text and are then using this model to automatically create a completely new text matching the proper spelling, grammar and style of the original text. ***Example:*** *ChatGPT, Bard AI*
- **Aerospace and Military:** Deep learning is being used to detect objects from satellites that identify areas of interest, as well as safe or unsafe zones for troops.
- **Industrial Automation:** Deep learning is improving worker safety in environments like factories and warehouses by providing services that automatically detect when a worker or object is getting too close to a machine.
- **Adding Colour:** Colour can be added to black-and-white photos and videos using deep learning models. In the past, this was an extremely time-consuming, manual process.
- **Medical Research.** Cancer researchers have started implementing deep learning into their practice as a way to automatically detect cancer cells.

1.5. HISTORY AND OVERVIEW OF RECOMMENDATION SYSTEM

The era of recommendation systems originally started in the 1990s based on the widespread research progress in Collective Intelligence[12]. During this period, recommendations were generated by consumers based on their rating structure. The first consumer-focused recommendation system was developed and commercialised by Goldberg,

Nichols, Oki and Terry in 1992. An electronic messaging system was developed to allow users only to rate messages as either a good or bad product or service. However, now there are plenty of methods to obtain information about the consumer's liking for a product through the Internet. These data can be retrieved in the forms of voting, tagging, reviewing and the number of likes or dislikes the user provides. Regardless of communication and presentation, medium preferences are expressed in the form of numerical values. *Table 1.1.* presents the history of the progress of fashion recommendation systems over the last few decades.

Year	Recommendation System Approach	Properties
Before 1992	Mafia, developed in 1990	• Content filtering. • Mail filtering agent for providing a cognitive intelligence-based service for document processing.
1992 to 1998	Tapestry, developed in 1992	• Collaborative filtering. • Developed by Palo Alto. • Allowed users only to rate messages as either good or bad product and service.
	Grouplens, first used in 1994	• Rate data to form the recommendation.
	Movielens, proposed in 1997	• Useful to construct a well-known dataset.
1999 to 2005	PLSA (Probabilistic Latent Semantic Analysis), proposed in 1999	• Developed by Thomas Hofmann. • Collaborative filtering.
2005 to 2009	Several Latent Factor Models such as Singular Value Decompositions (SVD), Robust Singular Value Decomposition (RSVD), Normalized Singular Value Deviation (NSVD).	• Collaborative filtering approach. • Find out factors from rating patterns.
2010 to onwards	Context-aware-based, instant-personalization-based	• Combined techniques of content and collaborative approach.

Table 1.1.: History of Recommendation Systems[12]

E-commerce retailers started implementing fashion recommendation systems in the early 2000s. However, implementation was mostly in the development stage until 2007–2008. As with other products such as electronics and books, fashion products were also recommended based on the user's previous purchase history.

1.6. STEPS INVOLVED IN RECOMMENDATION

The steps involved in building a recommendation system can vary depending on the specific approach and techniques used[5][8]. The steps followed in our project for the recommendation system can be summarised in some steps given below:

1.6.1. Extract features of every product's image present in the dataset using a Convolutional Neural Network (ResNet50)[1] and stored them in the form of an array.
1.6.2. When a user provides an input image to the system, its features are extracted and stored locally.
1.6.3. Each image has 2048 features and these features are plotted in a multi-dimensional plane.
1.6.4. Using the Euclidean distance function from K-Nearest Neighbours (KNN), distance is calculated with every vector against our input.
1.6.5. Five closet vectors are returned as the outputs to users and these are the most similar products to the input product.

1.7. BRIEF DISCUSSION ON THIS PROJECT

We chose this project named ***"Fashion Recommendation System"*** as we understand the importance of a good recommendation in the field of the fashion industry. With the development of artificial intelligence and integration of it with various fields such as Healthcare, Entertainment Industry, Transportation, etc. It is the need of E-commerce businesses to use the advantages of machine learning to give their users the utmost level of shopping experience. *For example*, when a user is searching for a product, the system should recommend similar items that will give multiple options for the user to select from. And, also when they upload a product's image to an e-commerce website, the user would get the same product or similar products to it in case of unavailability.

There are multiple phases involved in the recommendation system that develop the foundation of any state-of-the-art recommendation system[10][12]. These are defined as the information collection phase, the learning phase, and the recommendation phase. *Figure 1.4.* shows the interrelationship of these phases involved in the recommendation process. It shows that information collection is the initial stage of the Recommendation System (RS), which is followed by the learning phase and the recommendation phase. The recommendation provided in the last phase can be generated based on information gathered during the information collection phase.

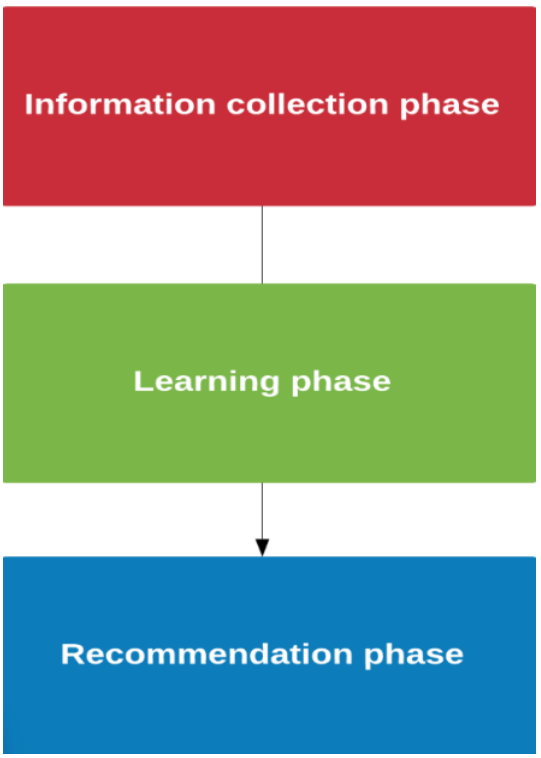


Figure 1.4.: The Recommendation Phase

CHAPTER 2

LITERATURE SURVEY

2.1. INTRODUCTION

A literature survey is a systematic and comprehensive review of existing scholarly research and publications on a specific topic. It serves as a crucial step in conducting research as it helps researchers gain a comprehensive understanding of the current state of knowledge, identify gaps, and highlight areas for further investigation. In the field of fashion recommendation systems, a literature survey plays a vital role in summarizing and analyzing the existing research on the topic.

The purpose of this literature survey is to explore and examine the existing literature on fashion recommendation systems that utilize Convolution Neural Network (CNN) and the K-Nearest Neighbors (KNN) algorithm. By conducting a thorough review of relevant studies, methodologies, and findings, this survey aims to provide insights into the effectiveness, challenges, and potential improvements of the CNN-KNN fusion model in the context of fashion recommendation systems.

The survey will delve into the advancements and state-of-the-art approaches in this field, analyzing the strengths and limitations of the CNN-KNN fusion model for fashion recommendation. It will also identify gaps and areas where further research is needed, providing a foundation for future studies and innovations. By consolidating and synthesizing the existing literature, this survey will contribute to the overall understanding and development of fashion recommendation systems using CNN and KNN.

2.2. LITERATURE SURVEY

Shinya M et.al. [1] The paper proposed a retrieval technique for fashion-related images on websites. It's quite difficult to find images of people wearing clothing from images of clothing. This is due to the wide range of differences between clothing images) and those in the fashion coordinate images). Conventional image retrieval methods are inapplicable in this case. As a result, we separate fashion imagery from other critical factors. The full-body fashion coordinate image is divided into four areas, and an image is returned that includes a similar clothing image to the query in the target area.

Congying Guan., & Shengfeng Qin [2] came up with a clever suggestion for the International Journal of Clothing Science and Technology. It's domain expertise knowledge of mixing and matting criteria facilitates exploring the interrelationship between fashion and the user through the usage of intelligent algorithms. To learn the talent of clothing attribute evaluation, they employed decision trees, analytical hierarchy process, sensory engineering, fuzzy math, genetic algorithms, neural networks, and support vector machines. Application of interactive evolutionary algorithms and the implementation of expert rules to construct an intelligent fashion suggestion system based on eye gaze monitoring and the prediction of users' style preferences.

Jun Xiang et al. [3] In this, the recommendations are based on previous sales, clothing purchase data, eye movement records, and item click rate. It delivers outfit ideas

based on the user's preferences and interests, utilizing an analytical hierarchy method. The CNN Algorithm can be used in conjunction with feature extraction and image classification to aid in the retrieval of comparable picture products.

Guan, et al. [4] developed a content-based filtering algorithm using CNN (Convolutional Neural Network). Using image features, the recommendation algorithm produced weather-related outfit pairing recommendations. On the Normalized Discounted Cumulative Gain (NDCG) ranking scale, the proposed Convolution Neural Network model received a maximum score of 0.50, exceeding the support vector machine (SVM), which received a score of 0.45.

Leininger et al. [5] The researchers propose a retail recommendation system using kNN and collaborative filtering approaches. Distance is calculated using cosine similarity between related things, then individual products are clustered. The accuracy in terms of AUC (91 per cent) was higher than the baseline model's AUC (85 per cent).

The most effective way to increase sales for a company is to provide recommendations to its user. The paper showcased both collaborative filterings as the base model and an ensemble model as a novel approach. Even though the novel approach model performed better but it consumed significantly more time to run. Thus, collaborative filtering is a more economical approach for the problem stated in the paper.

Sr. No.	Author(s)	Method	Advantage	Disadvantage
1.	Miura, Yamasaki, Aizawa	Fashion Pairing Recommendation System - Image detection technology is used to extract fashion styles with similar features.	Assists users in determining what kind of coordinates are appropriate for their clothing and in purchasing appropriate items.	It can't handle a change in a person's pose or figure.
2.	Congying Guan, Shengfeng Qin	Smart or intelligent recommendation – Its domain expertise knowledge of stirring and corresponding criteria enables intelligent algorithms to explore the inter-relationship between fashion and the user.	Produces highly accurate predictions on apparel.	NA
3.	Jun Xiang		**1.** A fashion sketch contains more information than a text. **2.** Sketch is a simple way to express fashion image styles without ambiguity.	NA

4.	Guan and Liu	Used CNN to develop a content-based filtering technique.	**1.** Image recognition problems require a high level of accuracy. **2.** Without any human intervention, it automatically detects the important features.	**1.** Lots of training a data are required. **2.** A lar amount of training data is required.
5.	Leininger	The KNN model calculated the distance to similar items using cosine similarity, accompanied by individual product clustering.	**1.** The algorithm is simple to comprehend and implement. **2.** It can quickly adapt to new training data.	**1.** The speed at which Calculations are performed rapidly decreases as your training data grow. **2.** If the model is not correctly chosen, it will be underfitted or overfitted to the data.

Table 2.1.: Summary of Literature Survey

These literature sources cover a range of topics, including recommender systems, fashion retailing, consumer behaviour, and emerging technologies in the fashion industry. They provide valuable insights, methodologies, and references to support our final year project *"Fashion Recommendation System"*.

CHAPTER 3

FEATURE EXTRACTION

3.1. INTRODUCTION

Feature extraction using CNNs involves passing an image through the network and extracting the output from one of the intermediate layers[1][3][5]. This output is a high-dimensional representation of the image that encodes the features that the network has learned to recognise.

The choice of which layer to extract features from depends on the task at hand. *For example*, if the goal is to recognize high-level concepts such as objects or scenes, it may be beneficial to extract features from a later layer in the network. On the other hand, if the goal is to recognize low-level features such as edges or textures, it may be better to extract features from an earlier layer.

Once the features have been extracted, they can be used for various tasks such as image classification, object detection, or image retrieval. The extracted features can also be fed into another machine-learning model for further processing.

CNNs are powerful tools for feature extraction. By providing an image through the network and extracting the output from an intermediate layer, we can obtain a high-dimensional representation of the image that encodes the features that the network has learned to recognise.

3.2. DATASET USED

It was a big challenge for us to gather a lot of data for training our model as the CNN model works on a vast amount of data. Web scraping was not possible for us as we needed thousands of images for our project and it might have required a lot of time to do so. We overcame this big hurdle when we learned about a website called *Kaggle*, a subsidiary of Google LLC, the world's largest data science community. We found the perfect dataset[13] for training our model here. It consists of around 44K images of different categories of fashion products[13]. This dataset is publicly available to everyone, making it easier for students or data scientists who want to train their models on real-world data.

Figure 3.1.: Representation of different products in the dataset

The dataset has both men's and women's products, and other unisex wearables[13]. Every product has a white background which makes feature extraction faster by reducing unnecessary details. There are many products in the dataset like T-shirts, Shirts, Jeans (Men & Women), Watches (Men & Women), Shoes, Saree, Kurti, Wallets, Handbags and many more.

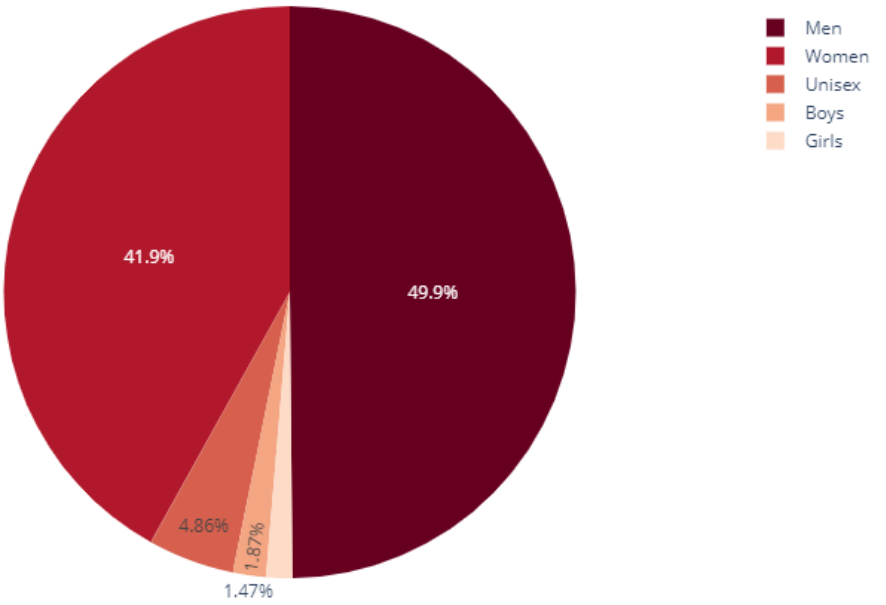


Figure 3.2.: Number of Men's items vs. Number of Women's items

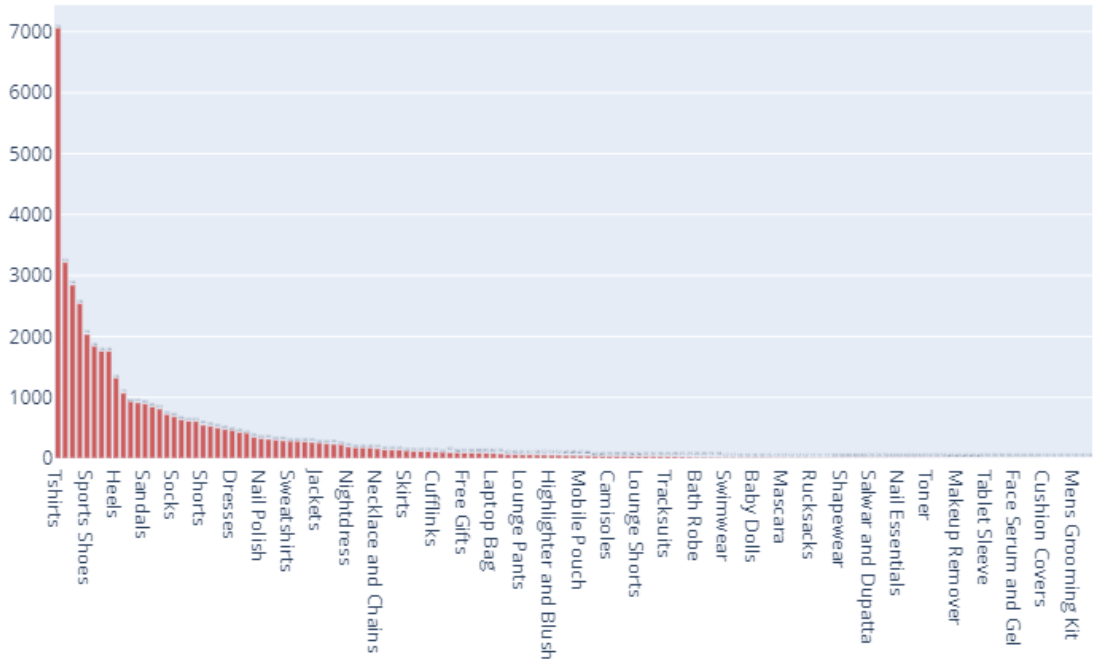


Figure 3.3.: Graphical representation for different products in the dataset

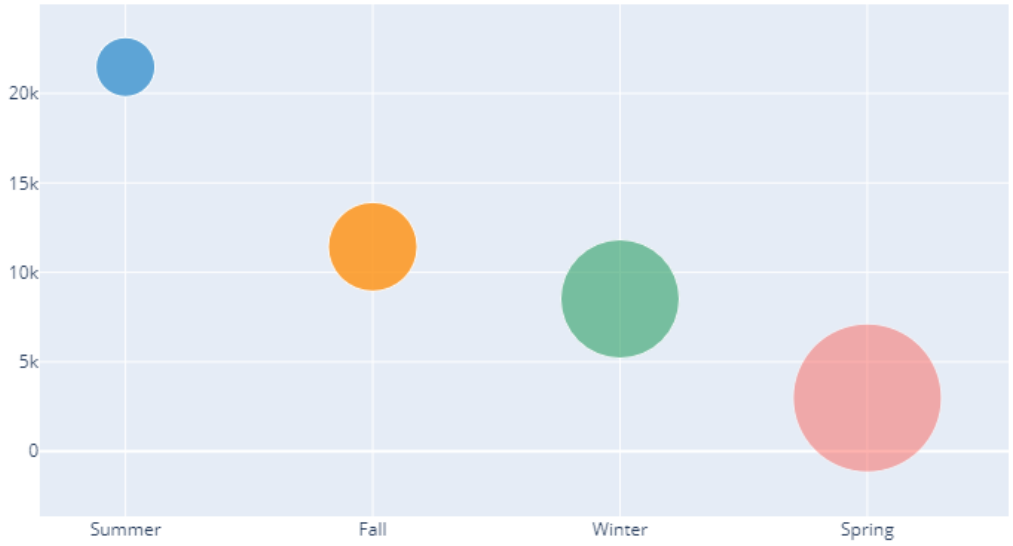


Figure 3.4.: Amounts of products according to different seasons

3.3. NEURAL NETWORKS

A ***neural network***[1] is a type of machine learning algorithm that is modelled after the structure and function of the human brain. It consists of layers of interconnected nodes or *"neurons"* that process and transmit information[6][9]. A neural network is a powerful and versatile machine-learning model inspired by the biological structure of the human brain. It is capable of learning and making predictions based on input data.

