

الجامعة المصرية للتعليم الإلكتروني الأهلية

Faculty of Computer & Information Technology

3D OutFit Maker App

By:

- | | |
|----------------------------|----------|
| 1. Mahmoud Mostafa Mofadal | 20-01341 |
| 2. Waleed Ahmed Korashy | 20-00863 |
| 3. Mohamed Mamdouh Mohamed | 18-00552 |
| 4. Mahmoud Saad NorEldin | 20-00491 |
| 5. Holy Rezk Melek | 20-00353 |
| 6. Alaa Hisham Said | 20-01163 |
| 7. Karoleen Eshak Helmy | 20-01142 |

Under Supervision of:

Prof. Mohammed Attia

Professor of Computer and Information Technology Egyptian E-Learning University

Eng. Mohammed Hussein

Assistant Lecturer in Faculty of Computer and Information Technology Egyptian E-Learning
University

Assiut 2024

Acknowledgement

In the name of God, we would to thank God before anything else for granting us success in achieving and completing this project.

Writing this book would have been impossible without the help of many people. We would like to commend the help of some honest people who supported us and gave us a lot of advice, guidance and contributed to the success of this project. Therefore, they deserve our utmost respect and appreciation.

So special thanks for Prof. Mohammed Attia for giving us many important ideas, tips and advice that helped us a lot during the work on this project.

We would also like to express our deep gratitude to Eng Mohammed Huessin, who started working with us directly and introduced us to the work methodology and was supportive of us permanently.

Acknowledgement	3
Chapter 1	8
Abstract:	9
INTRODUCTION:.....	9
Chapter 2	14
BACKGROUND	15
Mobile Application	15
Machine Learning[20]:-	19
Chapter 3	45
LITERATURE REVIEW	45
3.1. Paper 1	50
3.2. Paper 2	50
3.3. Paper 3	50
3.4. Paper 4	51
3.5. Paper 5	51

3.6. Paper 6	52
3.7. Paper 7	53
3.8. Paper 8	53
3.9. Paper 9	54
3.10. Paper 10.....	54
Chapter 4	56
IMPLEMENTATION:	57
4.1RECOMMENDATION ENGINE:	58
Pre-requisites and Dataset	59
Configuring the model	62
Extracting Features	64
Nearest Neighbours Algorithm	65
Uploading the image	66
4.2 SIZE PREDICTION	68
Decision Tree:-.....	72
Importing Libraries:	73
loaded and preparing the data:.....	73
Replacing Categorical Values with Numerical Values:	75
Train-Test Split:.....	76
Training a Decision Tree classifier and Evaluate:	77
input data and then providing predictions:	78
Deployment Flutter (Mobile App)	79
onboarding Screen	79
Login Screen.....	81
Forget password	82
Register screen	83
Home Screen.....	84
Design studio screen.....	86
Category screen.....	89

Order screen	91
Profile screen	92
Payment screen	94
Wishlist screen	95
Libraries used in the application :-	96
References:	99

List of figure:

- 1- Diagram for Image Preprocessing
- 2- one of most common of classification
- 3- Classification in Machine Learning
- 4- Regression in Machine Learning
- 5- DBSCAN Clustering
- 6- convolutional neural networks (CNN)
- 7- ResNet-50
- 8- Convolutions in CNN
- 9- Working of GlobalAveragePooling2D¹
- 10- Overfitting
- 11- Example training model
- 12- Flow-chart for Image Preprocessing
- 13- Diagram for Recommendation Engine
- 14- Some samples from the dataset
- 15- Representing image as a feature vector
- 16- Configuring the model
- 17- Working of GlobalAveragePooling2D¹
- 18- Extract features
- 19- choosing model
- 20- upload the image
- 21- upload the image
- 22- upload the image
- 23- Quiz Size Recommendation
- 24- import library

- 25- loaded the data
- 26- replace value
- 27- train-test-split
- 28- training & evaluate
- 29- input and prediction
- 30- Diagram for Mobile App
- 31- onboarding screen
- 32- Login screen
- 33- Forget password screen
- 34- Register screen
- 35- Home screen
- 36- Design studio screen
- 37- Category screen
- 38- Order screen
- 39- Profile screen
- 40- Payment screen
- 41- Wishlist screen

Chapter 1

Abstract:

Fashion recommendation systems are pivotal for enhancing online shopping using deep learning, specifically ResNet-50. Our Fashion Recommendation System (FRS) employs ResNet-50 to process and represent fashion images' key attributes like color and style. Content-based recommendations suggest visually similar items based on user preferences.

The FRS evolves through machine learning, delivering improved user engagement and conversion rates in e-commerce fashion. This paper reviews and explores fashion recommendation systems and filtering techniques, filling a gap in academic literature.

It serves as a valuable resource for machine learning, computer vision, and fashion retailing professionals. We trained our model using a large image dataset, including the multiple category labels, descriptions and high-res images of fashion products, which consists of 44k+ images. The results have been promising, indicating potential real world applications such as searching for a product using its digital copy in large sets of images.

INTRODUCTION:

Fashion is a dynamic and ever-evolving industry where trends change rapidly, making it challenging for consumers to navigate the vast array of clothing options available online. In response to this, fashion recommendation systems have emerged as essential tools for enhancing the online shopping experience. These systems leverage advanced technologies, such as deep learning, to provide users with personalized clothing suggestions that align with their unique tastes and preferences.

Deep learning techniques have played a pivotal role in revolutionizing the field of fashion recommendation. Among the various deep learning architectures, the ResNet-50 convolutional neural network (CNN) has gained prominence for its exceptional ability to extract rich and discriminative features from images. By leveraging ResNet-50, fashion recommendation systems can analyze and understand the visual characteristics of clothing items, including color, texture, pattern, and style, with remarkable accuracy.

This introduction sets the stage for exploring the Fashion Recommendation System (FRS) that utilizes ResNet-50 as its core technology. In the following sections, we will delve into how ResNet-50 is employed to process and encode fashion images, creating a robust feature representation that forms the foundation of personalized clothing recommendations. We will also explore the concept of content-based recommendation,

wherein the feature representations extracted by ResNet-50 are used to suggest fashion items that closely match users' visual preferences.

Furthermore, we will discuss the adaptability and continuous improvement of the FRS through machine learning algorithms. This adaptability ensures that the system remains up-to-date with changing fashion trends and user preferences, ultimately resulting in improved user engagement, increased conversion rates, and enhanced customer satisfaction in the e-commerce fashion domain.

By implementing the technology of ResNet-50 and advanced recommendation techniques, fashion recommendation systems have the potential to transform the online shopping experience, making it more enjoyable, personalized, and efficient for consumers. This exploration aims to provide insights into how ResNet-50 and deep learning are revolutionizing the fashion industry by offering tailored and visually appealing recommendations to users.

RELATED WORK

There are some previous works related to building of recommendation systems. Smart Clothing Recommendation System with Deep Learning In order to recommend a cloth, we develop two inception based convolutional neural networks as prediction part and one feed forward neural network as recommender. In this study, we reach to 98% accuracy on color prediction, 86% accuracy on gender and cloth's pattern predictions and 75% accuracy on clothing recommendation.

Deep Fashion Recommendation System with Style Feature Decomposition. Due to the mixed information of style and category, however, the clothes vector often recommends clothes that do not match. To solve this problem, we propose a style feature extraction (SFE) layer, which effectively decomposes the clothes vector into style and category.