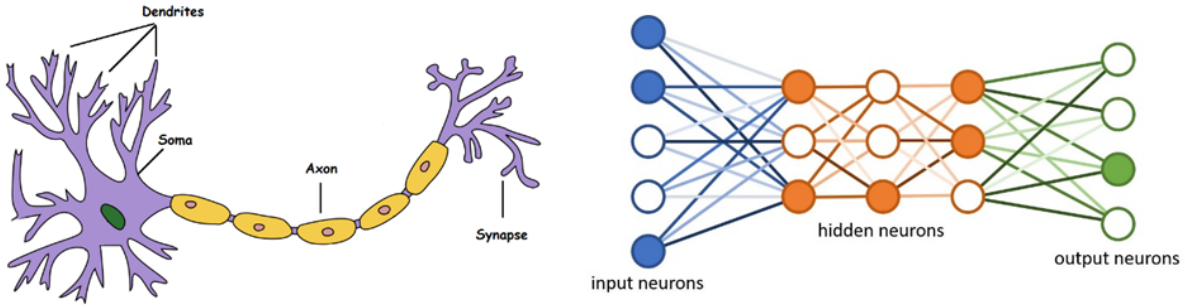


Figure 3.5.: Living Neuron vs. Artificial Neuron

The first layer of a neural network is called the *input layer*, where data is fed into the network. The data then passes through one or more hidden layers, where the neurons perform calculations and transformations on the data. The final layer is called the *output layer*, where the network produces its predictions or classifications.

Neural networks are trained using large amounts of data and a process called *backpropagation*[6]. During training, the network adjusts the weights and biases of its neurons to minimise the error between its predictions and the true values[7]. Once trained, a neural network can be used to make predictions on new data.

Neural networks have been successful in a wide range of applications, including image recognition, natural language processing, and predictive modelling[9]. They are particularly well-suited for tasks where the relationship between the input data and the desired output is complex and difficult to model using traditional techniques.

There are many variations of neural networks, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and Artificial neural networks (ANNs)[1].[9]. Each type has its own strengths and is suited for different types of problems.

Neural networks have found applications in various domains, including computer vision, natural language processing, speech recognition, recommender systems, and finance. They have achieved remarkable success in image classification, object detection, machine translation, sentiment analysis, and many other tasks.

Neural networks are a fundamental component of modern machine learning and have

revolutionised the field with their ability to learn and make complex predictions. Understanding the architecture, training process, and various components of neural networks is essential for effectively applying them to real-world problems.

3.4. CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional Neural Networks (CNNs) are a specialised type of deep learning model designed for processing structured grid-like data, such as images and videos. Inspired by the organisation of the visual cortex in animals, CNNs are capable of automatically learning hierarchical representations from raw pixel values[3]. This ability enables them to capture spatial dependencies and extract meaningful features, making CNNs highly effective in tasks like image classification, object detection, and image segmentation.

A convolutional neural network (CNN)[3] is a type of neural network that is commonly used in image recognition and processing tasks. CNNs are designed to take in input data in the form of images and process the data through multiple layers, each of which applies a different set of filters to the data to extract different features.

The architecture of a CNN is designed to take advantage of the 2D structure of an input image. This is achieved by using a series of convolutional[1] and pooling layers[1] that process the data in a hierarchical manner. The convolutional layers apply filters to the data to detect features such as edges, corners, and objects. The pooling layers then downsample the data to reduce its dimensionality. After passing through the convolutional and pooling layers, the data is fed into one or more fully connected layers, where the final classification or prediction is made.

3.4.1. CNN Architecture and Components

Convolutional Neural Networks (CNNs) have revolutionised the field of computer vision and image processing tasks, enabling remarkable advancements in areas such as image classification, object detection, and image segmentation. This section will provide an in-depth understanding of CNN architecture and its components.

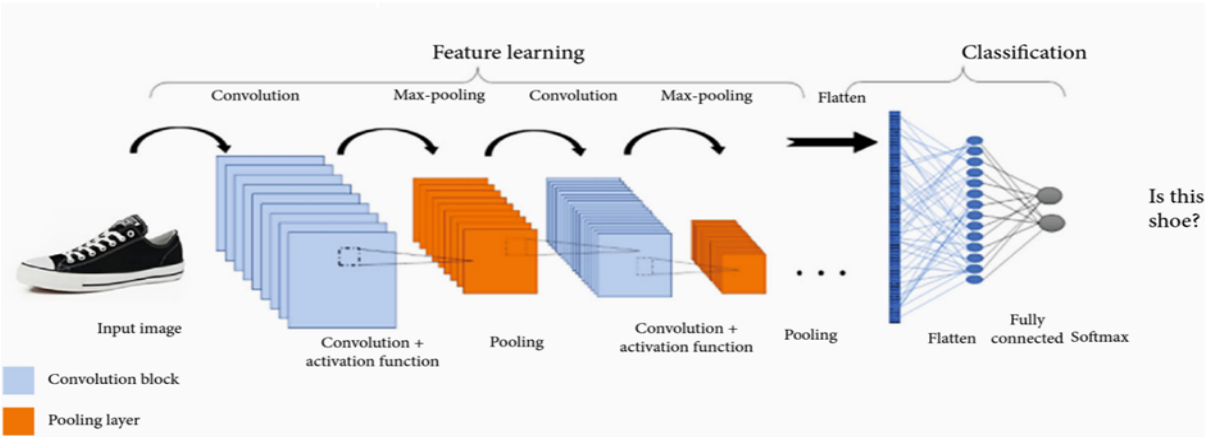


Figure 3.6.: CNN Architecture

3.4.1.1. Convolutional Layers

Convolutional layers are the core building blocks of CNNs. These layers perform the convolution operation, which involves sliding small filters (also known as kernels) over the input data to extract relevant features. Each filter learns to detect specific patterns or edges present in the data. The size and number of filters can be adjusted to control the complexity and expressiveness of the network.

The convolution operation works by taking the element-wise multiplication of the filter with the corresponding region of the input, followed by summation to produce a single value in the output feature map[1][8]. By sliding the filter across the entire input, multiple feature maps are generated, each representing a different learned pattern. This process allows CNNs to capture local patterns and spatial dependencies present in the input data.

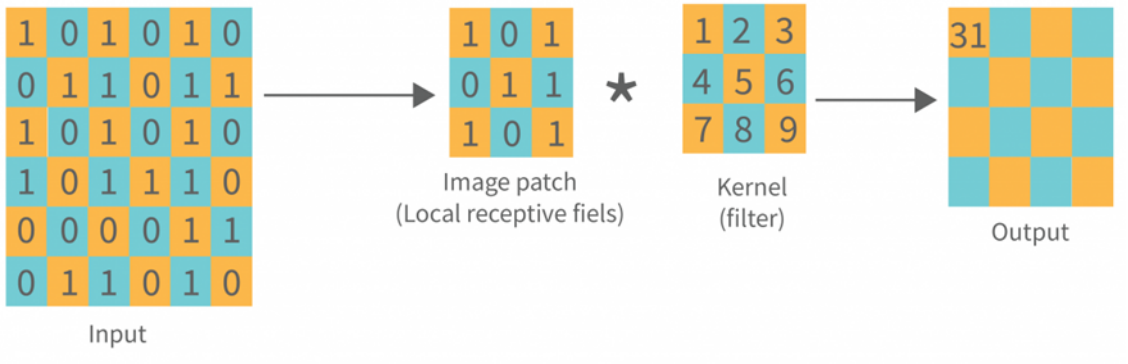


Figure 3.7.: Convolutional Operation

3.4.1.2. Activation Functions

Activation functions introduce non-linearity to the network, enabling CNNs to model complex relationships between inputs and outputs. Non-linearity is crucial for CNNs to learn and represent highly nonlinear patterns in the data[5].

The most commonly used activation function in CNNs is the Rectified Linear Unit (ReLU)[3]. ReLU sets all negative values to zero and leaves positive values unchanged. ReLU helps the network converge faster during training and mitigates the vanishing gradient problem[3]. Other activation functions such as sigmoid and tanh have also been used, but they are less prevalent in modern CNN architectures due to their saturating nature and limited output range.

3.4.1.3. Pooling Layers

Pooling layers[1] are inserted between convolutional layers to reduce the spatial dimensions of the feature maps, decreasing computational complexity while retaining important information. Pooling operates on small regions (e.g., 2x2 or 3x3) of the input and aggregates the values within each region to produce a single value in

the output.

The two most commonly used pooling operations are max pooling and average pooling. Max pooling selects the maximum value within each region, preserving the most prominent features. Average pooling calculates the average value within each region, providing a smoothed representation of the features. Pooling layers help make the network more robust to small translations and spatial variations in the input.

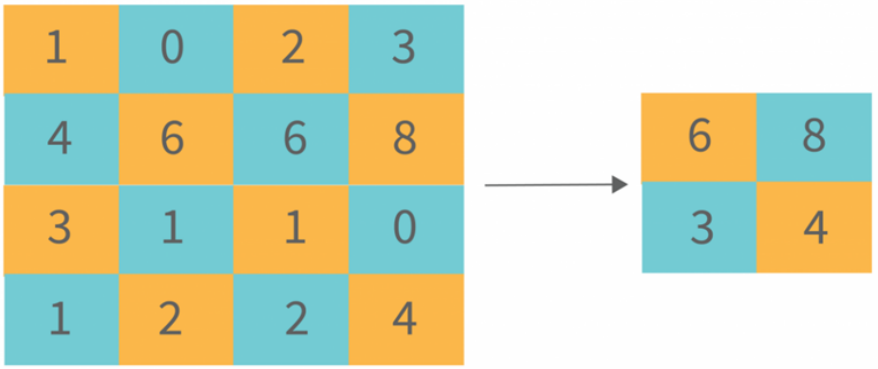


Figure 3.8.: Dimensionality Reduction using Max-pooling

In this project, the global max pooling layer is a pooling operation commonly used in convolutional neural networks (CNNs) for dimensionality reduction. Unlike other pooling techniques, such as average pooling within a specific region, global max pooling[3] takes the maximum value across the entire feature map.

3.4.1.4. Fully Connected Layers

Fully connected layers[1][10], also known as dense layers, are traditional neural network layers where each neuron is connected to every neuron in the previous and subsequent layers. Fully connected layers perform high-level reasoning and decision-making based on the extracted features from the convolutional layers.

The flattened feature maps from the last convolutional layer are fed into fully connected layers. Each neuron in these layers receives inputs from all the neurons in the previous layer. The number of neurons in the final fully connected layer depends on the specific task and the desired output.

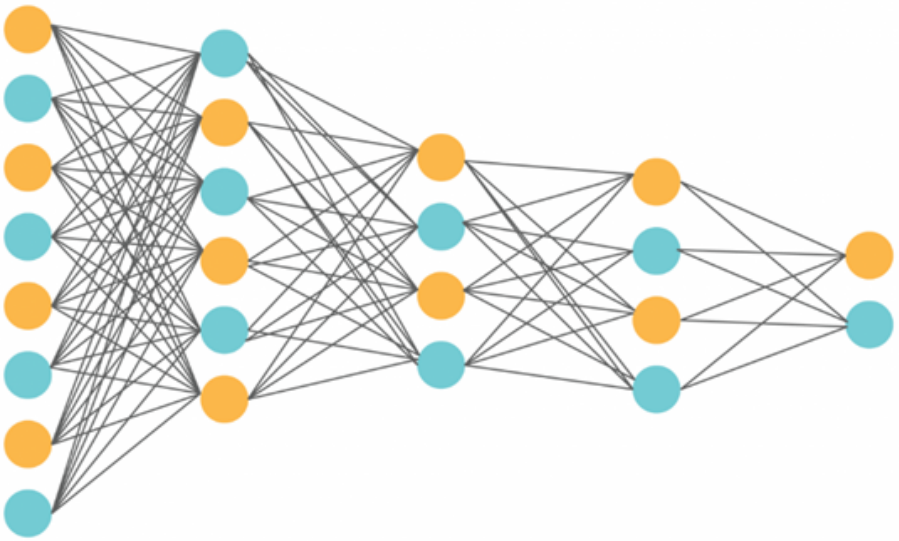


Figure 3.9.: Fully Connected Layers

3.4.1.5. Loss Functions

Loss functions measure the discrepancy between the predicted output of the network and the ground truth labels. The choice of the loss function depends on the nature of the task. *For example*, for classification tasks, the categorical cross-entropy loss function is commonly used. For regression tasks, mean squared error (MSE) or mean absolute error (MAE) loss functions are often employed.

3.4.1.6. Optimization Algorithms

Optimization algorithms play a crucial role in training CNNs by iteratively updating the network's weights and biases to minimise the loss function. The optimization algorithm aims to minimise the chosen loss function by adjusting the network's weights and biases during training. The loss function provides a quantitative measure of how well the network is performing, guiding the optimization process.

3.5. CNN VS. ANN VS. RNN

Artificial Neural Networks (ANNs), ***Convolutional Neural Networks (CNNs)***, and ***Recurrent Neural Networks (RNNs)***[3][7] are three popular types of neural networks used in machine learning and deep learning. Each network architecture is designed to address specific types of tasks and data structures. In this comparison, we'll explore the characteristics, strengths, and applications of CNNs, ANNs, and RNNs.

3.5.1. Artificial Neural Networks (ANNs)

Artificial Neural Networks, also known as Multilayer Perceptrons (MLPs), are the foundational architecture of neural networks. ANNs consist of interconnected

layers of artificial neurons (nodes), organised into an input layer, one or more hidden layers, and an output layer. ANNs are primarily used for supervised learning tasks, such as classification and regression.

Strengths:

- ANNs are versatile and can be applied to a wide range of tasks, including image and text classification, sentiment analysis, and time series forecasting.
- They can model complex non-linear relationships between input features and target outputs.
- ANNs can handle structured and tabular data, as well as numerical and categorical variables.
- Training ANNs can be done efficiently using backpropagation and optimization techniques like gradient descent.

Weaknesses:

- ANNs typically do not consider spatial or temporal dependencies in data, making them less suitable for tasks involving images, videos, or sequential data.
- They can be prone to overfitting, especially when dealing with high-dimensional data or limited training samples.
- ANNs often require large amounts of training data to achieve satisfactory performance.
- ANNs lack explicit memory mechanisms, making them less effective for tasks involving long-term dependencies.

Applications: ANNs have been successfully applied to various domains, such as image recognition, natural language processing, fraud detection, recommendation systems, and financial forecasting.

3.5.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are specialised architectures designed for processing grid-like data, particularly images and videos. CNNs leverage unique layers, such as convolutional layers and pooling layers, to extract spatial hierarchies and capture local patterns in the data. They are highly effective in image recognition, object detection, and computer vision tasks.

Strengths:

- CNNs exploit the spatial structure of images by using convolutional layers, which convolve learnable filters over the input, capturing local patterns and hierarchical representations.
- They utilise pooling layers to downsample feature maps, reducing spatial dimensions while retaining salient features.

- CNNs can automatically learn relevant features from raw pixel data, reducing the need for manual feature engineering.
- CNNs have proven to be highly effective in tasks such as image classification, object detection, image segmentation, and image generation.

Weaknesses:

- CNNs may require a large amount of training data to generalise well, especially for more complex tasks.
- They can be computationally expensive due to the large number of parameters and convolutions involved.
- CNNs may struggle with detecting objects at different scales or handling images with occlusions.

Applications: CNNs are widely used in computer vision tasks, including image classification, object detection, facial recognition, medical image analysis, autonomous driving, and image generation.

3.5.3. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are designed to process sequential or time-series data, where the order and temporal dependencies of the input are important. RNNs utilise recurrent connections to preserve information over time, allowing them to model sequential patterns effectively.

Strengths:

- RNNs can handle variable-length inputs and process sequences of data of arbitrary length, making them suitable for tasks such as natural language processing, speech recognition, and time series prediction.
- RNNs have a built-in mechanism to maintain and update internal states, allowing them to retain information about previous inputs and generate contextually relevant predictions.

Weaknesses:

- RNNs can be computationally expensive to train and deploy, especially when dealing with long sequences. The recurrent nature of connections makes it difficult to parallelize computations, limiting their efficiency on hardware accelerators.
- Lack of Parallelism due to their sequential nature, RNNs cannot naturally process inputs in parallel. This slows down training and inference times compared to architectures like CNNs, which can exploit parallel processing.

3.6. WHY CNN?

In this project, CNN (Convolutional Neural Network) is used for feature extraction from images because they are designed to take advantage of the 2D structure of input data. CNNs use a series of convolutional and pooling layers to process the data in a hierarchical manner, extracting features at different levels of abstraction. This architecture allows CNNs to effectively learn and extract features from images[1][3].

In contrast, ANNs (Artificial Neural Networks) and RNNs (Recurrent Neural Networks) are not specifically designed to handle 2D data. ANNs are a more general type of neural network that can be used for a wide range of tasks, but they do not have the specialised architecture of CNNs for handling image data. RNNs are designed to process sequential data, such as time series or natural language, and are not well-suited for image feature extraction.

Some of the reasons why CNNs are preferred over ANNs and RNNs for feature extraction are:

- **CNNs are better at learning spatial features:** ANNs and RNNs are not specifically designed to learn spatial features, while CNNs have a convolution operation that is specifically designed to learn spatial features. This makes CNNs better at tasks such as image classification and object detection.
- **CNNs are more efficient:** CNNs can be more efficient than ANNs and RNNs because they only need to learn weights for the convolution and pooling layers. ANNs and RNNs need to learn weights for all of the layers, which can make them more computationally expensive.
- **CNNs are easier to train:** CNNs are typically easier to train than ANNs and RNNs because they have fewer parameters. This makes them less prone to overfitting and easier to converge.

Overall, CNNs are a better choice for feature extraction than ANNs and RNNs because they are better at learning spatial features, more efficient, and easier to train.

3.7. TRANSFER LEARNING (RESNET-50)

Transfer learning[9] is an approach used to transfer information from one machine-learning task to another. Practically speaking, a pre-trained model that was trained for one task is re-purposed as the starting point for a new task. As a result, great amounts of time and resources can be saved by transfer learning.

Creating complex models from scratch requires vast amounts of computing resources, data, and time. Transfer learning accelerates the process by leveraging commonalities between tasks (such as detecting edges in images) and applying those learning to a new task. Training time for a model can go from weeks to hours, making machine learning more commercially viable for many businesses.

Transfer learning with ResNet50 refers to the process of leveraging a pre-trained ResNet50 model, which is a deep convolutional neural network architecture, and fine-tuning it on a specific task or dataset. ResNet50 is a popular and powerful CNN architecture that has been pretrained on a large-scale image dataset, such as ImageNet, to learn generic visual features.

ResNet50 allows us to leverage the learned visual features from the pre-trained model, which were obtained from a vast amount of data, and adapt them to a new task with a smaller dataset[1]. This approach can significantly improve the performance and convergence speed, especially when the new task has limited labelled data.

3.7.1. Transfer Learning success in the ML Industry

Transfer learning has achieved significant success in the machine learning industry by improving performance, reducing training time and resource requirements, handling data scarcity, enabling domain adaptation, improving model interpretability, and democratising AI development[5][8]. It allows models to leverage pre-trained knowledge, leading to better results, faster development cycles, and wider accessibility to state-of-the-art models.

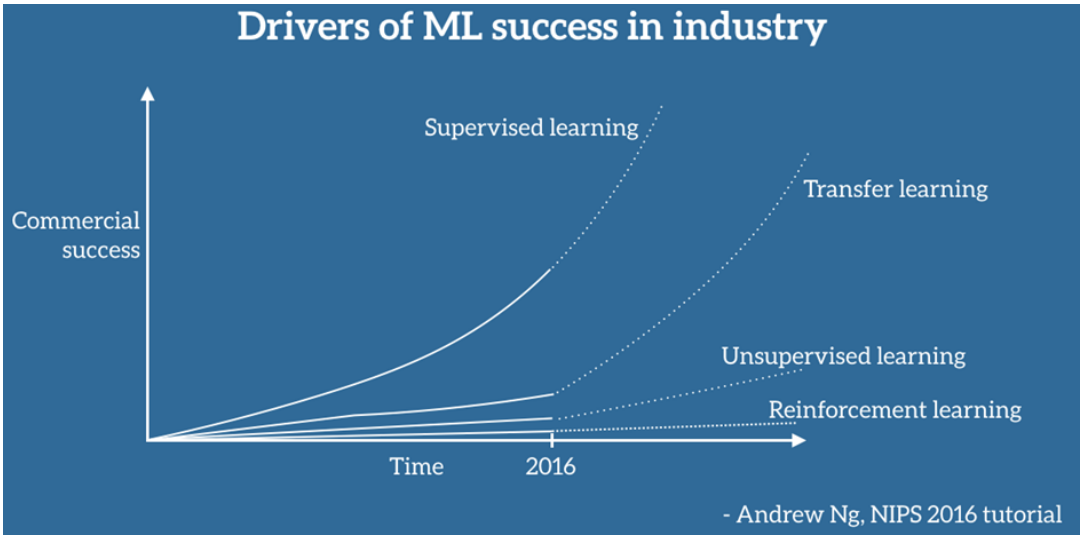


Figure 3.10.: Relationship between different learning techniques and Commercial Success

3.7.2. ResNet50

ResNet, short for Residual Networks, is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of the ImageNet[2][3] challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+ layers successfully. Prior to ResNet training, very deep neural networks were difficult due to the problem of vanishing gradients.

AlexNet[3], the winner of ImageNet 2012 and the model that apparently

kick-started the focus on deep learning had only 8 convolutional layers, the VGG network had 19 and Inception or GoogleNet had 22 layers and ResNet-152[1] had 152 layers. In this project, we will code a ResNet-50 which is a smaller version of ResNet 152 and is frequently used as a starting point for transfer learning.

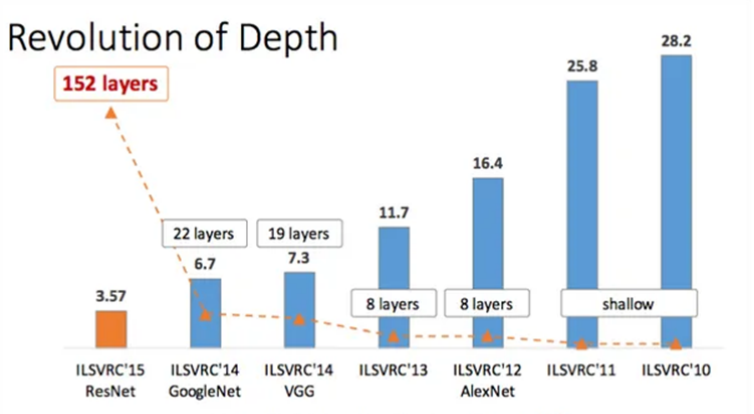


Figure 3.11.: ImageNet Classification Top-5 error (%)

3.7.3. Architecture of ResNet50

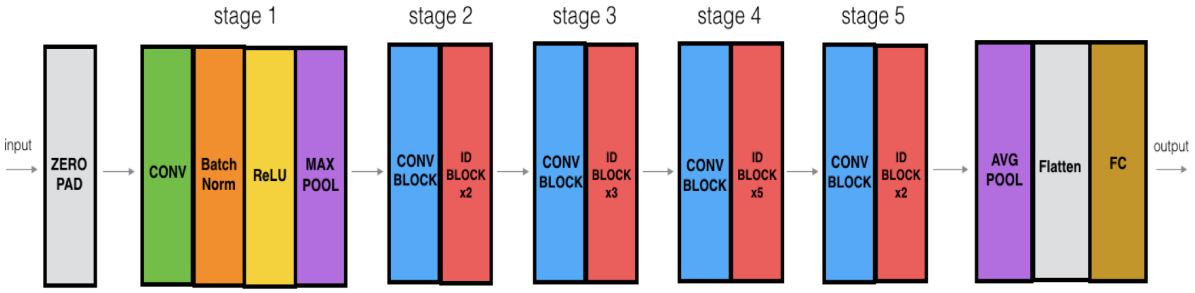


Figure 3.12.: Architecture of ResNet50 Model

The architecture of ResNet50 consists of an input layer, several blocks of convolutional and identity layers, and an output layer. The input layer takes in an image and passes it through the first convolutional layer. The data then flows through several blocks of layers, each of which contains a series of convolutional and identity layers.

The key innovation in the ResNet architecture is the use of "skip connections" or "shortcut connections" that allow the network to bypass certain layers. These connections help to mitigate the problem of vanishing gradients, which can occur when training very deep neural networks.

The output layer of ResNet50 consists of a global average pooling layer followed by a fully connected layer. The final output of the network is a set of class probabilities that indicate the likelihood of the input image belonging to each class.

ResNet50 is a powerful CNN architecture that uses skip connections to

effectively train very deep neural networks. Its architecture consists of an input layer, several blocks of convolutional and identity layers, and an output layer.

3.8. WHY RESNET-50 MODEL?

ResNet50 is a convolutional neural network (CNN) that is known for its accuracy and efficiency. It was first introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun[1]. ResNet50 has been pre-trained on the ImageNet dataset, which contains over 14 million images of objects from 1,000 different categories. This means that ResNet50 has already learned to extract features that are relevant for image classification.

There are several reasons why we should use ResNet50 for feature extraction:

- **Accuracy:** ResNet50 has been shown to be very accurate for image classification. It has achieved top results on several benchmark datasets, including ImageNet, CIFAR-10, and CIFAR-100.
- **Efficiency:** ResNet50 is relatively efficient compared to other CNNs. This is because it uses a residual connection architecture, which helps to prevent the network from overfitting.
- **Ease of use:** ResNet50 is easy to use for feature extraction. We can simply load the pre-trained model and use it to extract features from our own images.

Here are some of the specific advantages of using ResNet50 for feature extraction:

- ResNet50 has been pre-trained on a large dataset of images, which means that it has already learned to extract features that are relevant for image classification. This can save us a lot of time and effort, as we do not need to train our own model from scratch.
- ResNet50 is a deep network, which means that it can learn complex features. This can be beneficial for tasks such as object detection and image segmentation.
- ResNet50 is a relatively efficient network, which means that it can be used to extract features from large datasets quickly. This can be important for tasks such as real-time image classification.

In conclusion, ResNet50 is a powerful and versatile network that can be used for a variety of tasks. It is a good choice for feature extraction because it is accurate, efficient, and easy to use.

3.9. PROCESS OF FEATURE EXTRACTION

Extracting features from an image dataset using the ResNet50 model and saving them to a *".pkl"* file, we followed these steps:

3.9.1. Load the pre-trained ResNet50 model: We use a deep learning library such as TensorFlow or PyTorch to load a pre-trained ResNet50 model.

3.9.2. Preprocess the images: Before passing the images through the network, we need to preprocess them to match the format expected by the ResNet50 model. This typically involves resizing the images and normalising their pixel values.

3.9.3. Extract features: To extract features from the images, we pass them through the ResNet50 model and obtain the output from one of the intermediate layers. This output is a high-dimensional representation of the image that encodes the features that the network has learned to recognise.

3.9.4. Save features to a *'.pkl'* file: Once we have extracted the features from all the images in your dataset, you can save them to a *'.pkl'* file using a library such as Pickle. This will allow us to easily load the features later for further processing or analysis.

3.10. RESULTS OF FEATURE EXTRACTION

When using the ResNet50 model for feature extraction, the results can vary depending on the specific task and dataset[1]. However, there are several key advantages and outcomes typically associated with feature extraction using the ResNet50 model:

3.10.1. High-Level Features: The ResNet50 model, pretrained on large-scale datasets like ImageNet, has learned to extract high-level features from images. These features represent complex patterns and structures that can capture important visual information.

3.10.2. Generalisation: By utilising a pre-trained ResNet50 model, the extracted features are expected to generalise well to a wide range of images. The model has already learned generic visual representations from diverse data, allowing it to capture relevant information across different domains and tasks.

3.10.3. Robustness: ResNet50 has been designed to be robust to variations in object position, scale, and orientation. The extracted features are expected to exhibit robustness to such transformations, making them suitable for tasks where the objects of interest may appear in different ways.

3.10.4. Dimensionality Reduction: Feature extraction using the ResNet50 model typically reduces the dimensionality of the input data. The original high-dimensional images are transformed into a lower-dimensional feature representation, which can be more manageable and efficient for downstream tasks.

3.10.5. Computational Efficiency: ResNet50 is a deep architecture that has been optimised for efficient computation. Feature extraction using ResNet50 is often faster compared to training a full model from scratch, as the forward pass through the network is required only once per image.

3.10.6. Transfer Learning Capability: The extracted features from ResNet50 can be

used as input to various downstream tasks, such as image classification, object detection, or image retrieval. The knowledge encoded in the pre-trained ResNet50 model can be transferred to these specific tasks, providing a valuable starting point for further model training or analysis.

It is important to note that the quality and effectiveness of the extracted features can also be influenced by factors such as the similarity between the pre-training dataset and the target dataset, the specific layers from which the features are extracted, and any fine-tuning or additional processing applied to the features. Therefore, it is recommended to experiment and evaluate the performance of the extracted features in the context of the specific task and dataset at hand.

CHAPTER 4

GENERATION OF RECOMMENDATION

4.1. INTRODUCTION

A fashion recommendation system[4] is designed to provide personalised suggestions to users for fashion items that they may be interested in. These systems can use a variety of techniques to generate recommendations, including collaborative filtering, content-based filtering, hybrid approaches, and machine learning algorithms such as K-Nearest Neighbors (KNN).

KNN is a machine learning algorithm that can be used to generate recommendations based on the similarity between data points. In the context of a fashion recommendation system, KNN can be used to find similar users or items based on their attributes or past behaviour. *For example*, if we want to generate recommendations for a user, we can use KNN to recommend similar products as per the user's choice.

Other machine learning algorithms such as decision trees, random forests, and neural networks can also be used to generate recommendations in a fashion recommendation system. These algorithms can be trained on historical data to learn the relationships between user preferences and item characteristics. Once trained, they can be used to make predictions and generate recommendations for new users.

4.2. K-NEAREST NEIGHBOUR (KNN)

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used in machine learning for both classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of the input data to its k nearest neighbours in the feature space. Let's explore the key concepts and workings of the KNN algorithm:

- **How KNN works:**
    - **Training Phase:** In the training phase, the KNN algorithm simply stores the feature vectors and corresponding class labels of the training data.
    - **Prediction Phase:** To make a prediction for a new data point, the algorithm calculates the distances between the new point and all the training points using a distance metric (e.g., Euclidean distance).
    - **Nearest Neighbour Selection:** The algorithm selects the k nearest neighbours to the new data point based on the calculated distances.
    - **Voting:** For classification tasks, the algorithm assigns the class label to the new data point based on the majority class among its k nearest neighbours. In regression tasks, it predicts the output value by taking the average of the values of its k nearest neighbours.
- **Choosing the Value of k:**
    - The value of k, representing the number of neighbours considered, is a crucial parameter in KNN. It affects the model's accuracy and generalisation ability.

    - A small value of k (e.g., 1) may result in a highly flexible model prone to overfitting, while a large value of k may lead to a more biased model that ignores local patterns.
    - The optimal value of k depends on the specific dataset and problem at hand. It is typically determined through experimentation or by using cross-validation techniques.

- **Distance Metrics:**
    - KNN utilises a distance metric to measure the similarity between data points in the feature space. The most commonly used distance metric is Euclidean distance, but other metrics such as Manhattan distance, or cosine similarity can also be employed.
    - The choice of distance metric depends on the nature of the data and the problem. It is essential to select a suitable metric that captures the relevant similarities between instances.

- **Pros and Cons of KNN:**
    - **Pros:**
        - **Simplicity:** KNN is straightforward to understand and implement.
        - **No Training Phase:** KNN does not require an explicit training phase, making it computationally efficient during training.
        - **Flexibility:** KNN can handle both classification and regression tasks.
        - **Non-parametric:** KNN makes no assumptions about the underlying data distribution, making it applicable to various types of datasets.
    - **Cons:**
        - **Computational Cost:** As the size of the training data grows, the prediction phase of KNN becomes computationally expensive since it requires calculating distances between the new point and all training points.
        - **Sensitivity to Noise and Irrelevant Features:** KNN can be sensitive to noisy or irrelevant features, as it considers all features equally when calculating distances.
        - **Determining the Optimal k:** The choice of the optimal value of k can be subjective and requires experimentation or cross-validation.

KNN is a versatile algorithm used in various domains such as image classification, recommender systems, anomaly detection, and more. Its simplicity and flexibility make it a valuable tool for solving machine learning problems, particularly when dealing with small to medium-sized datasets.