Based on the characteristics the category information has small variations in the same class while being distinguished from other classes, we extract and remove the category information from the clothes vector to obtain more accurate style information.

SYSTEM ARCHITECTURE

The proposed system is divided into three main parts:

- Image pre-processing
- Recommendation Engine
- Web App

IMAGE PRE-PROCESSING

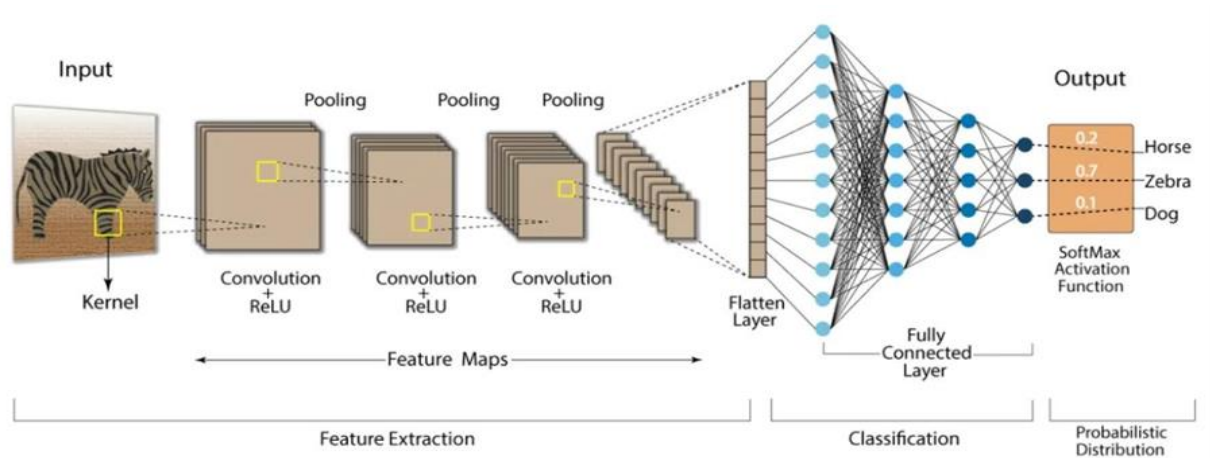


Fig. 1 Diagram for Image Preprocessing

Image preprocessing in the context of ResNet-50 involves preparing images for use with this deep learning architecture. Key steps include resizing images to a fixed size (e.g., 224x224 pixels), normalizing pixel values (typically in the range $[0, 1]$ or standardized), mean subtraction, data augmentation (applying random transformations), and consistent preprocessing during training and inference.

These steps ensure that the model processes images effectively, improves training stability, and generalizes well to new data. The steps to pre-process the image are as follows:

- **Read image:** Image provided by the user is taken as the input and stored in a temporary folder on the server
- **Resize image:** The saved image is resized in accordance with the input size the model is trained with i.e. (224 x 224)

- **Segmentation:** In this stage the saved image is converted from RGB to BGV to aid in better extraction of features.
- **Flatten:** In this stage after pre-processing the saved, the 2D matrix of the image is converted into Vector.

RECOMMENDATION ENGINE:

A recommendation engine filters the information using different algorithms and recommends the relevant items to users. It first captures the past behavior of a customer and recommends products which the users might be likely to buy. The working of recommendation engine is as follows:

Collection of Data:

Gathering data is the first step in creating the recommendation engine. Data can be either explicit or implicit data. Explicit data can be the input by users such as ratings and comments on products. And order history/return history, Cart events, Page views would be the implicit data, Click thru and search log. For every user visiting the site, dataset will be created.

Analyzing the Data:

The filtering of data is done by Real-time system analysis. The Real-time systems can process data as it's created. This system usually involves tools that can process and analyze streams of events. It is required to give in-the-moment recommendations.

Filtering the Data: The next step is to filter the data to get the necessary data to provide recommendations to the user. Content based filtering approach is used in this project. Content-based filtering uses meta data or characteristics of items to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

MobileAPP:

The Mobile app is created using the Streamlit library. Streamlit is a Python library that simplifies the process of building mobile applications and interactive dashboards. It allows you to create mobile interfaces directly from Python scripts with minimal code.

Chapter 2

BACKGROUND

Mobile Application (Flutter & Asp.Net)

A mobile application, also referred to as a mobile app or simply an app, is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch. Apps they're originally intended for productivity assistance such as email, calendar, and contact databases, but the public demand for apps caused rapid expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchases, so that there are now millions of apps available.

Apps are generally downloaded from application distribution platforms which are operated by the owner of the mobile operating system, such as the App Store (iOS) [29] or Google Play Store. Some apps are free, and others have a price, with the profit being split between the application's creator and the distribution platform. Mobile applications often stand in contrast to desktop applications which are designed to run on desktop computers, and they applications which run in mobile they browsers rather than directly on the mobile device.

Flutter. [25]

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible. In this project they develop the mobile application using flutter platform to detect images by training the DL model on this application and recognize it contains any letter of the alphabet , words , words or sentence easily.

Advantages of Flutter [30]

- Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfills all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below
- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may to be extended to other platform as they'll in the future.
- Flutter needs lesser testing. Because of its single code base, it is sufficient if they write automated tests once for both the platforms.

Flutter runs the code on Android [29] :-

The engine's C and C++ code are compiled with Android's NDK. The Dart code (both the SDK's and yours) are ahead-of-time (AOT) compiled into native, ARM, and x86 libraries. Those libraries are included in a "runner" Android project, and the whole thing is built into an .apk. When launched, the app loads the Flutter library. Any rendering, input, or event handling, and so on, is delegated to the compiled Flutter and app code. This is similar to the way many game engines work.

During debug mode, Flutter uses a virtual machine (VM) to run its code in order to enable stateful hot reload, a feature that lets you make changes to your running code without recompilation. You'll see a "debug" banner in the top right-hand corner of your app when running in this mode, to remind you that performance is not characteristic of the finished release app.

Flutter runs the code on IOS [29] :-

The engine's C and C++ code are compiled with LLVM. The Dart code (both the SDK's and yours) are ahead-of-time (AOT) compiled into a native, ARM library. That library is included in a "runner" iOS project, and the whole thing is built into an .ipa. When launched, the app loads the Flutter library. Any rendering, input or event handling, and so on, are delegated to the compiled Flutter and app code. This is similar to the way many games engines work.

During debug mode, Flutter uses a virtual machine (VM) to run its code in order to enable stateful hot reload, a feature that lets you make changes to your running code without recompilation. You'll see a "debug" banner in the top right-hand corner of your app when running in this mode, to remind you that performance is not characteristic of the finished release app.

Dart Flutter [18] :-

Dart is an open-source general-purpose programming language. It was originally developed by Google. Dart is an object-oriented language with C-style syntax. It supports programming concepts like interfaces and classes, unlike other programming languages. Dart doesn't support arrays. Dart collections can be used to replicate data structures such as arrays, generics, and optional typing.

Variable is named storage location and Data types simply refers to the type and size of data associated with variables and functions.

Dart language supports the following data types :-

- Numbers: - It is used to represent numeric literals Integer and Double.
- Strings: - It represents a sequence of characters. String values are specified in either single or double quotes.
- Booleans: - Dart uses the bool keyword to represent Boolean values true and false.
- Lists and Maps: - It is used to represent a collection of objects.