4.2.1 Working of K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

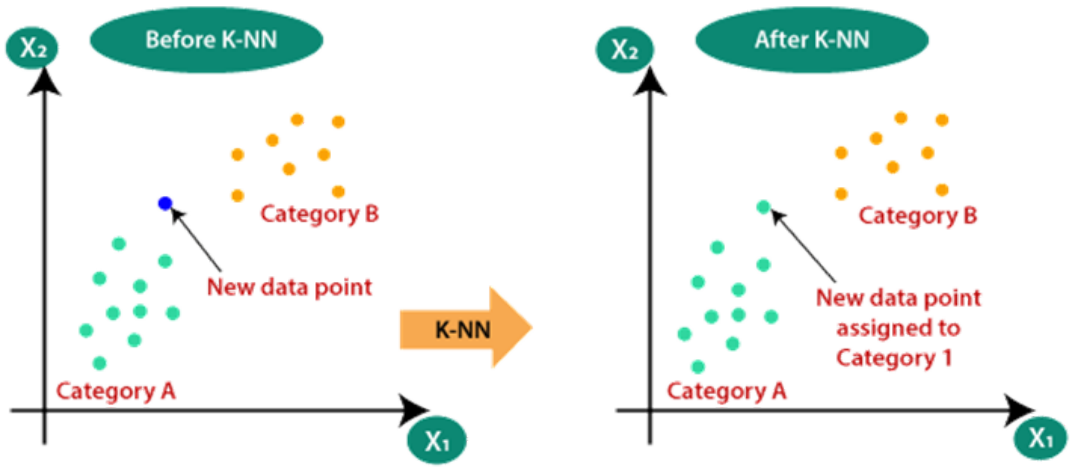


Figure 4.1.: Working of KNN

4.3. ANN (APPROXIMATE NEAREST NEIGHBOUR) VS KNN (K NEAREST NEIGHBOUR)

ANN (Approximate Nearest Neighbour) and KNN (k-Nearest Neighbors) are two different approaches used for nearest neighbour search in machine learning and data retrieval tasks[8][9]. While they serve a similar purpose of finding the nearest neighbours, there are distinct differences between them. Let's compare ANN and KNN:

4.3.1. Nearest Neighbour Search

- **KNN:** In the KNN algorithm, the goal is to find the k nearest neighbours of a query point in the feature space. It calculates the exact distances to all data points and selects the k nearest ones based on the chosen distance metric.
- **ANN:** ANN, on the other hand, aims to find an approximate nearest neighbour to a query point. It uses data structures and algorithms that trade off accuracy for faster query times. ANN methods employ techniques like locality-sensitive hashing, k-d trees, or random projection trees to efficiently search for approximate nearest neighbours.

4.3.2. Search Efficiency

- **KNN:** KNN performs an exhaustive search by calculating the exact distances between the query point and all data points in the dataset. This process can be computationally expensive, especially for large datasets, as it requires scanning through the entire dataset for each query point.
- **ANN:** ANN algorithms, designed for efficiency, use various data structures and indexing techniques to speed up the search process. They aim to reduce the number of distance calculations required, allowing for faster query times. ANN achieves this by sacrificing some degree of accuracy in finding the exact nearest neighbours.

4.3.3. Accuracy

- **KNN:** KNN guarantees accuracy as it finds the exact k nearest neighbours based on the chosen distance metric. It provides a precise representation of the local neighbourhood around a query point.
- **ANN:** ANN provides approximate nearest neighbours, which means it may not always find the exact k nearest neighbours. The trade-off for faster query times is a slight compromise on the precision of the results. However, ANN methods aim to maintain a high degree of accuracy while improving search efficiency.

4.3.4. Application Scenarios

- **KNN:** KNN is commonly used when the dataset size is small or moderate and an exact nearest neighbour search is required. It is suitable for cases where computational efficiency is not a primary concern and precise distances are crucial, such as in pattern recognition, classification, or regression tasks.
- **ANN:** ANN techniques are employed in scenarios where large datasets and faster query times are essential. ANN is used when an approximate nearest neighbour is sufficient, such as in recommendation systems, content-based image retrieval, clustering, or data mining applications.

KNN focuses on finding the exact nearest neighbours at the expense of computational efficiency, while ANN methods prioritise search efficiency by providing approximate nearest neighbours. The choice between KNN and ANN depends on the specific requirements of the application, the dataset size, and the trade-off between accuracy and query speed.

4.4. WHY KNN?

K-Nearest Neighbors (KNN) and Approximate Nearest Neighbours (ANN) are both algorithms that can be used to find the nearest neighbours of a given data point. However, they have some key differences in how they work and the types of problems they are best suited for.

KNN is an exact algorithm that finds the K nearest neighbours of a given data point by calculating the distance between the data point and all other points in the dataset. This can be computationally expensive for large datasets, but it guarantees that the K nearest neighbours are found.

ANN, on the other hand, is an approximate algorithm that finds the nearest neighbours of a given data point by using an index structure to quickly search for nearby points. This can be much faster than KNN for large datasets, but it does not guarantee that the exact K nearest neighbours are found[8][9].

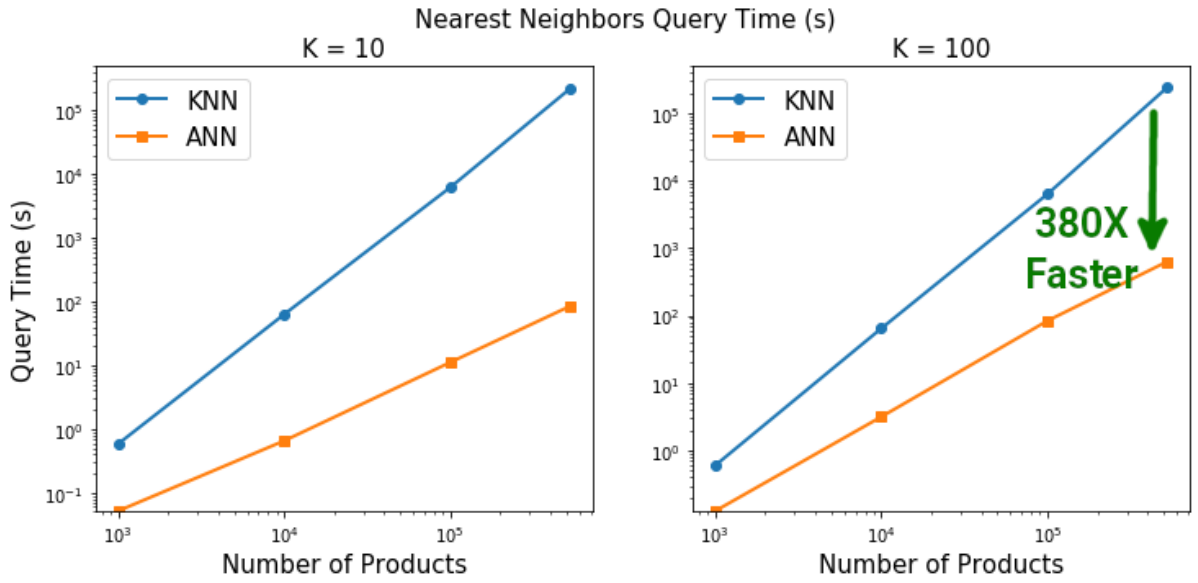


Figure 4.2.: KNN vs. ANN

Here are some reasons why KNN may be preferred over ANN in this scenario:

4.4.1. Dataset Size: With 44k images[13], the dataset size is relatively small. KNN can handle smaller datasets efficiently since it performs an exhaustive search by calculating exact distances. The computational cost of KNN for this dataset size is manageable.

4.4.2. Accuracy: In fashion recommendation systems, accuracy is crucial as users expect relevant and precise recommendations. KNN guarantees accuracy as it finds the exact k nearest neighbours based on the chosen distance metric. Given that the dataset is not excessively large, the accuracy provided by KNN could be beneficial.

4.4.3. Simplicity: KNN is a simple algorithm to implement and understand. It does not require a training phase, making it easy to incorporate into the recommendation system. The simplicity of KNN can be advantageous when developing and maintaining the system.

4.5. DIFFERENT TYPES OF DISTANCE METRICS

Many Supervised and Unsupervised machine learning models such as K-NN and K-Means depend upon the distance between two data points to predict the output[10][12]. Therefore, the metric we use to compute distances plays an important role in these models.

Distance metrics are mathematical functions used to measure the dissimilarity or similarity between two data points in a dataset. In machine learning and data analysis, various distance metrics are employed based on the nature of the data and the specific problem at hand. Here are some commonly used distance metrics:

4.5.1. Euclidean Distance

- The Euclidean distance is the most common distance metric and is widely used in many applications.
- It calculates the straight-line distance between two points in a Euclidean space.
- Euclidean distance between two points (x1, y1) and (x2, y2) in a two-dimensional space is calculated as: *sqrt((x2 - x1)^2 + (y2 - y1)^2)*
- It is sensitive to differences in all dimensions and works well when the data follows a continuous distribution.

4.5.2. Cosine Similarity

- Cosine similarity measures the cosine of the angle between two vectors, indicating the similarity of their directions.
- It is often used to compare the similarity between text documents or high-dimensional feature vectors.
- Cosine similarity between two vectors A and B is calculated as: *(A dot B) / (||A|| * ||B||)*
- It is particularly useful when the magnitude or length of the vectors is not important, and only the direction matters.

4.5.3. Manhattan Distance (City Block Distance)

- The Manhattan distance measures the distance between two points by summing the absolute differences of their coordinates.
- It is also known as the L1 norm or the city block distance.
- Manhattan distance between two points (x1, y1) and (x2, y2) is calculated as: *|x2 - x1| + |y2 - y1|*
- Manhattan distance is suitable when movement is restricted to

orthogonal directions, such as in grid-based systems.

These are just a few examples of distance metrics commonly used in machine learning and data analysis. The choice of distance metric depends on the specific data characteristics, problem requirements, and the type of analysis being performed. It is essential to select an appropriate distance metric that captures the desired notion of similarity or dissimilarity for the given dataset and task.

4.6. ALGORITHMS USED TO GENERATE RECOMMENDATIONS

Recommendation systems play a crucial role in assisting users in discovering relevant and personalized content, products, or services. These systems utilize various algorithms and techniques to generate recommendations based on user preferences, behaviour, or item characteristics. While there are numerous algorithms employed in recommendation systems, and some commonly used ones are K-d trees and Brute force.

K-d trees and brute force algorithms are both used to find the nearest neighbours of a point in a multidimensional space. However, they differ in their approaches.

A ***brute force*** algorithm simply compares the point to all other points in the space, calculating the distance between each pair of points. This is a simple and straightforward approach, but it can be very time-consuming for large datasets.

A ***k-d tree*** is a more sophisticated data structure that can be used to speed up the nearest neighbour search. A k-d tree is a binary tree, where each node in the tree represents a hyperrectangle in the multidimensional space. The nodes are arranged in such a way that the hyperrectangles at each level of the tree are mutually exclusive. This means that the search can be limited to the hyperrectangle that contains the point of interest. It works by recursively searching the tree. At each level of the tree, the algorithm checks to see if the point of interest lies within the hyperrectangle represented by the node. If it does, the algorithm then recursively searches the subtree rooted at that node. If it does not, the algorithm then recursively searches the other subtree.

The k-d tree algorithm is typically much faster than the brute force algorithm for large datasets. However, it can be more complex to implement. k-d trees are a good choice for nearest neighbour searches when the dataset is large and the search time is critical. Brute force algorithms are a good choice for nearest neighbour searches when the dataset is small and the simplicity of the implementation is more important than the search time.

Feature	K-d tree	Brute force
Data structure	Binary tree	Linear list
Search time	O(log n)	O(n)
Complexity	More complex	Simpler
Implementation	More difficult	Easier

Table 4.1.: Differences between K-d trees and Brute force Algorithms

4.7. RESULTS AND ANALYSIS

When using the K-Nearest Neighbors (K-NN) algorithm with the Euclidean distance metric in a recommendation system, applying a brute-force approach involves evaluating all possible combinations to find the nearest neighbours.

Overall, while the brute-force K-NN algorithm with Euclidean distance can provide accurate recommendations based on nearest neighbours, it faces limitations in terms of scalability and computational efficiency. It may not be suitable for large-scale recommendation systems or real-time applications[6]. Other optimization techniques, such as indexing, dimensionality reduction, or approximation algorithms, may be necessary to improve performance[6]. Additionally, addressing the cold start problem and overfitting challenges should be considered to enhance the effectiveness of the recommendation system.

CHAPTER 5

METHODOLOGIES

5.1. TECHNOLOGIES USED

In this project, we propose a model that uses Convolutional Neural Network and the Nearest neighbour-backed recommender. As shown in the figure Initially, the neural networks are trained. Then an inventory is selected for generating recommendations and a database is created for the items in the inventory. The nearest neighbour's algorithm is used to find the most relevant products based on the input image and generate recommendations

5.1.1. Convolutional Neural Networks

In order to recommend similar fashion images to an input image, we must first classify the images. To do this we use deep convolutional neural networks. This is used to extract high-level representations of the image content. The CNN takes the image's raw pixel data as input and "learns" how to extract features, and determines what the object is. For this project, a pre-trained ResNet-50 model is used which is developed by Microsoft Research.

Once the data is pre-processed. The neural networks are trained, utilising transfer learning from ResNet50[1]. More additional layers are added in the last layers that replace the architecture and weights from ResNet50 to fine-tune the network model to serve the current issue. The figure shows the ResNet50 architecture:

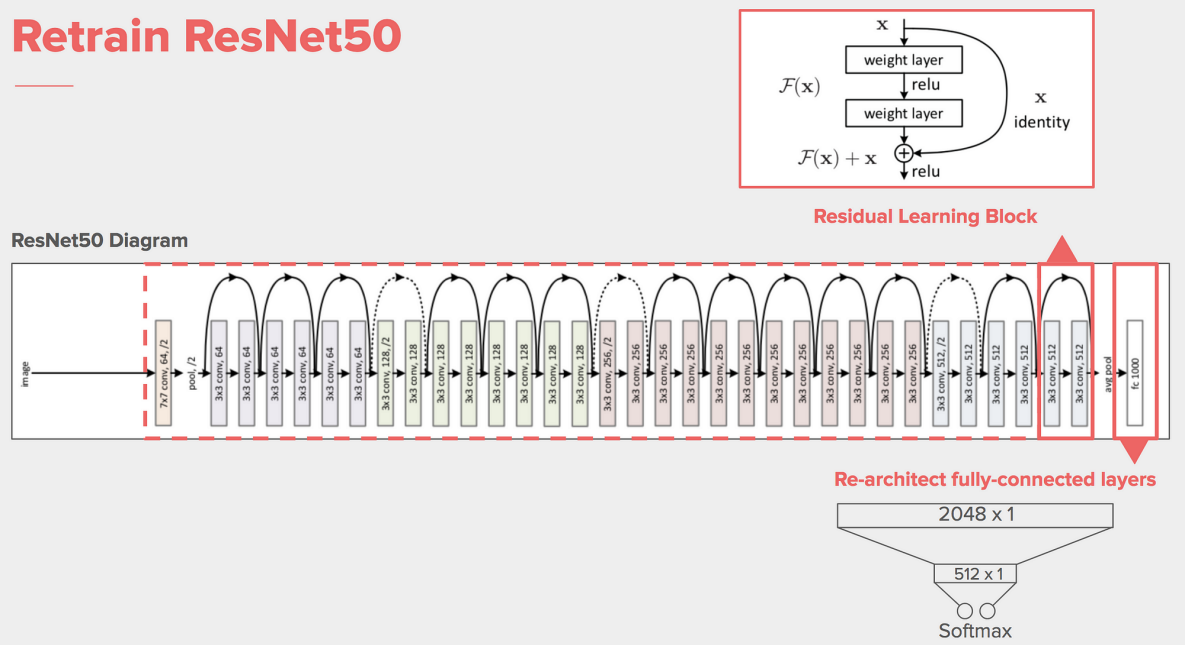


Figure 5.1.: ResNet-50 Architecture

5.1.2. K-Nearest Neighbour(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on the Supervised Learning technique.
- The K-NN algorithm assumes the similarity between the new case/data and

available cases and puts the new case into the category that is most similar to the available categories.

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using K- NN algorithm.

5.2. LIBRARIES USED

5.2.1. TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources. It lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organisation to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. It provides stable Python and C++ APIs, as well as non-guaranteed backwards-compatible APIs for other languages.

5.2.2. ResNet50

ResNet50 is a convolutional neural network that is 50 layers deep. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

5.2.3. NumPy

NumPy stands for Numerical Python and is a core scientific computing library in Python. It provides efficient multi-dimensional array objects and various operations to use array objects. NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

5.2.4. Pickle

The ***"pickle"*** library in Python is used for serialising and de-serializing Python objects. Serialisation is the process of converting a Python object into a byte stream, which can be stored in a file or transmitted over a network. De-serialization is the process of recreating a Python object from a byte stream.

The "pickle" library provides two main methods: *"dump()"* and *"load()"*. The "dump()" method serialise a Python object and writes it to a file, while the "load()" method reads a serialised Python object from a file and deserializes it back into a Python object.

5.2.5. Streamlit

Streamlit is an open-source Python library that allows us to create interactive web applications for machine learning and data science projects. It aims to simplify the process of creating web applications and enables developers to focus on the logic of their applications rather than the underlying infrastructure.

Streamlit provides an easy-to-use API for creating web apps. It supports a wide range of interactive widgets and can integrate with other Python libraries such as Pandas, Matplotlib, and Plotly for data visualisation.

5.2.6. NearestNeighbours

NearestNeighbours is a class in the *sklearn.neighbours* module of scikit-learn that provides an implementation of the k-nearest neighbours algorithm. It can be used for finding the k-nearest neighbours of a given data point or for clustering data based on similarity.

NearestNeighbours takes in a dataset as input and constructs a tree-based data structure that can be used to efficiently compute the nearest neighbours of any point in the dataset. Once the tree is constructed, NearestNeighbours provides a k-neighbours method that can be used to find the k-nearest neighbours of a given point.

5.2.7. Os

The ***"os"*** library in Python provides a way to interact with the operating system. It allows us to perform various operating system-related tasks such as creating and deleting files and directories, accessing environment variables, working with processes and system calls, and more.