A simple List can be defined as below.

Dart libraries: -

Many Dart libraries like.

- Built-in types, collections, and other core functionality for every Dart program (dart: core).
- Richer collection types such as queues, linked lists, hash maps, and binary trees (Dart: collection).
- Encoders and decoders for converting between different data representations, including JSON and UTF-8 (Dart: convert).
- Mathematical constants and functions, and random number generation (Dart: math).
- File, socket, HTTP, and other I/O support for non-they applications (dart: io).

- `async` and `await` are keywords that provide a way to make asynchronous operations appear synchronous (Dart: `async` `await`)
- Lists that efficiently handle fixed-sized data (for example, unsigned 8-byte integers) and SIMD numeric types (dart: `typed_data`).

ASP.NET. [23]

ASP.NET extends .NET

ASP.NET extends the .NET platform with tools and libraries specifically for building web apps.

These are some things that ASP.NET adds to the .NET platform:

- **Base framework for processing web requests in C# or F#**
- **Web-page templating syntax**, known as Razor, for building dynamic web pages using C#
- **Libraries for common web patterns**, such as Model View Controller (MVC)
- **Authentication system** that includes libraries, a database, and template pages for handling logins, including multi-factor authentication and external authentication with Google, X, and more.
- **Editor extensions** to provide syntax highlighting, code completion, and other functionality specifically for developing web pages

Back-end code

When using ASP.NET your back-end code, such as business logic and data access, is written using C#, F#, or Visual Basic. Because ASP.NET extends .NET, you can use the large ecosystem of packages and libraries available to all .NET developers. You can also author your own libraries that are shared between any applications written on the .NET platform.

Key Features: -

- Extremely High Performance
- Support for Cross-Platform and Container Environments
- Independence from a particular language
- Security
- NET Web API (Application Programming Interface)

Cross-Platform & Container Support:

With the introduction of .NET Core, you can create ASP.NET applications and deploy them to Windows, Linux, and macOS. Containers (e.g., Docker, Kubernetes) are also supported.

libraries: -

- ASP.NET Core
- ASP.NET Web AP
- Entity Framework

Machine Learning[20]:-

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine Learning tutorial provides basic and advanced concepts of machine learning. Our machine learning tutorial is designed for students and working professionals. This machine learning tutorial gives you an introduction to machine learning along with the wide range of machine learning techniques such as Supervised and Unsupervised learning. You will learn about regression, classification models and clustering [19].

Classes of Machine Learning

Supervised Learning[17]

Supervised learning addresses the task of predicting targets y given input data. The targets, also commonly called labels, are generally denoted y . The input data points, also commonly called examples or instances, are typically denoted x . The goal is to produce a model f_θ that maps an input x to a prediction $f_\theta(x)$.

In probabilistic terms, they typically are interested in estimating the conditional probability $P(y|x)$. While it's just one among several approaches to machine learning, supervised learning accounts for the majority of machine learning in practice. Partly, that's because many important tasks can be described crisply as estimating the probability of some unknown given some available evidence:

- Predict cancer vs not cancer, given a CT image.
- Predict the correct translation in French, given a sentence in English.

- Predict the price of a stock next month based on this month's financial reporting data.

Even with the simple description ‘predict targets from inputs’ supervised learning can take a great many forms and require a great many modeling decisions, depending on the type, size, and the number of inputs and outputs. For example, they use different models to process sequences (like strings of text or time series data) and for processing fixed-length vector representations.

Classification[42]:-

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a “sub populations.” With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories.

Classification algorithms used in machine learning utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories. One of the most common applications of classification is

for filtering emails into “spam” or “non-spam”, as used by today's top email service providers.[16]

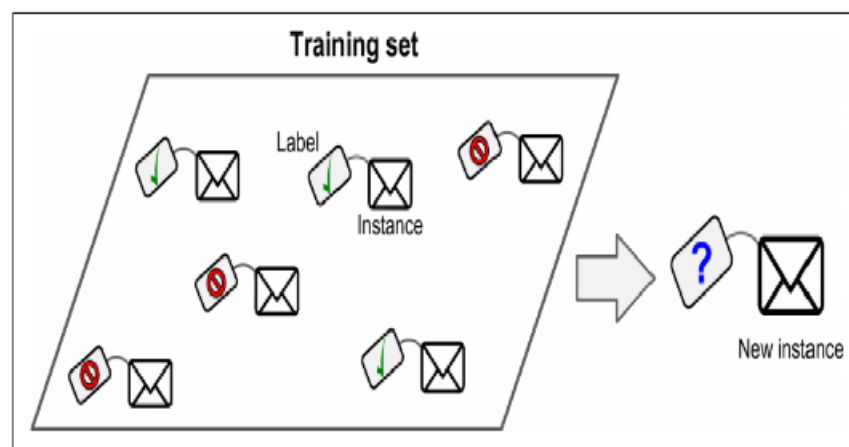


Figure 2 one of most common of classification

In short, classification is a form of “fashion recommendations”. Here, classification algorithms applied to the training data find similarities between fashion items based on their visual characteristics in future data sets.

They will explore classification algorithms in detail and discover how a text analysis software can perform actions like sentiment analysis -used for categorizing unstructured text by opinion polarity (positive, negative, neutral, and the like).

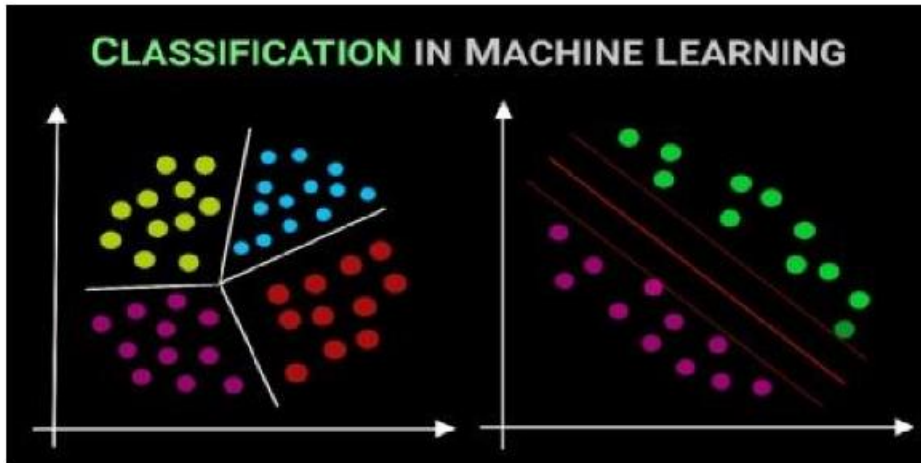


Figure 3 Classification in Machine Learning

Regression[41]

Perhaps the simplest supervised learning task to wrap your head around is Regression. Consider, for example, a set of data harvested from a database of home sales. They might construct a table, where each row corresponds to a different house, and each column corresponds to some relevant attribute, such as the square footage of a house, the number of bedrooms, the number of bathrooms, and the number of minutes (walking) to the center of town. Formally, they call one row in this dataset a feature vector, and the object (e.g. a house) it's associated with an example.

Feature vectors like this are essential for all the classic machine learning problems. They'll typically denote the feature vector for any one example x_i and the set of feature vectors for all our examples X . What makes a problem a regression is the outputs. Say that you're in

the market for a new home, you might want to estimate the fair market value of a house, given some features like these. The target value, the price of sale, is a real number. They denote any individual target, Y_i .

(Corresponding to example x_i) and the set of all targets y (corresponding to all examples X). When our targets take on arbitrary real values in some range, they call this a

regression problem. The goal of our model is to produce predictions (guess of the price, in our example) that closely approximate the actual target values .

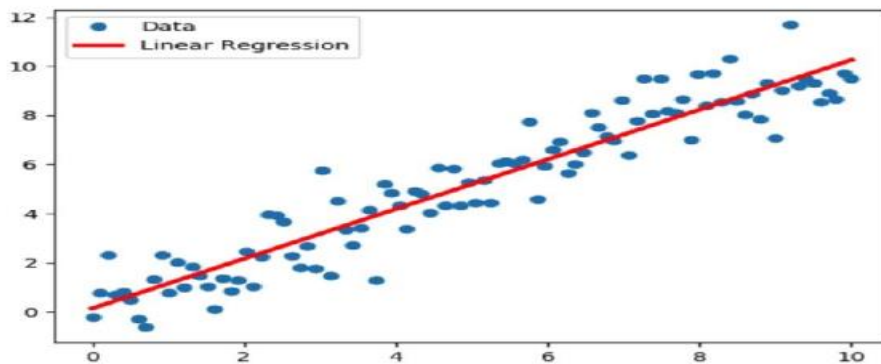


Figure 4 Regression in Machine Learning

Unsupervised Learning [15]:-

Unsupervised learning is a machine learning technique in which models are not supervised using a training dataset. Instead, models themselves find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allotted to act on that data without any supervision.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, they have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Clustering [14]:-

The goal of unsupervised learning is to discover hidden patterns in any unlabeled data. One of the approaches to unsupervised learning is clustering. Clustering groups data points based on their similarities. Each group is called a cluster and contains data points with high

similarity and low similarity with data points in other clusters. In short, data points of a cluster are more similar to each other than they are to the data points of other clusters. The goal of clustering is to divide a set of data points in such a way that similar items fall into the same cluster, whereas dissimilar data points fall in different clusters.

Clustering is crucial in multiple research fields in Bio Informatics such as analyzing unlabeled data which can be gene expressions profiles, biomedical images and so on. For example, clustering is often used in gene expression analysis to find groups of genes with similar expression patterns which may provide a useful understanding of gene functions and regulations, cellular processes and so on.

There are many algorithms available for data clustering which use different ways to establish similarity between data points. The clustering algorithms can be broadly divided into many categories such as connectivity model, centroid model, density model, distribution model, group model, graph-based model and so on.

Some of these are discussed below:

Connectivity model: This model assigns higher similarity to data points which are closer in one or multi-dimensional space than those points which are farther away. There are two approaches - first, it categories all data points into different clusters and then merges the data points in relation to the distances among them. Second, it categories all data points into one single cluster and then partitions them into different clusters as the distance increases. This model is easy to understand but has problems in handling large datasets. One example is hierarchical clustering and its variants.

Centroid model: It is an iterative clustering algorithm in which similarity is based on the proximity of a data point to the centroids of the clusters. K-means clustering is one example of this model. It needs a few clusters before running and then divides data points into these many clusters iteratively. Therefore, to use K-Means, users should acquire some prior knowledge about the dataset.

Density model: This model searches one or multi-dimensional space for dense regions (having many data points in a small region). A popular example of a density model is DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

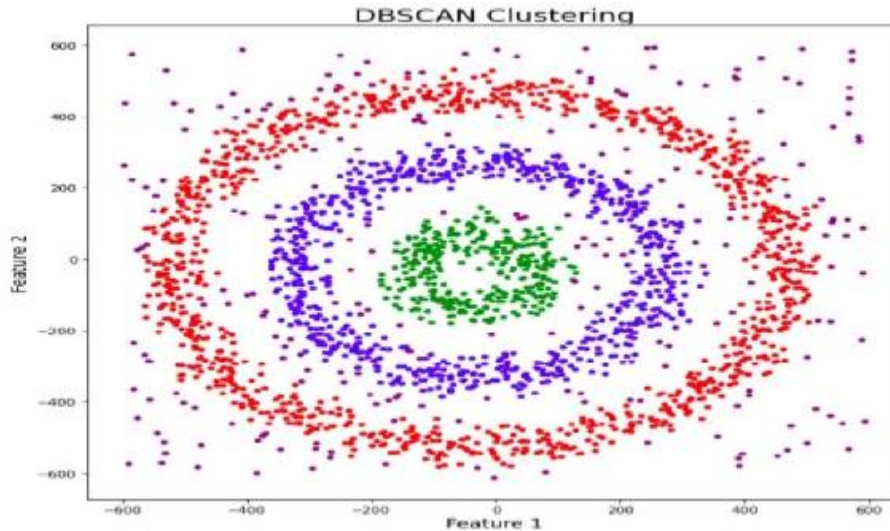


Figure 5 DBSCAN Clustering

Deep Learning[13] :-

Deep learning models are widely used in extracting high-level.

abstract features, providing improved performance over the traditional models, increasing interpretability and also for understanding and processing plants data . To predict splicing action of exons, a fully connected feedforward neural network was designed by Xiong et al. In recent years, CNNs they are applied on the crop dataset directly without the requirement of defining features a priori. Compared to a fully connected network, CNNs use less parameters by applying a convolution operation on the input data space and also parameters are shared between the regions[24].

Deep Learning Using convolutional neural networks (CNN)[12]:-

In several of our previous examples, they have already come up against image data, which consist of pixels arranged in a 2D grid. Depending on whether they are looking at a black and white or color image, they might have either one or multiple numerical values corresponding to each pixel location. Until now, they have dealt with this rich structure in the least satisfying possible way. They simply threw away this spatial structure by

flattening each image into a 1D vector and fed it into a fully-connected network. These networks are invariant to the order of their inputs [22].

They will get qualitatively identical results out of a multilayer perceptron whether they preserve the original order of our features or if they permute the columns of our design matrix before learning the parameters. Ideally, they would find a way to leverage our prior knowledge that nearby pixels are more related to each other.

In this chapter, they introduce convolutional neural networks (CNNs), a powerful family of neural networks that are designed for precisely this purpose. CNN based network architectures now dominate the field of computer vision to such an extent that hardly anyone these days would develop a commercial application or enter a competition related to image recognition, object detection, or semantic segmentation, without basing their approach on them.