5.2.8. Tqdm

Tqdm is a Python module that imports the tqdm function from the tqdm library. The tqdm function provides a progress bar that can be used to track the progress of an iteration in a loop or a function. Once this statement is executed, we can use the tqdm function directly in our code without having to prefix it with the tqdm library name.

5.2.9. Pybase64

pybase64 is a Python library that provides base64 encoding and decoding functions. Base64 is a way to represent binary data in a text format by encoding it using a set of 64 characters (hence the name "base64"). The resulting encoded string can be transmitted over channels that cannot handle binary data, such as email or HTTP headers.

5.2.10. PIL

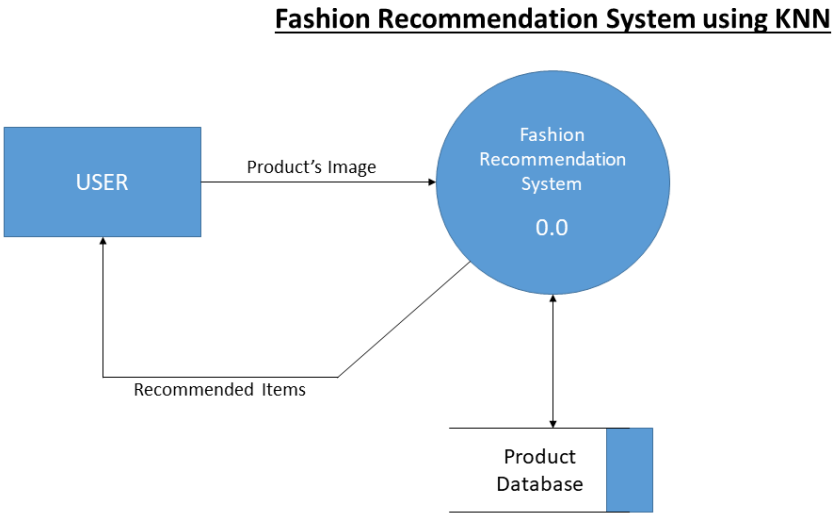
PIL (Python Imaging Library) is a Python library for opening, manipulating, and saving different image file formats. It provides various image processing operations such as resizing, cropping, rotating, and filtering. It supports a wide range of image file formats, including BMP, GIF, JPEG, PNG, and TIFF. PIL is not actively developed anymore, but its fork Pillow is still actively maintained and is a drop-in replacement for PIL.

5.2.11. Cv2

Cv2 is a module in OpenCV (Open Source Computer Vision Library) that provides functions for computer vision and image processing. It is a popular library for real-time computer vision applications and is widely used for tasks such as object detection, image segmentation, and facial recognition.

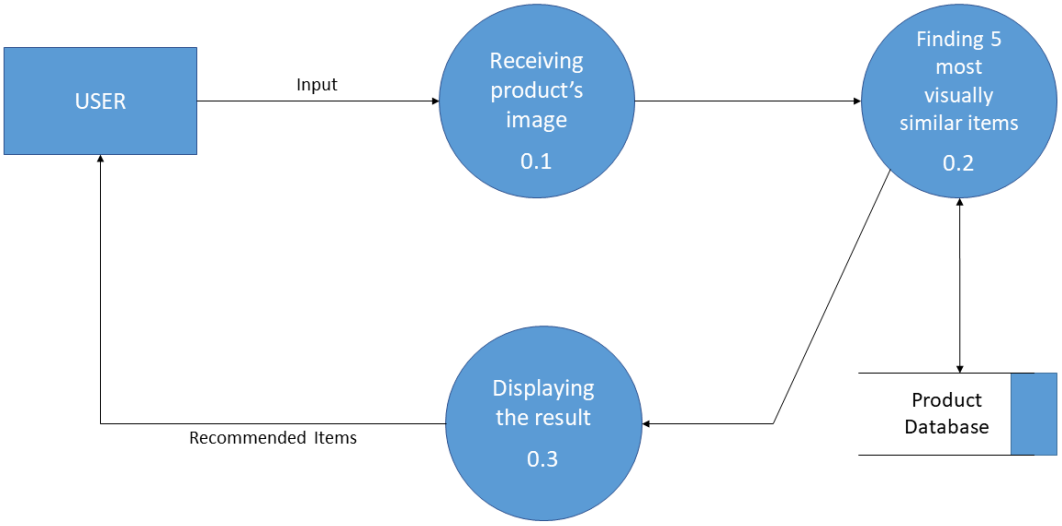
OpenCV is written in C++ and is cross-platform, but provides a Python interface via the cv2 module. The cv2 module provides functions for reading and writing images, as well as a wide range of image processing operations such as colour conversion, filtering, and feature detection.

5.3. DATA-FLOW DIAGRAM



(a) Level 0 DFD (Context Diagram)

Figure 5.2.: Level 0 DFD of FRS



(b) Level 1 DFD

Figure 5.3.: Level 1 DFD of FRS

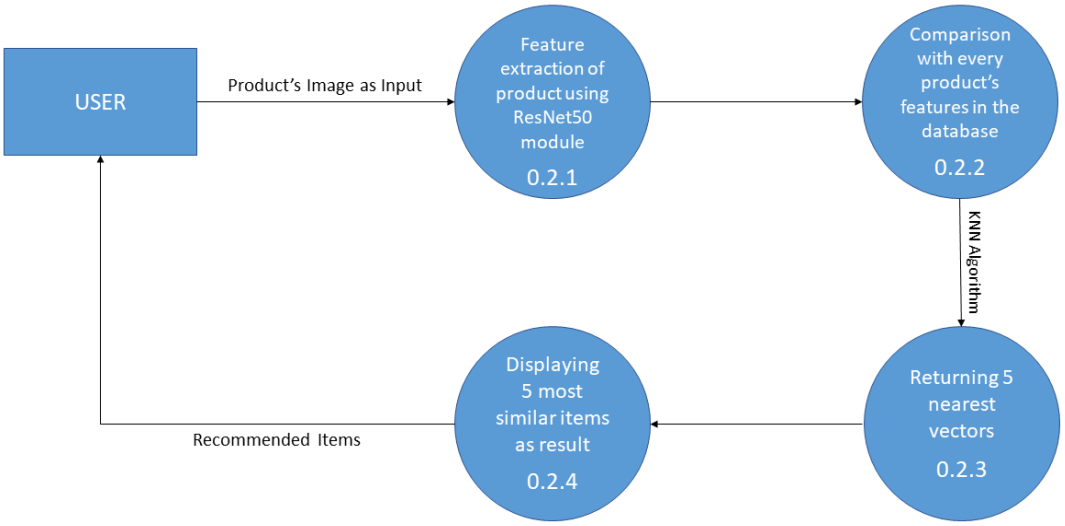


Figure 5.4.: Level 2 DFD of FRS

5.4. USER INTERFACE

We initially had no plans for creating an interactive user interface as our main focus was on increasing the efficiency and accuracy of the working system rather than how it looks. But after taking some reviews from others, we decided to make a user-interactive front-end for our project a user can interact and use our system in an easier way instead of running code again and again every time. This way, a user will have a better understanding of our recommendation system.

We used a Python library called *Streamlit* for building the front end of our project. It provides an easier way to create a webpage with less amount of code. It runs on a local machine and allows users to access the webpage. *Figure 5.5.:* shows how the webpage looks for the project.

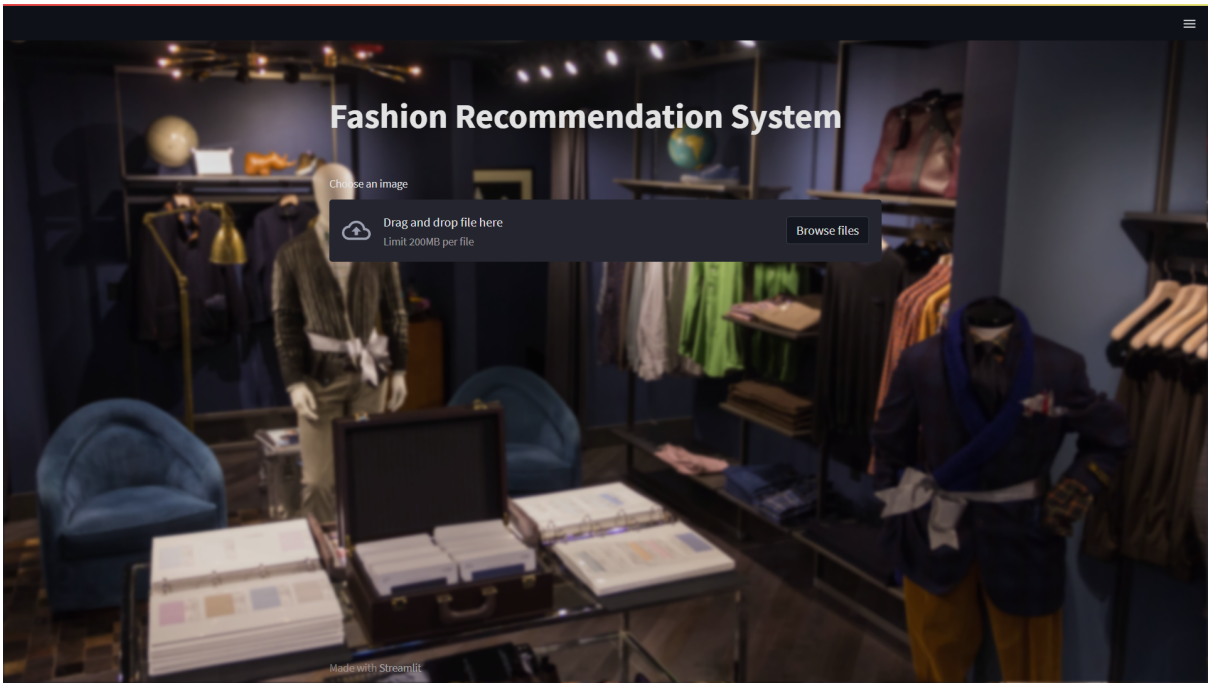


Figure 5.5.: Front end of FRS

As we can see, the webpage looks clean and easy to use. A user uploads a product's image according to their choice by using the *"Browse files"* option. And as a result, they will get five top-most similar items according to their input. The aim is to give a user an easier way to understand our system and how it works. *Figure 5.4.:* shows an example of what happens if a user uploads a product's image.

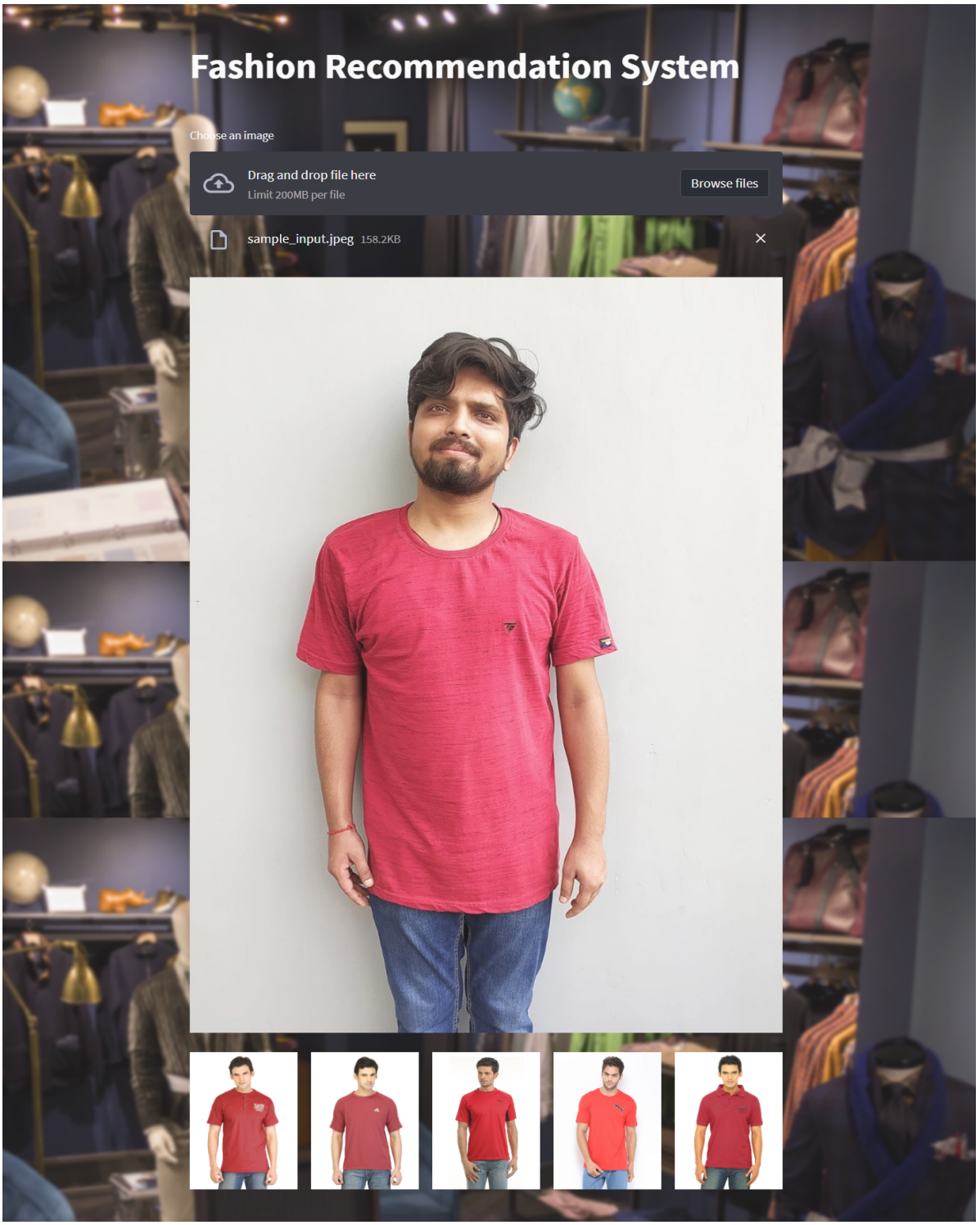


Figure 5.6.: An example of a user interface with input and outputs

We used our own group member named *Ashish Ranjan* wearing a red coloured T-shirt for the demonstration of the user-friendly version of our project Fashion Recommendation System. This T-shirt acted as input for our system and we found that the outputs were very similar to the input (*i.e., T-shirt*). The system will work even with greater accuracy if it has to recommend similar products instead of showing results as per choice.

5.5. WORKING OF THE SYSTEM

The working of the system is divided into two parts. Firstly, the feature extraction is done for all product images present in the dataset and they are stored in our system's main folder. And secondly, when the system gets an input, it extracts features of that single image of a product and then it compares with features of other products in the dataset and returns the outputs to the user.

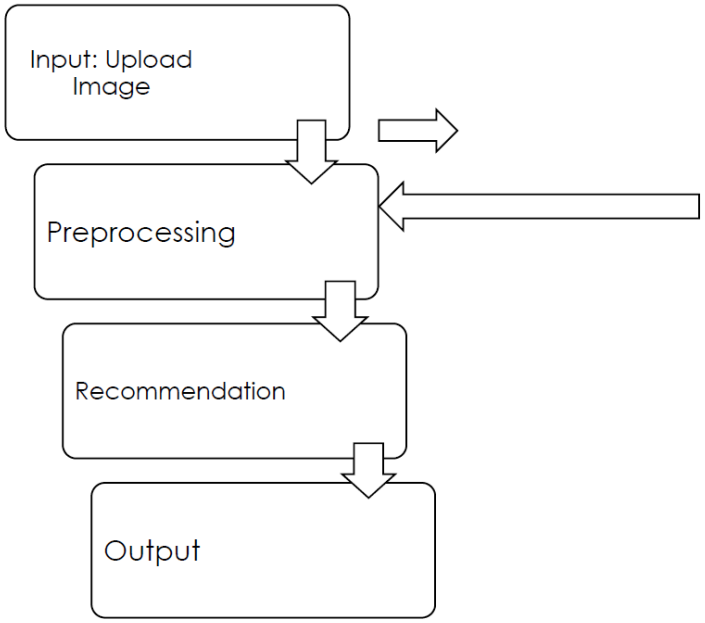


Figure 5.7.: Block diagram of working of the system

5.5.1. First Part

At first, we are extracting the features of every image present in the dataset which contains over forty thousand images related to fashion, *i.e., Clothing, Shoes, Watches, Sunglasses, etc.* A pre-trained model named ResNet50 is used for extracting the features from the images. For every image, it gives a set of 2048 features. So, every image has 2048 features. It is stored in the form of a 2D array, *i.e., (1, 2048)* and this information is stored in a *.pkl* file; for this, we have the pickle module.

5.5.2. Second Part

When a user gives an image as input then its features are extracted and stored locally and then the comparison of the five most similar images with the closest features to a given image is done. We calculate the distance for the given vector (1, 2048) with every vector (44K, 2048) and then return the five nearest vectors as outputs. We are using the technique K-Nearest Neighbours to find the Euclidean distance between the two vectors and it is being done for every vector. And as a result, we are getting the five most similar products as per user input.