Modern ‘convnets’, as they are often called other their design to inspirations from biology, group theory, and a healthy dose of experimental tinkering. In addition to their strong predictive performance, convolutional neural networks tend to be computationally efficient, both because they tend to require further parameters than dense architectures and because convolutions are easy to parallelize across GPU cores. As a result, researchers have sought to apply convnets whenever possible, and

have emerged as credible competitors even on tasks with 1D sequence structure, such as audio, text, and time series analysis . Some clever adaptations of CNNs have also brought them to bear on graph structured data and in recommender systems [24]

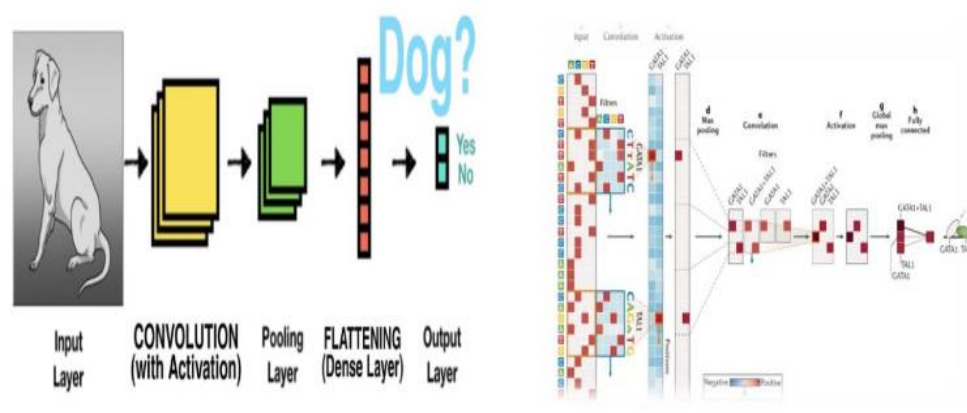


Figure 6 convolutional neural networks (CNN)

Similar to feedforward networks, but they're usually utilized for image recognition, pattern recognition, and/or computer vision. These networks harness principles from linear algebra, particularly matrix multiplication, to identify patterns within an image.

Is ResNet50 a CNN?

CNNs are particularly adept at handling image-related tasks by leveraging convolutional layers that recognize patterns and hierarchies within the data. Therefore, since ResNet was trained on ImageNet (a large-scale dataset containing millions of labeled images across thousands of categories) and also incorporates convolutional layers, it is a CNN, allowing us to handle various visual recognition tasks.

ResNet-50 Model

ResNet stands for Residual Network and is a specific type of convolutional neural network (CNN) introduced in the 2015 paper "Deep Residual Learning for Image Recognition" by He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. CNNs are commonly used to power computer vision applications.

ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). Residual neural networks are a type of artificial neural network (ANN) that forms networks by stacking residual blocks.

ResNet-50 is a pretrained Deep Learning model for image classification of the Convolutional Neural Network(CNN, or ConvNet), which is a class of deep neural networks, most commonly applied to analyzing visual imagery. ResNet-50 is 50 layers deep and is trained on a million images of 1000 categories from the ImageNet database. Furthermore the model has over 23 million trainable parameters, which indicates a deep architecture that makes it better for image recognition.

Using a pretrained model is a highly effective approach, compared if you need to build it from scratch, where you need to collect great amounts of data and train it yourself. Of course, there are other pretrained deep models to use such as AlexNet, GoogleNet or VGG19, but the ResNet-50 is noted for excellent generalization performance with fewer error rates on recognition tasks and is therefore a useful tool to know.

ResNet stands for Residual Network and more specifically it is of a Residual Neural Network architecture. What characterizes a residual network is its identity connections. Identity connections takes the input directly to the end of each residual block, as shown below with the curved arrow:

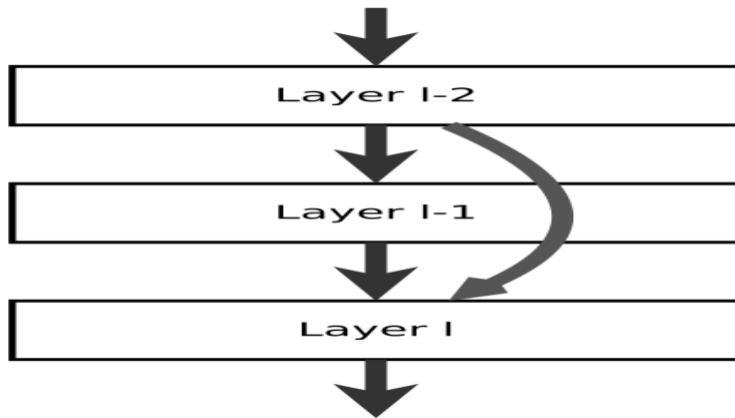


Figure 7 ResNet-50

Specifically, the ResNet50 model consists of 5 stages each with a residual block. Each residual block has 3 layers with both 1×1 and 3×3 convolutions. The concept of residual blocks is quite simple. In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away, called identity connections.

In this article, we will train a classification model which uses the feature extraction + classification principle, i.e., firstly, we extract relevant features from an image and then use these feature vectors in machine learning classifiers to perform the final classification.

Steps:

- **Input Layer:**
 - The journey begins with the input layer, where the raw image data is fed into the network.
- **Convolutional Layers:**
 - Convolutional layers process the input image to detect features and generate feature maps. Convolutional neural networks, has to do with the convolution between a kernel (or a filter) and an image in each convolutional layer.

A filter refers to a small matrix, and the convolution operator (denoted as $**$, sometimes called cross-correlation) gives rise to a new image where each element is the highest combination of the entries of a region or patch of the original image (also called local receptive field, with the highest given by the filter). In other words, a convolution is a dot product of 2 flattened matrices: a kernel and a patch of an image of the same size.

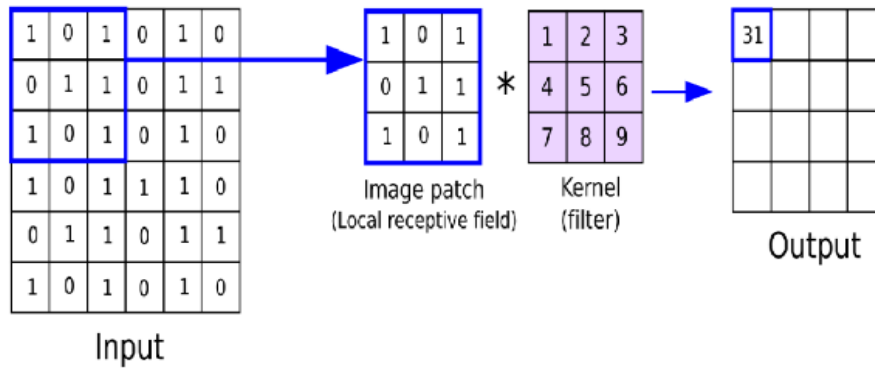


Figure 8 Convolutions in CNN

- **Max Pooling:**
 - Max pooling layers follow these convolutional layers. They down sample the spatial dimensions of the feature maps, retaining important information and reducing computational load.
- **Convolution and Residual Blocks:**
 - Next are the convolutional blocks made up of multiple convolutional layers followed by residual blocks.
- **More Convolutional Blocks:**
 - More convolutional blocks follow the residual blocks, refining features and learning intricate patterns.
- **Global Average Pooling:**
 - Instead of fully connected layers, ResNet typically employs global average pooling. Global average pooling reduces the spatial dimensions of the feature maps to a single value per feature, simplifying the architecture.
- **Output Layer:**
 - The reduced feature maps undergo a final classification step in the output layer, producing the model's predictions.

Importance of adding GlobalAveragePooling2D layer3:-

The GlobalAveragePooling2D layer in a convolutional neural network (CNN) computes the average value of each feature map in the previous layer, resulting in a single value for each feature map. This process reduces the spatial dimensions of the feature maps to a fixed size, effectively collapsing the spatial information and retaining only the channel-wise (feature) information.

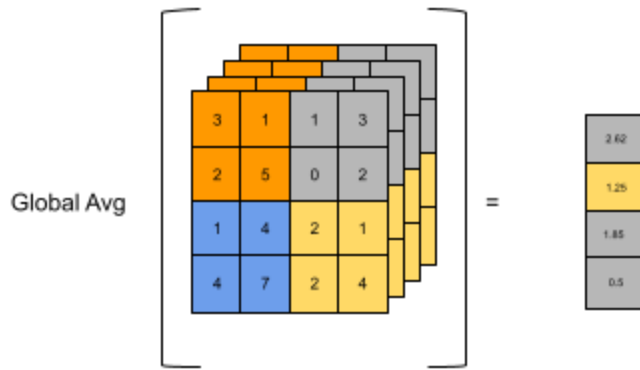


Figure 9 Working of GlobalAveragePooling2D¹

As demonstrated in the previous code, the GlobalAveragePooling2D layer is employed after the base ResNet50 model in the context of the fashion recommender system. It helps transform the complex feature maps generated by the convolutional layers into a compact and informative feature vector that can be further used for similarity calculations and recommendation purposes.

Here's how the GlobalAveragePooling2D layer works:

1. **Input:**

The input to the GlobalAveragePooling2D layer is a set of feature maps (also known as activation maps) generated by the previous convolutional layers in the network. Each feature map represents the presence of certain features in different regions of the input image.

2. **Pooling Operation:**

For each feature map, the GlobalAveragePooling2D layer computes the average value of all the elements within that feature map. This operation effectively summarizes the feature map by capturing the average response of that particular feature across the entire spatial extent of the input image.

3. **Reduced Spatial Dimensions:**

Unlike traditional max pooling or average pooling layers, which operate on local patches of the feature map, the GlobalAveragePooling2D layer computes a single average value for the entire feature map. This results in a drastic reduction in spatial dimensions.

4. **Flattening:**

The computed average values from all the feature maps are typically flattened into a 1D vector. This vector serves as a compact representation of the input image's features.

Overfitting [35] :-

Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points. It is the result of an overly complex model with an excessive number of training points. A model that is overfitted is inaccurate because the model has effectively memorized existing data points.

The trend line of an overfitted model reflects the errors in the data that it is trained with, instead of accurately predicting unseen data.

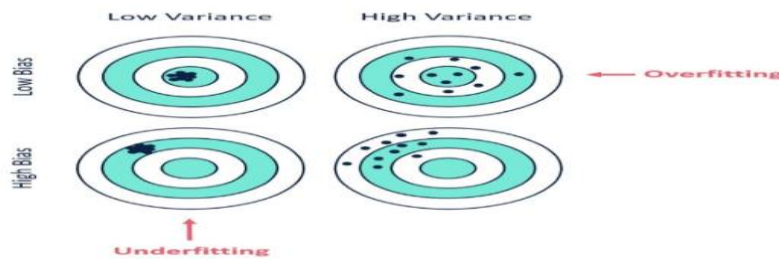


Figure 10 Overfitting

Time per inference step is the average of 30 batches and 10 repetitions. To Address the issue of overfitting in CNN [38] (Convolutional Neural Networks), you can follow the following steps:

- **Increase the size of the training dataset:**
 - Overfitting can occur when there is a lack of training data. You can collect more data if possible or use data augmentation techniques to generate additional data from existing samples.
- **Split the data into training, testing, and test sets:**
 - Divide your dataset into separate training, testing, and test sets. Use the testing set to evaluate the model's performance during training and test it on the test set. You can employ techniques like cross-testing to assess performance on different subsets.
- **Apply regularization techniques:**

- Use regularization techniques to reduce overfitting. You can employ techniques like L1 or L2 regularization to reduce the magnitude of the highest and prevent excessive growth of the model. Techniques such as Dropout can also be utilized to avoid over-reliance on specific subsets of parameters.
- **Reduce the model's complexity:**
 - Overfitting can occur if the model is too complex. Try reducing the number of layers or the width of the layers in the network. Techniques like pruning can also be used to reduce the number of parameters in the model.
- **Monitor the learning rate:**
 - Keep track of the learning rate during training. A high learning rate can lead to overshooting and instability, while a low learning rate can slow down convergence. Experiment with different learning rates to find an optimal value.

transfer learning Model :-

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data. Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data.

training model:-

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output [27].

The result from this correlation is used to modify the model. Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

[21] Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of the highest and biases that have low loss, on average, across all examples.

Example: -

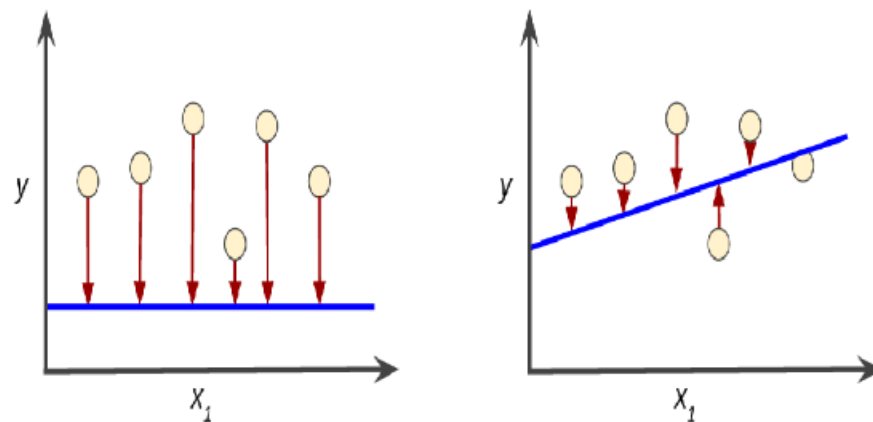


Figure 11 Example training model

Notice [21]

1. High loss in the left model; low loss in the right model.
2. the arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the line in the right plot is a much better predictive model than the line in the left plot.

There are 7 primary steps involved in creating a machine learning model [27]: -

1. Data Gathering
2. Preparing Data
3. Choosing a Model
4. Training
5. Testing & Testing
6. Hyper Parameter
7. Prediction

Then a stage begins Deployment model

Data Gathering:-

It is the most important step in solving any supervised machine learning problem. Your text classifier can only be as good as the dataset it is built from. Gathering data in

deep learning refers to the process of collecting and preparing a large set of training examples for a deep learning model.

Deep learning algorithms require vast amounts of data to train effectively and learn complex patterns and relationships in the data. The most common data sources to collect data for a DL model:

Open-Source Datasets:-

They are publicly available datasets that can be used for training and evaluating deep learning models. These datasets are often curated, pre-processed, and annotated to facilitate their use in machine learning applications .

The best sources for public datasets are:

- Kaggle (That they are used in our Model)
- Amazon
- UCI Machine Learning Repository
- Google's Datasets Search Engine
- Microsoft

Open-source datasets provide a valuable resource for deep learning researchers and practitioners, allowing them to train and evaluate models on standardized datasets and compare their results with other research teams. They also help to reduce the cost and time required for data collection and annotation, particularly for small research teams or individuals without access to large amounts of data.

They Are scraping:-

It refers to the process of automatically extracting data from they sites using deep learning techniques. This involves using machine learning models to learn how to identify and extract relevant information from the pages, such as text, images, or links .

Here is a short list of free tools for they-scraping :

- Scrapy
- Pro They Scraper
- Scraper API

Synthetic Datasets:-

It refers to artificially generated datasets that are designed to simulate real-world scenarios for training and evaluating machine learning models. These datasets are created using computer graphics techniques or other simulation methods and can be used to supplement or replace real-world datasets, particularly in cases where collecting real-world data is impractical or expensive.