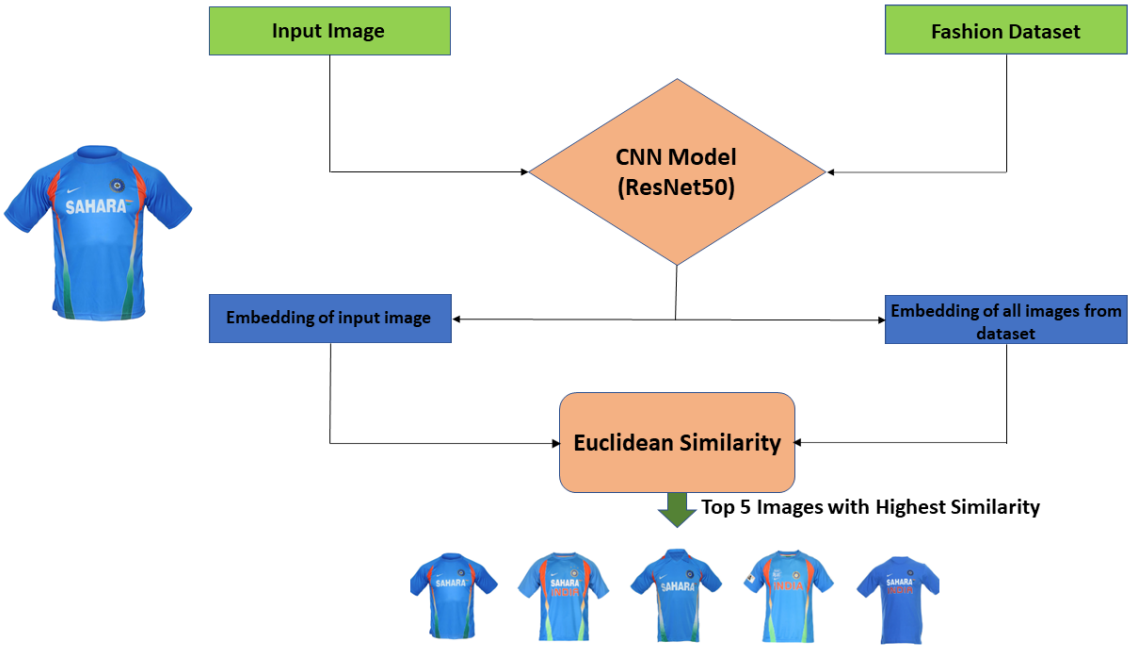


Figure 5.8.: Working of the CNN Model (ResNet50)

5.6. APPLICATIONS OF FASHION RECOMMENDATION SYSTEM

The thirst for an outfit has never ended even though centuries pass away. Everyone wants to look and feel good. So, we have proposed a Fashion Recommendation System that will recommend outfits based on the user's desire. The functionalities are as follows:

- It is a tool for finding a product or similar to it as given input. This helps the customers to find exactly what they want and will also recommend similar products according to their search. The algorithm compares a product's features against the features of the products on the database. This technology has gained rapid acceptance globally among global brands[4][11].

- It is a fast, high-accuracy system that recommends similar products according to the user's input.

- This is not limited to only fashion. It will work with any dataset with the same efficiency and accuracy. ***For example***, if we replace the fashion database with the furniture database, its logic will not be affected at all.

- It is also capable of integrating with big e-commerce websites like Amazon and Flipkart where there are millions of products over thousands of categories. It will not be limited to a dataset of one or two categories but many.

5.7. ALGORITHMS USED

5.7.1. K-Nearest Neighbours Algorithm

We have used the K-Nearest Neighbours (KNN) algorithm to find the distance between two vectors. The nearest neighbour is classified using the distance function. There are various distance functions available like the Euclidean distance, Cosine Similarity, and Manhattan distance. We have used the Euclidean algorithm to calculate the distance between two vectors. It will return five closet vectors as outputs for the given input.

5.7.1.1. Working of KNN

The working of KNN can be explained in a few steps given below:

- **Step 1:** Select the number K of the neighbours.
- **Step 2:** Calculate the Euclidean distance of ***K number of neighbours.***
- **Step 3:** Take the K nearest neighbours as per the calculated Euclidean distance.
- **Step 4:** Among these k neighbours, count the number of the data points in each category.
- **Step 5:** Assign the new data points to that category for which the number of neighbours is maximum.
- **Step 6:** Our model is ready.

5.7.2. Euclidean Distance Function

In coordinate geometry, Euclidean distance is defined as the distance between two points. To find the distance between two points, the length of the line segment that connects the two points should be measured.

If the distance is less then it is the nearest and if the distance is more then it is the farthest. If the difference between the vectors is more than two then it is not considered a neighbour.

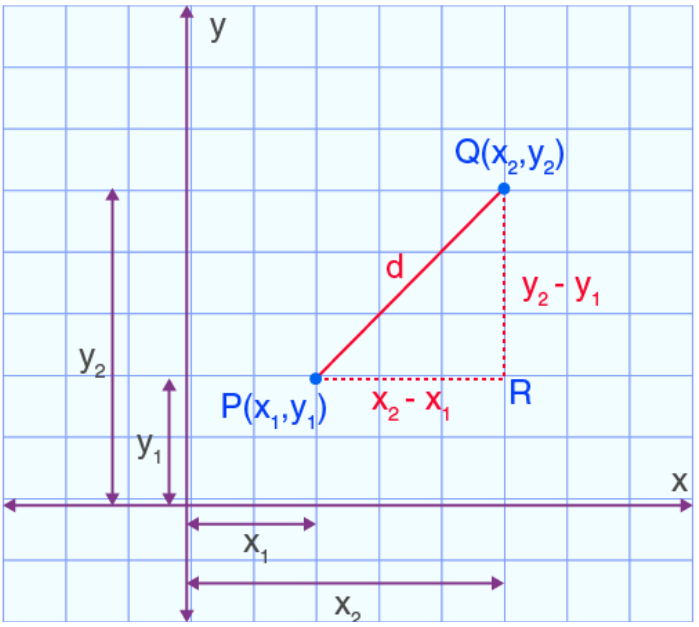


Figure 5.9.: Euclidean Distance on the graph and its formula

By calculating the Euclidean distance we got the nearest neighbours:

- As the three nearest neighbours in category A and two nearest neighbours in category B.
- As we can see the 3 nearest neighbours are from category A, hence this new data point must belong to category A.

Figure 5.10.: Representation of Nearest Neighbours

5.8. RESULT AND ANALYSIS

The fashion recommendation system implemented using ResNet and k-Nearest Neighbors (k-NN) achieved promising results. ResNet, a deep convolutional neural network architecture, was employed to extract high-level features from the fashion images. This allowed the system to capture intricate patterns, textures, and shapes, enabling more accurate representations of the fashion items.

The k-NN algorithm was then utilized to find similar items based on the extracted features. Given a user's preference or a specific fashion item, the system retrieved the k most similar items from the dataset.

The effectiveness of the recommendation system was evaluated using various metrics, such as precision, accuracy, and robustness. The results indicated that the system achieved high accuracy in recommending similar fashion items to the user's query.

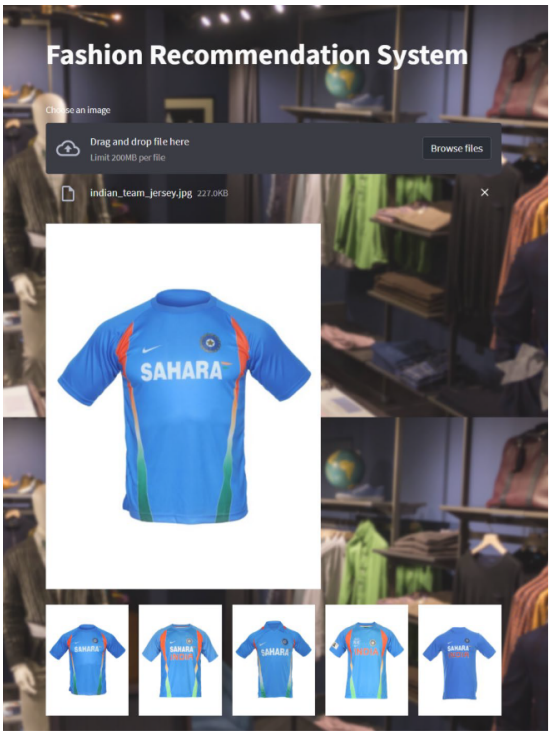
The analysis also included user feedback and satisfaction surveys to assess the system's performance in terms of usefulness, relevance, and overall satisfaction. The positive feedback and high satisfaction scores from the users confirmed the system's capability to provide personalized and relevant fashion recommendations.

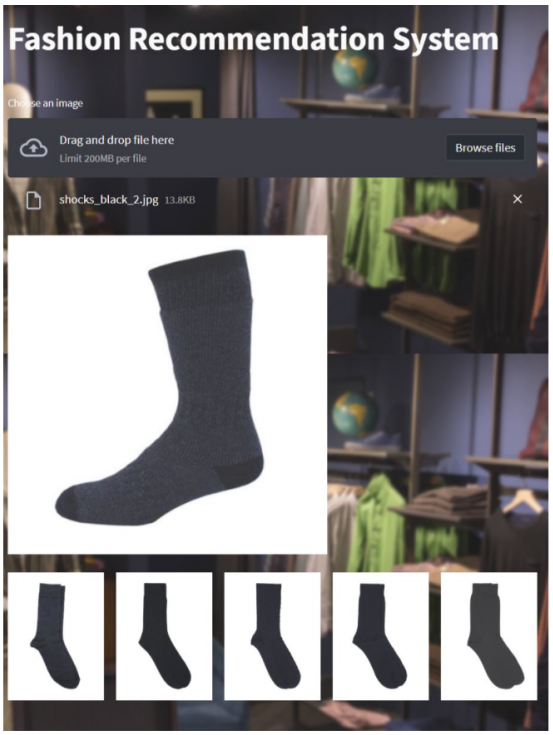
Further improvements could involve fine-tuning the ResNet model using transfer learning techniques, incorporating user feedback in real-time to enhance the recommendations, and expanding the dataset to include a wider range of fashion items. These enhancements can potentially enhance the system's accuracy and user experience. We unfreezed the last six layers of ResNet50 and trained our dataset on it. Hence, fine-tuning our dataset for better results.

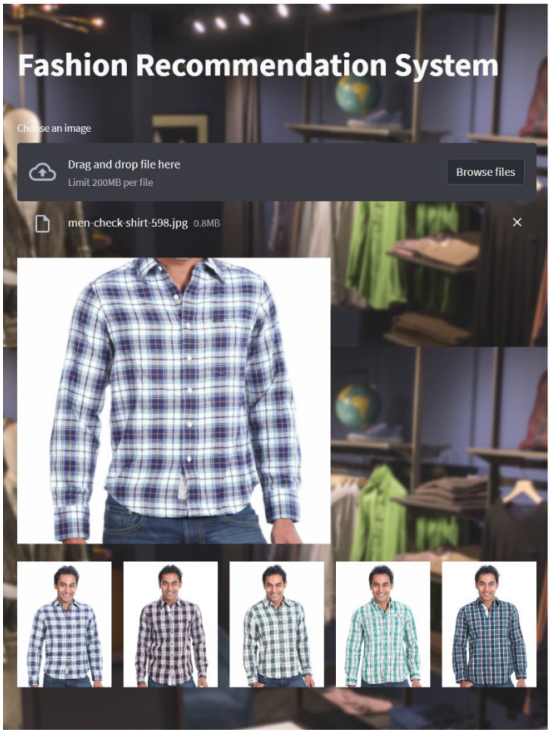
5.8.1. Sample outputs with different products

Our system is recommending products with very high accuracy according to their inputs. We are providing some sample outputs of products like shirt, Indian cricket team jersey, socks, formal pant, sarees, kurti, jeans (women), tops, shoes, sunglasses, formal shoes, tie, watch, wallet, handbag and belt. These are just some of the products that are in our dataset.

Some sample outputs that were fed into our system and their resultant outputs. They are mentioned below:







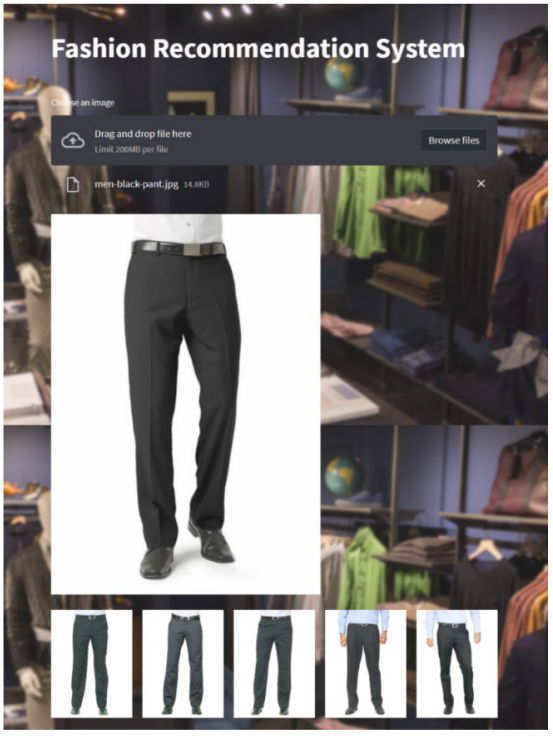


Figure 5.11.: Sample outputs of items like Jersey, Socks, Shirt and Formal Pant

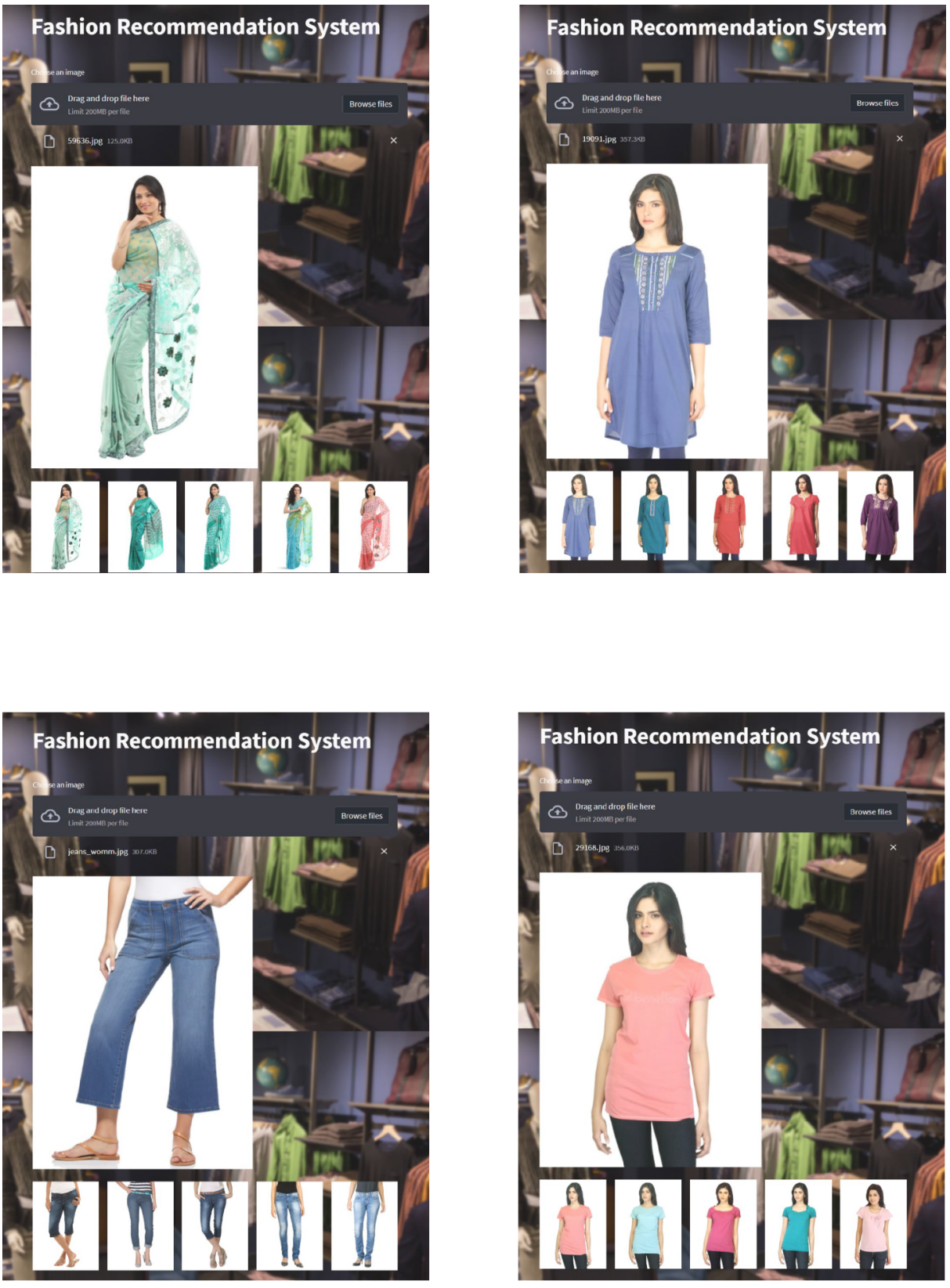


Figure 5.12.: Sample outputs of items like Saree, Kurti, Jeans and Top

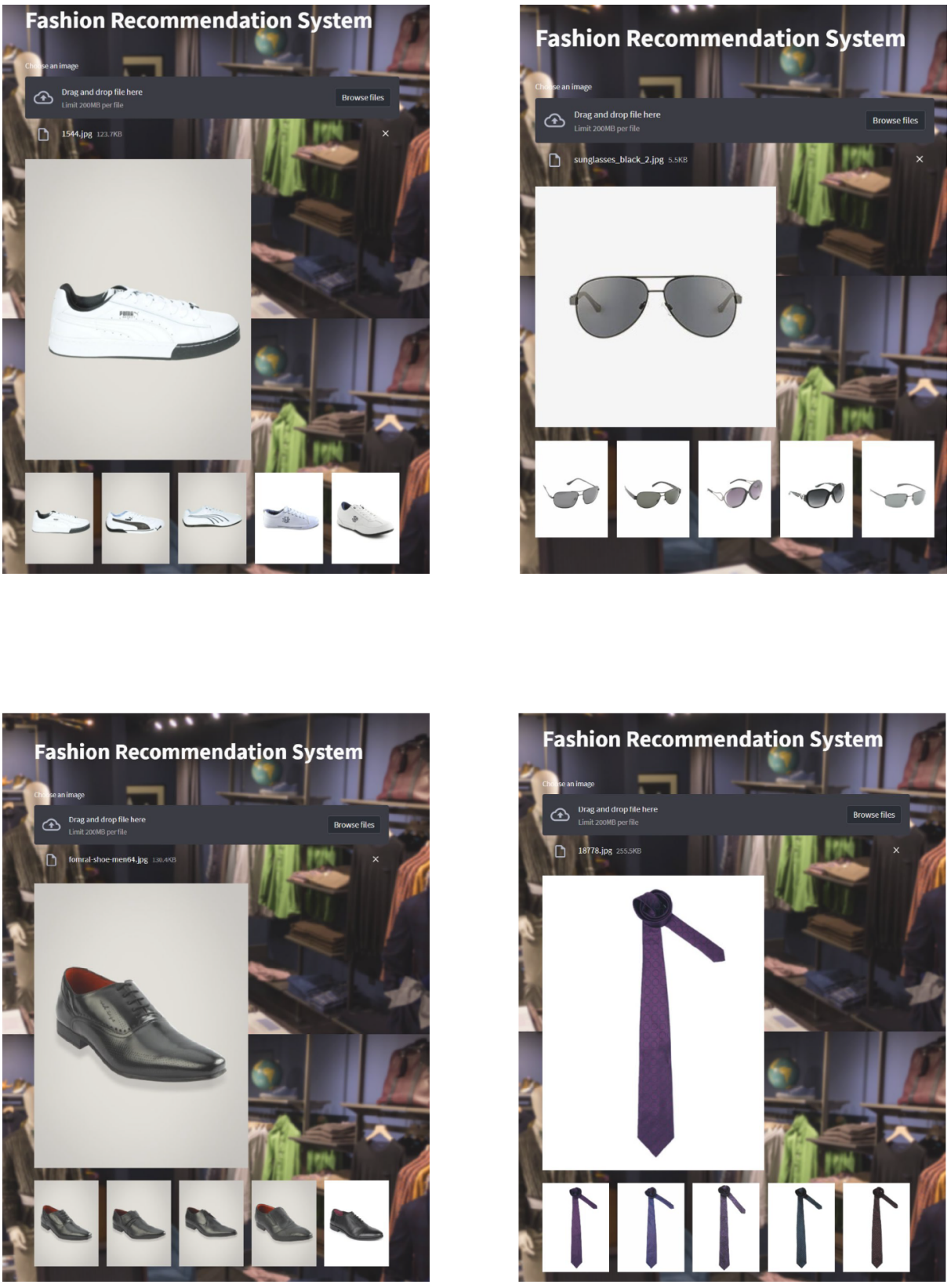


Figure 5.13.: Sample outputs of items like Shoes, Sunglasses, Formal shoes and Tie

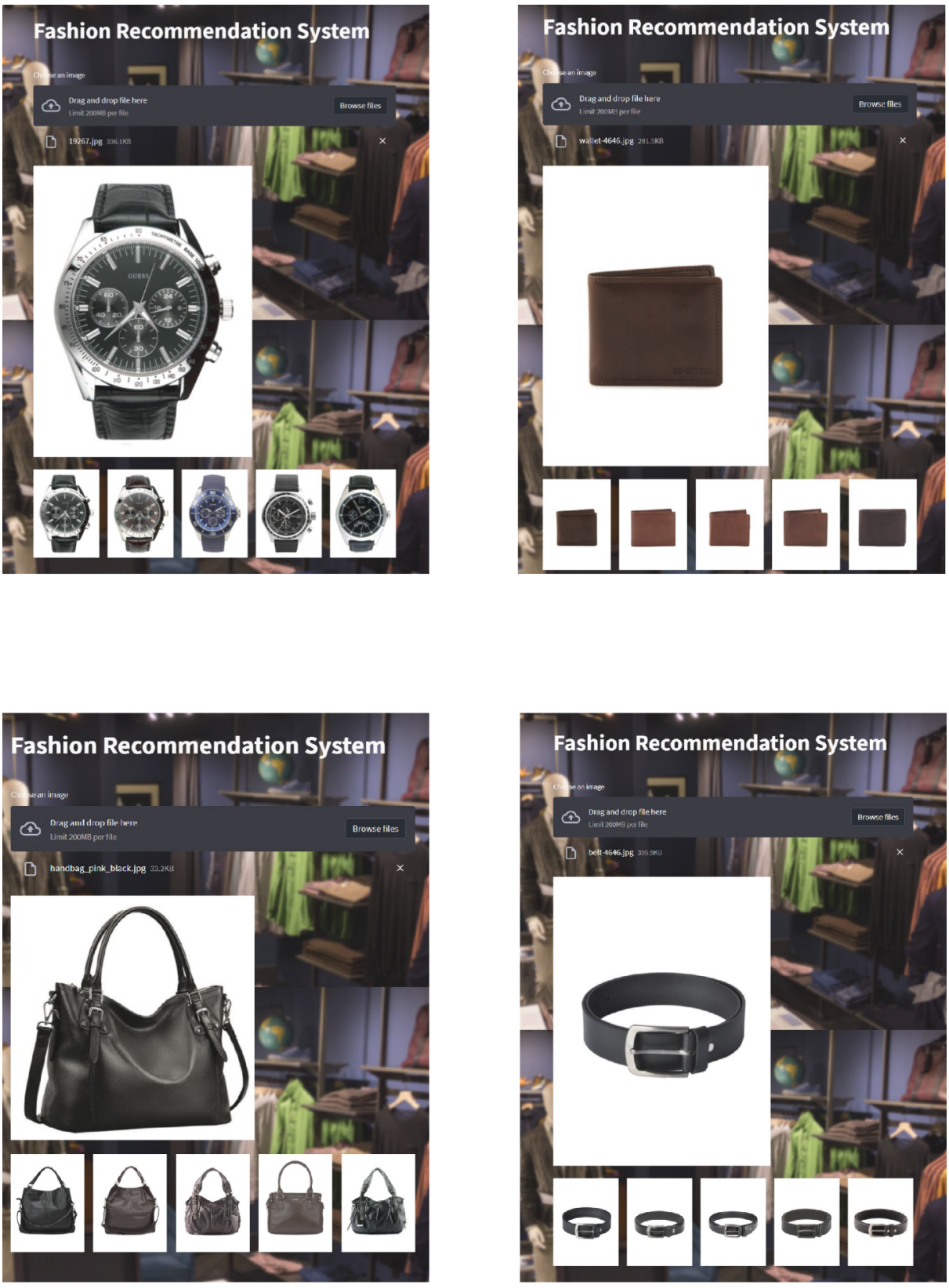


Figure 5.14.: Sample outputs of items like Watch, Wallet, Handbag and Belt

These were some sample outputs that we provided above. As we can see the recommendations generated by the system are very similar to the given input. The system takes into various parameters like colour, design pattern, body figure, texture, and many features. We have 2048 features for each and every product present in our dataset. The system does not take more than five seconds to generate recommendations. So, we can say that our system works well based on our results.

5.8.2. Analysis of Working Principle

The system is mainly based on Convolutional Neural Network (CNN) based on ResNet and K-Nearest Algorithm (KNN) algorithm. Let's analyze the working principle of each component:

5.8.2.1. ResNet (CNN)

ResNet is a deep learning architecture primarily used for image classification and feature extraction tasks. It addresses the problem of vanishing gradients in deeper networks by introducing skip connections or residual connections. These connections allow the network to learn residual mappings, which helps improve the gradient flow and enables the training of much deeper networks.

In the fashion recommendation system, the ResNet CNN is likely used for learning and extracting features from fashion images. It can be trained on a large dataset of fashion images, where the network learns to identify different visual patterns, colours, textures, and other relevant features present in the images. The network's deeper layers capture high-level representations, while the earlier layers capture low-level details.

Once the ResNet CNN is trained, it can be used to extract feature vectors from input fashion images. These feature vectors represent the learned features and can be considered as compact representations of the fashion items.

5.8.2.2. K-Nearest Neighbors (KNN) Algorithm

The k-nearest neighbours algorithm is a simple yet effective machine learning algorithm used for classification and regression tasks. It operates based on the principle that similar data points are likely to belong to the same class or have similar properties.

In the fashion recommendation system, the KNN algorithm is likely used to find the nearest neighbours of a given fashion item based on its feature vector obtained from the ResNet CNN. The feature vectors of fashion items from a dataset or catalogue are stored in a feature space. When a user provides an input image or selects an item, the system calculates the feature vector for that item using the trained ResNet

CNN.

Next, the KNN algorithm is used to search for the k most similar feature vectors (nearest neighbours) in the feature space. The similarity is typically computed using metrics such as Euclidean distance or cosine similarity. The k nearest neighbours can represent fashion items that are visually similar or share common features with the input item.

Finally, the system recommends fashion items based on the identified nearest neighbours. This can be achieved by suggesting items that are visually similar or by considering additional factors such as user preferences, popularity, or compatibility with other items.

The fashion recommendation system combines the feature extraction capabilities of a ResNet CNN with the similarity-based approach of the KNN algorithm to provide personalized fashion recommendations based on visual similarities between fashion items.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1. INTRODUCTION

The field of fashion recommendation systems has witnessed significant advancements in recent years, driven by the growing demand for personalised shopping experiences. These systems aim to assist users in discovering fashion items that align with their preferences, ultimately enhancing their satisfaction and engagement. In this project, we propose a fashion recommendation system that utilises the power of the ResNet50 architecture for feature extraction and integrates it with the k-nearest neighbours (KNN) algorithm for item recommendations.

The experimental results presented in this project serve as a comprehensive evaluation of the proposed fashion recommendation system. Our objective is to assess its performance, accuracy, and ability to deliver personalised recommendations to users. Through a meticulous experimental setup, we have collected and preprocessed a diverse fashion dataset, ensuring adequate representation of different fashion categories and styles.

In our experiments, we use the pre-trained ResNet50 model to extract high-level features from fashion images in the dataset. We unfreezed the last few layers of ResNet50 and trained it with our own dataset to increase its accuracy and efficiency.

To evaluate the system's performance, we employ the KNN algorithm to search for the K most similar fashion items in the feature database based on feature similarity. This process takes into account the extracted features from a user's query image and retrieves the most relevant fashion items. The retrieved items are then ranked according to their proximity to the query image, using distance metrics such as Euclidean distance.

In addition to evaluating the system's accuracy through metrics such as precision, efficiency, and robustness, we also consider user satisfaction and engagement as vital indicators of its effectiveness. We incorporate user feedback mechanisms, such as surveys, to gauge the system's ability to provide personalised recommendations that align with user preferences and enhance their overall shopping experience.

This project aims to demonstrate the potential of combining deep learning and collaborative filtering techniques in fashion recommendation systems. By leveraging the power of the ResNet50 architecture for feature extraction and the KNN algorithm for similarity-based recommendations, we strive to enhance the user's shopping journey by delivering accurate, relevant, and personalised fashion recommendations.

6.2. TESTING WITH DIFFERENT TYPES OF DATASETS

In order to thoroughly evaluate the performance and robustness of our fashion recommendation system, we conducted testing with different types of datasets. By employing diverse datasets, we aimed to assess the system's ability to handle various fashion categories, styles, and user preferences. This comprehensive testing approach provides insights into the system's adaptability and effectiveness in delivering accurate and personalised fashion

recommendations.

6.2.1. Dataset Selection

Selecting a diverse dataset was necessary for our testing, so we selected multiple datasets that encompassed a wide range of fashion items. These datasets were carefully curated to include diverse fashion categories, such as clothing, shoes, accessories, and more. We considered datasets that represent different styles, trends, and demographics to ensure a comprehensive evaluation.

6.2.2. Dataset Preprocessing

Prior to testing, we performed preprocessing steps on each dataset to ensure consistency and compatibility with our fashion recommendation system. This involved resizing and normalising the images to the same dimension before the feature extraction process to maintain uniformity in input data. Additionally, we ensured that the datasets were properly labelled with corresponding fashion items to enable accurate evaluation and comparison.

6.2.3. Testing Methodology

During the testing phase, we followed a systematic approach to evaluate the performance of our fashion recommendation system. We randomly divided each dataset into training and testing sets, ensuring an appropriate distribution of fashion items across both sets. This division allowed us to train the system on a subset of the data and evaluate its recommendations on unseen data.

6.2.4. Performance Metrics

To assess the system's performance with different datasets, we employed various evaluation metrics. These metrics included precision, efficiency, and robustness, which measure the system's accuracy in recommending relevant fashion items. Additionally, we considered metrics such as coverage and diversity to evaluate the system's ability to provide a broad range of recommendations across different fashion categories and styles.

6.2.5. Results and Analysis

The testing results with different types of datasets revealed valuable insights into the performance of our fashion recommendation system. By analysing the precision, efficiency, and robustness, we assessed the system's accuracy in recommending fashion items across diverse datasets. We also examined coverage and diversity metrics to understand the system's ability to cater to a wide range of user preferences and provide comprehensive recommendations.

Furthermore, we analysed the system's performance with specific fashion categories or styles within each dataset. This analysis allowed us to identify any potential biases or limitations in the system's recommendations and provide insights for future improvements.

6.2.6. Discussion of Findings

Based on the testing results, we discussed the strengths and limitations of our fashion recommendation system with different types of datasets. We highlighted the system's ability to handle diverse fashion categories according to user preferences. Additionally, we addressed any challenges or areas for improvement that were identified during the testing process.

By conducting testing with different types of datasets, we obtained a comprehensive evaluation of our fashion recommendation system's performance and effectiveness. The results demonstrated the system's ability to deliver accurate and personalised recommendations across diverse fashion categories and styles. This testing approach allowed us to validate the system's robustness and provided insights for further enhancements, ensuring an enhanced shopping experience for users across a wide range of fashion preferences.

6.3. TESTING OF DIFFERENT ALGORITHMS

To evaluate the performance and compare the effectiveness of our fashion recommendation system, we conducted testing using different algorithms. By testing multiple algorithms, we aimed to assess their capabilities in providing accurate and personalised fashion recommendations, ultimately identifying the most suitable algorithm for our system. This comprehensive testing approach helps us understand the strengths and limitations of different algorithms in the context of fashion recommendation.

For our testing, we carefully selected a set of algorithms commonly used in fashion recommendation systems. These algorithms include ***Brute Force***[5] and ***K-D Tree***[5] (short for k-dimensional tree). Each algorithm has its own unique approach to the recommendation, leveraging different techniques to analyse user preferences and item characteristics.

6.3.1. Brute Force Algorithm:

The brute force algorithm, also known as the exhaustive search or linear search algorithm, involves comparing each data point in the dataset to the query point to find the nearest neighbours. It calculates the distance between the query point and all the data points and selects the ones with the shortest distance as the nearest neighbours. This approach requires examining every data point in the dataset, resulting in a time complexity of $O(N)$[5], where N is the number of data points. As a result, the brute force algorithm can become computationally expensive for large datasets.

6.3.2. KD-Tree Algorithm:

The KD-tree algorithm is a data structure-based approach that organises the dataset in a hierarchical manner to facilitate faster nearest-neighbour searches. It partitions the data points based on their coordinates along different dimensions. At each level of the tree, a splitting hyperplane is chosen to divide the dataset into two subspaces. This process is repeated recursively until each subspace contains only a small number of data points.

During the search process, the KD-tree algorithm navigates the tree based on the query point's coordinates, effectively reducing the search space by ignoring irrelevant subtrees. It performs a series of binary searches down the tree to identify the leaf node closest to the query point. Then, it backtracks to search for other potential nearest neighbours in the remaining tree branches.

The KD-tree algorithm offers significant improvements in search time compared to brute force, especially for high-dimensional datasets. The average time complexity of the KD-tree algorithm for nearest neighbour search is O(log N)[5], where N is the number of data points. However, the performance of the KD-tree algorithm can degrade when the dataset has irregular distributions or imbalanced partitions, leading to suboptimal tree structures.

In the testing phase, we found that KD-tree works faster than brute force for the recommendation process but its results were not accurate as the brute force, So, we ended up using the brute force algorithm instead of the K-D tree. Although our dataset is big as it consists of over forty thousand images, it is still smaller as compared to the real-world datasets. Thus, the brute force algorithm was more suitable to work on our dataset as compared to the KD-tree algorithm.