Manual Data Generation:-

The last possibility to collect data is just that: collect data by yourself. This technique is very similar to synthetic datasets with the exception that it contains real data, and you need to generate the data manually instead of automatically.

Preparing Data:-

They are crucial steps in the process of building a deep learning model. These steps involve transforming raw data into a format that can be used for training and removing any noise or errors in the data.

By preparing the data and cleaning it properly, they can ensure that the deep learning model has a high-quality dataset to learn from and can make accurate predictions on new, unseen data.

Choosing a Model:-

It refers to the process of selecting an appropriate ResNet50 as the model architecture for your task. You can use pre-trained weights from models trained on large-scale datasets like ImageNet or train the model from scratch if you have a sufficiently large and representative dataset. That is best suited for the given task. The choice of the model architecture can significantly affect the performance of the deep learning system.

In deep learning, there are various types of neural network architectures that can be used, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers. Each architecture has its strengths, and they are suited for different types of tasks, and they are here choosing ResNet50 model.

Training:-

Training in deep learning refers to the process of adjusting the parameters of a neural network model to optimize its performance on a given task. This is achieved by exposing the model to a large set of labeled training data and iteratively adjusting the weights and biases of the model to minimize the error between the predicted output and the actual output.

During the training process, the model learns to identify patterns and relationships in the training data that can be used to make accurate predictions on new, unseen data. The

performance of the model is evaluated on a separate testing set during the training process to ensure that it is not overfitting to the training data.

Once the training process is complete, the model can be used to make predictions on new data.

This step is where you start to know what your problem is, and you will see how you will collect the data or data that you will work on training, and you will collect them through any tools and there are many data sources to collect data for the DL model.

Testing & Testing:-

Testing is the process of evaluating the performance of the trained model on a separate dataset that is not used during the training process. This is done to ensure that the model does not overfit to the training data and can generalize they'll to new data. The testing set is typically used to tune the hyperparameters of the model, such as the learning rate and number of epochs, and to select the best model architecture.

Testing, on the other hand, is the process of evaluating the performance of the final model on a completely new dataset that has not been seen by the model during the training or testing process. This provides an estimate of the model's performance on real-world data and is a crucial step in ensuring that the model can be deployed for practical use.

Hyper Parameter Tuning:-

Hyperparameter tuning in deep learning refers to the process of selecting the optimal hyperparameters for a neural network model to achieve the best performance on a given task.

Prediction:-

Once the model is trained and tested, it can be used for prediction on new, unseen images of hand gestures. The ResNet50 model takes the input image, performs a series of Input Layer, Convolutional Layers, Max Pooling, Convolution and Residual Blocks, More Convolutional Blocks, Global Average Pooling, Output Layer.

Pre-requisites and Dataset:-

A pretrained model is a deep learning model someone else built that's trained on large datasets to accomplish a specific task and solve some problem, and it can be used as is or customized to suit application requirements across multiple industries. Instead of building a model from scratch, developers can use pretrained models and customize them to meet their requirements.

For example, you could re-purpose a deep learning model built to identify dog breeds to classify dogs and cats, instead of building your own. This could save you the pain of finding an effective neural network architecture, the time you spend on training, the trouble of building a large corpus of training data and guarantee good results.

You could spend ages coming up with a fifty layered CNN for perfectly differentiating your cats from your dogs or you could simply re-purpose one of the many pre-trained image classification models available online. There are mainly three different ways in which a pre-trained model can be re-purposed [39]. They are.

1-Extracting Features

This mechanism is called a fixed feature extraction. They re-train only the new output layer they added and retain the heights of every other layer.

2-Copy the architecture of a pre-trained network

They follow this approach when they have a large corpus of data to train on, but it isn't very similar to the data the pre-trained model was trained on.

3-Freeze some layers and train the others

This approach is adopted when our dataset is small, and the data similarity is also low. The other layers focus on the most basic information that can be extracted from the data and hence this could be used as it is on another problem, as often the basic level information would be the same.

Disadvantages: -

[40] Many pre-trained models are trained for less or more different purposes, so may not be suitable in some cases. It will take large number of resources (time and computation power) to train big models from scratch.

ResNet 50:-

ResNet50 is a specific variant of the ResNet architecture, belonging to the ResNet family of convolutional neural networks. It is named "ResNet50" because it consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers.

ResNet50 was introduced as part of the ResNet architecture by Kaiming He et al. in their seminal paper "Deep Residual Learning for Image Recognition" in 2015. It was designed to address the challenges of training very deep neural networks by introducing residual connections, which allow gradients to flow more easily during training.

Key features of ResNet50 include:

1. **Residual Blocks:**

ResNet50 is composed of residual blocks, each containing multiple convolutional layers. These blocks utilize skip connections (or shortcut connections) to bypass one or more layers, allowing the network to learn residual mappings.

2. **Bottleneck Layers:**

ResNet50, like other variants in the ResNet family, uses bottleneck layers in its residual blocks. These bottleneck layers consist of a sequence of 1x1, 3x3, and 1x1 convolutions, which help reduce the computational cost while preserving representational power.

3. **Global Average Pooling:**

Instead of using traditional fully connected layers at the end of the network, ResNet50 typically ends with global average pooling followed by a softmax activation function for classification.

ResNet50 has become a widely used architecture for various computer vision tasks, particularly image classification on datasets like ImageNet. It strikes a balance between model depth and computational efficiency, making it suitable for both research and practical applications. Additionally, ResNet50 serves as the basis for deeper variants like ResNet101 and ResNet152, which have even more layers and increased representational power.

Advantages of ResNet50:-

ResNet50 offers several advantages over earlier architectures and even over some contemporary ones, making it a popular choice for various computer vision tasks:

1) **Deeper Architecture:**

ResNet50 is relatively deeper compared to earlier convolutional neural network (CNN) architectures like VGG or AlexNet. This depth enables the model to learn more complex features and representations, leading to improved performance on challenging tasks.

2) **Residual Connections:**

ResNet50 introduces residual connections, which address the vanishing gradient problem encountered in very deep networks. These shortcut connections allow gradients to flow more easily during training, facilitating the training of deeper models without suffering from degradation in performance.

3) **Higher Accuracy:**

Due to its deeper architecture and residual connections, ResNet50 often achieves higher accuracy compared to shallower architectures on various image classification benchmarks, such as ImageNet.

4) **Parameter Efficiency:**

ResNet50 achieves its depth with relatively fewer parameters compared to earlier architectures, thanks to the use of bottleneck layers in residual blocks. This parameter efficiency allows ResNet50 to achieve higher accuracy without significantly increasing computational cost or memory requirements.

5) **Transfer Learning:**

Pre-trained ResNet50 models trained on large datasets like ImageNet are readily available. These pre-trained models can be fine-tuned on smaller datasets or used as

feature extractors in transfer learning scenarios, where they have demonstrated excellent performance across a wide range of tasks.

6) **Generalization:**

ResNet50's depth and architecture enable it to generalize well to diverse datasets and tasks beyond image classification. It has been successfully applied to tasks such as object detection, semantic segmentation, and image captioning with minimal modifications.

7) **Community Support and Availability:**

ResNet50 is a well-studied architecture with extensive documentation, implementations in popular deep learning frameworks like TensorFlow and PyTorch, and a large community of researchers and practitioners actively working with and improving upon it.