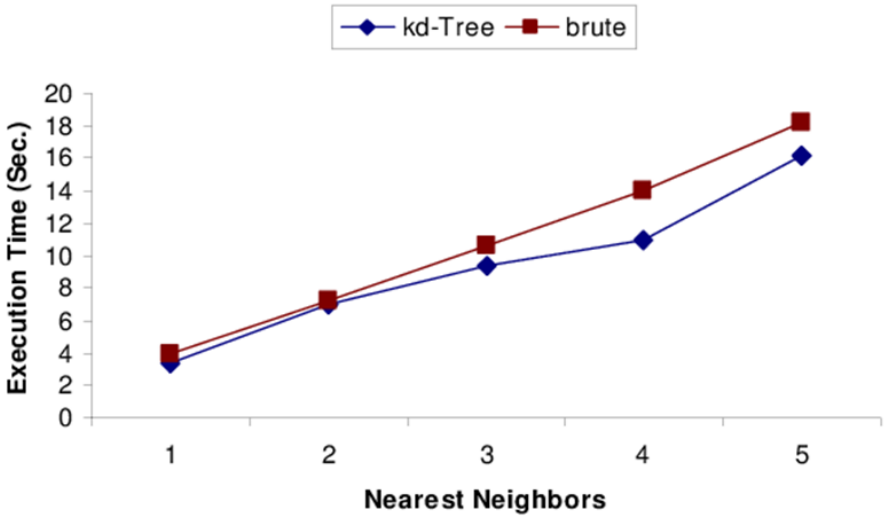


Figure 6.1.: KD-Tree vs. Brute Force

6.4. RESULT OF RECOMMENDATION USING CNN (RESNET-50) AND KNN (EUCLIDEAN DISTANCE)

We also had options of using ResNet101 or ResNet152 instead of ResNet50. The main reason to choose ResNet50 over others as it is much faster and provides almost the same level of accuracy when compared to ResNet101 or ResNet152. Another reason was that it uses less computing resources and we did not have that many resources to try others. So, we used the most famous and successful version of ResNet, i.e., ResNet50 for training our dataset. ResNet50 not only met our expectations but it surpassed them. ResNet is currently one of the best models for feature extraction.

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 6.1.: Top-1 and Top-5 Error Rate on ImageNet Validation Set[1]

We had the option of using two different techniques for the nearest neighbour for the recommendation process. The first one was KNN (K-Nearest Neighbours) and Annoy (Approximate Nearest Neighbours). KNN is a traditional algorithm for the exact nearest neighbour search. It calculates the distances between a query point and all the points in the dataset and selects the k nearest neighbours based on the distances. Annoy is a library offered by Spotify and is an approximate nearest neighbour algorithm that provides an efficient way to find approximate nearest neighbours. It builds an index structure, such as a binary tree or a forest of trees, to organise the data points and optimise the search process. Annoy trades off some level of accuracy for improved efficiency.

KNN provides an exact nearest neighbour search, ensuring accurate results. It considers the k nearest neighbours based on their distances, which can be useful for certain applications where precision is crucial. Whereas, annoy is an approximate nearest neighbour search algorithm, meaning it may not guarantee the exact nearest neighbours. However, it provides an efficient way to find close approximate neighbours, which can still be suitable for many applications.

The outputs for both KNN and ANN were the same and KNN was taking less time to recommend the results. ANN would be faster for larger datasets which contain millions of images but in our case, we had only 44K images in our dataset. So, we selected KNN over ANN because it was faster for our dataset.

The metric we used to calculate the distances between the query point and the data points in the training set is the Euclidean distance. We also had other options like Cosine similarity and Manhattan distance which measures the cosine of the angle between two vectors instead of distance.

6.4.1. Euclidean distance

This metric calculates the straight-line distance between two points in a multidimensional space. It is based on the Pythagorean theorem and is defined as the square root of the sum of squared differences between corresponding features of two points.

Euclidean distance is a widely used metric due to its simplicity, intuitive interpretation, and applicability in various domains. Its calculation is straightforward, and it provides valuable insights into the relationships between points in multidimensional space.

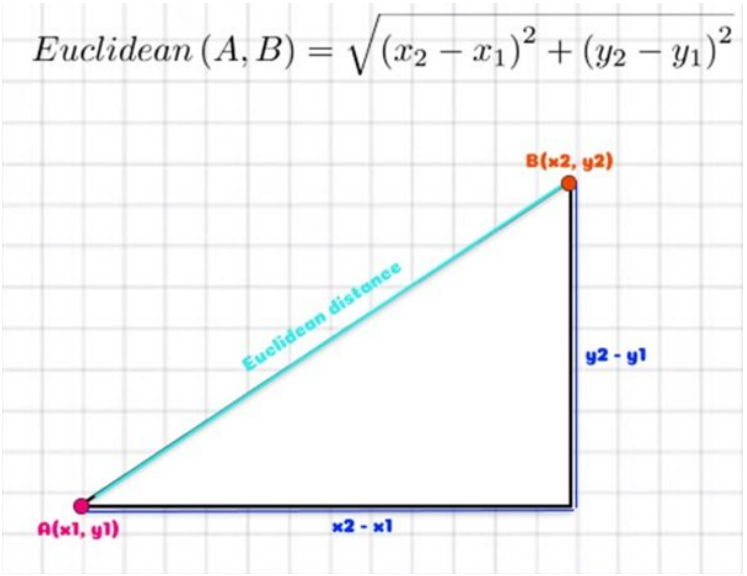


Figure 6.2.: Euclidean Distance

6.4.2. Cosine similarity

This metric measures the similarity between two vectors by calculating the cosine of the angle between them. It determines similarity based on the direction of the vectors rather than their magnitude. Cosine similarity is widely employed in recommendation systems, information retrieval, text analysis, and other domains where vector representations are utilized.

Cosine similarity is a powerful metric for capturing similarity based on the direction of vectors. Its range of -1 to 1 allows for flexible interpretation and enables effective comparisons in various domains where vector representations are employed.

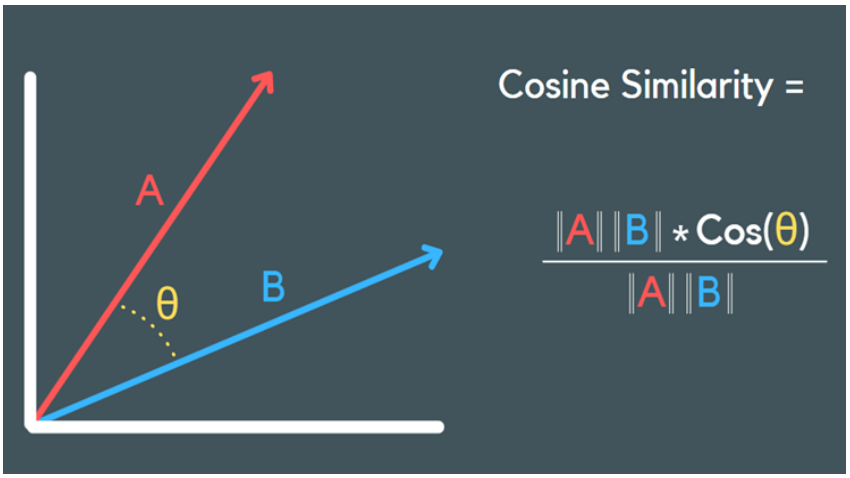


Figure 6.3.: Cosine Similarity

6.4.3. Manhattan Distance

This metric measures the distance between two points in a grid-like or Euclidean space. It is named after the urban street layout of Manhattan, where one can only travel along the perpendicular grid lines.

Manhattan distance provides a straightforward and intuitive measure of distance in grid-like or Euclidean spaces. Its application in various fields reflects its practicality and usefulness in scenarios where movement is restricted along perpendicular directions.

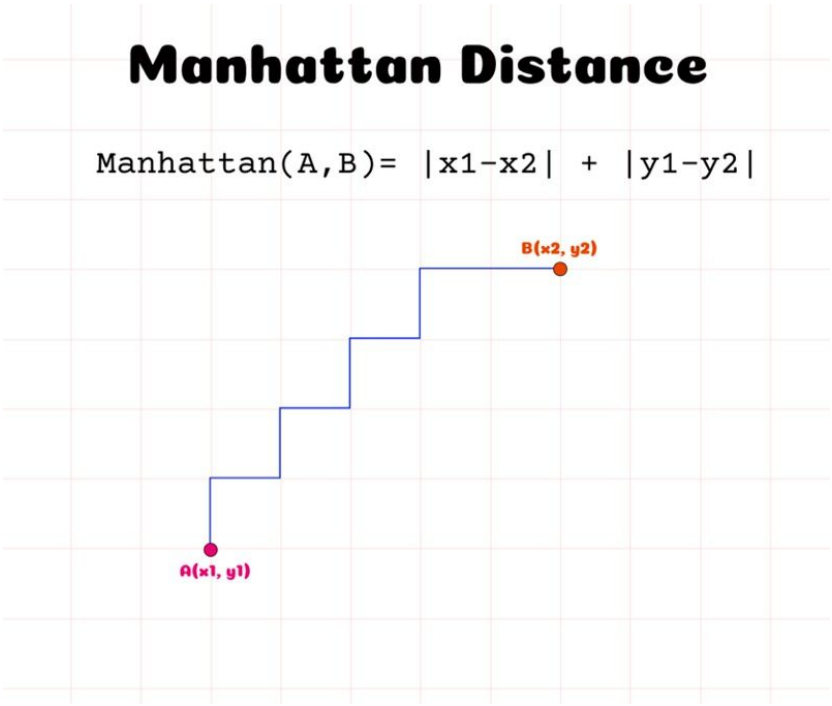


Figure 6.4.: Manhattan distance

Choosing the Euclidean distance metric for our project was driven by several factors that made it a favourable choice. Firstly, its ease of implementation was a crucial consideration. The straightforward calculation based on the Pythagorean theorem made it simple to integrate into our project without significant complexity or additional dependencies.

Moreover, the understandability of Euclidean distance played a role in our decision. Its intuitive interpretation as the straight-line distance between two points in a multidimensional space made it easy for our team to grasp and explain to stakeholders. This facilitated effective communication and collaboration throughout the development process.

In terms of performance, we found that Euclidean distance delivered good results in both speed and accuracy. The efficiency of its computation, especially in lower-dimensional spaces, allowed us to process large datasets or perform real-time calculations without significant performance bottlenecks. Additionally, the Euclidean distance metric showed satisfactory accuracy in capturing the relationships between data points, particularly when the

data exhibited a clear geometric structure and meaningful feature magnitudes.

While Euclidean distance served us well in our project, it's important to note that the choice of distance metric should always depend on the specific characteristics of the data and project requirements. Different metrics, such as Manhattan distance for grid-like spaces or cosine similarity for text data, may offer better performance in certain scenarios. Nonetheless, our selection of Euclidean distance was driven by its ease of implementation, understandability, and satisfactory results in terms of both speed and accuracy.

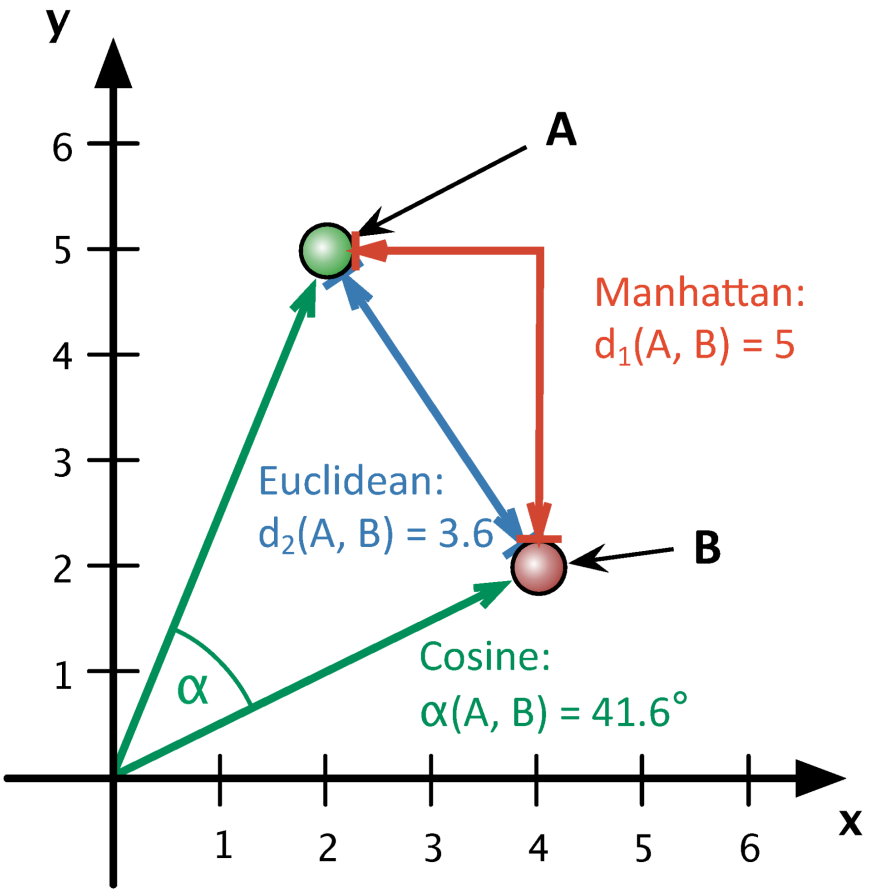


Figure 6.5.: Difference between Euclidean, Cosine and Manhattan

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

3.1. CONCLUSION

We have presented a novel framework for fashion recommendation that is driven by data, visually related and simple effective recommendation systems for generating fashion product images. The proposed approach uses a two-stage phase. Initially, our proposed approach extracts the features of the image using a CNN classifier[3] i.e., for instance allowing the customers to upload any random fashion image from any E-commerce website and later generating similar images to the uploaded image based on the features and texture of the input image. Such research must go forward to facilitate greater recommendation accuracy and improve the overall experience of fashion exploration for direct and indirect consumers alike.

Product recommendation engines are the best way to deliver customers an improved user experience. Through machine learning, manual curation, and specific algorithms, a product recommendations engine can help bring customers the relevant products they want or need[4].

Recommending clothes using images has made tremendous progress over the years. E-commerce websites are hugely benefited from this. As research in this field continues, more and more interesting methods have come to light. Work once started using text-to-visual methods with image processing and use of neural network-based methods, turned to visual methods with image processing and use of neural networks, convolutional neural networks and now transfer learning with deep neural networks.

The system aims to assist users in discovering fashion items that align with users' preferences. The system recommends the best possible output combinations to users and helps to enhance their shopping experience and increase their likelihood of finding suitable and appealing fashion items based on their interests. The system recommends similar products based on users' searches or users can simply take photos of the appealing clothes they saw on the magazine, web page or even street, then get the recommended clothing with similar fashion and style in seconds. When people find a product they like but don’t know what it is called or they want to find exactly a product like it or similar to it, the Fashion Recommendation System provides a convenient way to help find those clothes to the users.

Fashion Recommendation systems have the potential to explore new opportunities for retailers also by enabling them to provide customised recommendations to consumers based on information retrieved from the internet. They help consumers instantly find the products and services that closely match the user's choices. The system provides personalised and relevant fashion suggestions and enhances the shopping experience for users. Overall the fashion recommendation system offers a personalised and efficient solution for users seeking fashion guidance. Through the integration of machine learning algorithms, the system aims to provide accurate and relevant fashion recommendations, enhancing the shopping experience and also user satisfaction in this dynamic fashion world.

3.2. FUTURE SCOPE

- Reduce the latency of the end-to-end application. Try different approaches like building your model with a good score.

- Since we have approximately 44k images it takes generally 3 to 4 hours (may vary according to a system's specification) for the feature extraction process of every product's images. This time taken can be shortened by using higher systems with higher processing power. But in real life, we have millions of images for one only E-commerce website for which this won't be sufficient[10]. Hence, we can use Spotify's library called ***Annoy*** and it uses Approximate Nearest Neighbours (ANN) instead of K-Nearest Neighbours (KNN).

- The next steps will be to integrate this system with E-commerce websites where it would recommend similar products to users as per their searches or input.

- We can also build a website using Flask or any other web application framework and have the recommendation system running in the backend. Therefore users can select their choice of products and the website will suggest similar items to them.

- Train the model on different segments of products such as furniture, utensils, etc. The program would work with the same efficiency without any major changes.

References

1. **He, Kaiming, et al. "Deep Residual Learning for Image Recognition."** *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.*

2. **Russakovsky, Olga, et al. "ImageNet Large Scale Visual Recognition Challenge."** *International Journal of Computer Vision (IJCV), vol. 115, no. 3, 2015, pp. 211-252.*

3. **Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks."** *Advances in Neural Information Processing Systems (NIPS), vol. 25, 2012, pp. 1097-1105.*

4. **Liu, Ziwei, et al. "Fashion-Forward: Forecasting Visual Style in Fashion."** *Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.*

5. **Andreeva, Elena, et al. "Extraction of visual features for recommendation of products via deep learning."** *Analysis of Images, Social Networks and Texts: 7th International Conference, AIST 2018, Moscow, Russia, July 5–7, 2018, Revised Selected Papers 7. Springer International Publishing, 2018.*

6. **Shankar, Devashish, et al. "Deep learning based large scale visual recommendation and search for e-commerce."** *arXiv preprint arXiv:1703.02344 (2017).*

7. **Covell, Michele, et al. "Deep Learning for Fashion Recommendation and Personalization."** *arXiv preprint arXiv:1606.03291, 2016.*

8. **Chauhan, Yashaswi, et al. "Fashion Recommendation System Using Deep Learning."** *Proceedings of the International Conference on Intelligent Computing and Applications, Springer, 2020.*

9. **Zhang, Jianming, et al. "FashionNet: Personalized Outfit Recommendation with Deep Learning."** *Proceedings of the ACM International Conference on Multimedia (MM), 2017.*

10. **Ding, Zhongqian, et al. "Deep Learning for Fashion Recommendation on Real-World Data."** *Proceedings of the IEEE International Conference on Big Data (Big Data), 2015.*

11. **Sulthana, Razia. "A review on the literature of fashion recommender system using deep learning."** *International Journal of Performability Engineering 17.8 (2021): 695.*

12. **Chakraborty, Samit, et al. "Fashion Recommendation Systems, Models and Methods: A Review."** *Informatics, vol. 8, no. 3, July 2021, p. 49.*

13. **Param Agarwal. (2022, June). Fashion Product Images Dataset,** *Version 1. Retrieved January 11, 2023.*