Overall, ResNet50's combination of depth, residual connections, parameter efficiency, and widespread adoption make it a powerful and versatile architecture for various computer vision tasks, making it a preferred choice for many deep learning practitioners

ResNet50, ResNet101, and ResNet152:-

	ResNet-50	ResNet-101	ResNet-152
Introduction	<p>This function returns a Keras Image classification model, optionally loaded with the weights pre-trained on ImageNet.[1]</p> <p>ResNet50 is a variant of the ResNet architecture introduced by Kaiming He et al. in their 2015 paper "Deep Residual Learning for Image Recognition."</p> <p>It consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers.</p> <p>ResNet50 uses residual connections (skip connections) to enable the training of very deep neural networks by addressing the vanishing gradient problem.</p> <p>This architecture strikes a balance between depth and computational efficiency, making it a popular choice for various computer vision tasks.</p> <p>ResNet50 typically offers high accuracy on image classification benchmarks while being more computationally efficient compared to deeper variants like ResNet101 and ResNet152.</p>	<p>it uses inverted residual blocks with bottlenecking features.</p> <p>ResNet101 is another variant of the ResNet architecture, which includes 101 layers.</p> <p>It is deeper than ResNet50, incorporating additional residual blocks and layers to learn more complex features and representations.</p> <p>ResNet101 often achieves slightly higher accuracy compared to ResNet50, especially on challenging datasets and tasks.</p> <p>However, the increased depth comes at the cost of higher computational requirements and longer training times.</p>	<p>Each Keras Application expects a specific kind of input preprocessing.</p> <p>ResNet152 is the deepest variant among the three, consisting of 152 layers.</p> <p>It includes even more residual blocks and layers compared to ResNet101, allowing it to capture even more intricate patterns in the data.</p> <p>ResNet152 tends to offer slightly higher accuracy compared to both ResNet50 and ResNet101, especially on tasks where fine-grained distinctions between classes are crucial.</p> <p>However, training and deploying ResNet152 require significantly more computational resources and memory compared to shallower architectures.</p>

Argument	<ul style="list-style-type: none"> • input_shape • include_top • weights • input_tensor • pooling • classes • classifier_activation • kwargs 	<ul style="list-style-type: none"> • input_shape • include_top • weights • input_tensor • pooling • classes • classifier_activation • kwargs 	<ul style="list-style-type: none"> • input_shape • include_top • weights • input_tensor • pooling • Classes • classifier_activation • kwargs
----------	--	--	--

Datasets

[ImageNet](#) is a dataset of millions of labeled high-resolution images belonging roughly to 22k categories. The images were collected from the internet and labeled by humans using a crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2013) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. There are approximately 1.2 million training images, 50k validation, and 150k testing images.

The [PASCAL VOC](#) provides standardized image data sets for object class recognition. It also provides a standard set of tools for accessing the data sets and annotations, enables evaluation and comparison of different methods and ran challenges evaluating performance on object class recognition.

[7] It might go without saying that you cannot do data science without data. They could lose hundreds of pages pondering the precise nature of data but for now they'll err on the practical side and focus on the key properties to be concerned with. Generally, they are concerned with a collection of examples (also called data points, samples, or instances). To work with data usefully, they typically need to come up with a suitable numerical representation. Each example typically consists of a collection of numerical attributes called features or covariates.

If they're working with image data, each individual photograph might constitute an example, each represented by an ordered list of numerical values corresponding to the brightness of each pixel. A 224×224 color photograph [4].

survive, given a standard set of features such as age, vital signs, diagnoses, etc. When every example is characterized by the same number of numerical values, they say that the data consists of fixed-length vectors, and they describe the (constant) length of the vectors as the dimensionality of the data. As you might imagine, fixed length can be a

convenient property. If they wanted to train a model to know the letter (alphabet letters) or word [6].

However, not all data can easily be represented as fixed length vectors. While they might expect images to come from standard equipment, they can't expect images mined from the internet to all show up in the same size. Images to a standard size, text data resists fixed-length representations even more stubbornly.

One major advantage of deep learning over traditional methods is the comparative grace with which modern models can handle varying-length data. Generally, the more data they have, the easier our job becomes. When they have more data, they can train more powerful models, and rely less heavily on pre-conceived assumptions. The regime changes from (comparatively small) to big data [5] is a major contributor to the success of modern deep learning. To drive the point home, many of the most exciting models in deep learning don't work without large data sets. Some others work in the low-data regime, but no better than traditional approaches.

Finally, it's not enough to have lots of data and to process it cleverly. They need the right data. If the data is full of mistakes, or if the chosen features are not predictive of the target quantity of interest, learning is going to fail. Moreover, poor predictive performance isn't the only potential consequence. In sensitive applications of machine learning, like predictive policing, résumé screening, and risk models used for lending, they must be especially alert to the consequences of garbage data.

One common failure mode occurs in datasets where some groups of people are unrepresented in the training data. Failure can also occur when the data doesn't merely under-represent some groups but reflects societal prejudices. For example, if past hiring decisions are used to train a predictive model that will be used to screen resumes, then machine learning models could inadvertently capture and automate historical injustices. Note that this can all happen without the data scientist being complicit, or even aware[8].

Taxonomies:-

Most machine learning involves transforming the data in some sense. They might want to build a system that ingests photos. By model, they denote the computational machinery for ingesting data of different type A collection of photos (ASL), and spitting out predictions of a possible different type (letter of the alphabet , words , words or sentence) .

The models for this project are classified according to the following steps :-

- Supervised Learning
- Classification
- multiclass classification

Chapter 3

LITERATURE REVIEW

Abstract:

A literature review is a comprehensive summary of previous research on a topic. The literature review surveys scholarly articles, books, and other sources relevant to a particular

area of research. In this chapter, we will delve into some topics related to sign language recognition for disabled in the past years.

N O	Paper	Year	Steps	Algorithms	Accuracy	Dataset
1	Enhancing Ship Classification with CNNs and Transfer Learning	2024	1-Prepare the data that you will use 2-Data processing (Cleaning Data) 3-Obtain the pre-trained model 4-create a base model 5-make layers 6-trained this layer 7-check accuracy and improve model	CNN TensorFlow Lite, OpenCV	74%	The five ship types—Cargo, Tanker, Military, Carrier, and Cruise—are represented in the dataset by a group of photos taken by survey boats. The dataset offers a wide variety of visual data for model construction, with 6252 images for training and 2680 images for testing.
2	Image Prediction Using a Pre-trained Model	2022	1-Import Necessary Libraries 2-Load Pre-trained Model 3-Load and Preprocess Image 4-Perform Prediction 5-Interpret Predictions	VGG	89.8%	The pre-trained models included with Keras were developed using the smaller data set utilized for this competition. Images of 1,000 various types of objects, including food and animal breeds, are included in the data set. For instance, a Granny Smith apple is one of the object types in the data set. More than 1200 images of just this type of apple are included in the data collection.
3	An Evaluation of Machine Learning Models For Deep Learning Image Classification With Fashion-MNIST Dataset	2024	1-Import Necessary Libraries 2-Load and Preprocess the data 3-Define Model Architecture 4-Compile the Model 5-Train the Model 6-Evaluate the Model	Feed-forward NN, CNN, Random Forest, XGBoost	85%, 90%, 78.6%, 88%	This research uses the Fashion-MNIST dataset [2] to assess and contrast several machine learning and deep learning models for image classification. The Fashion-MNIST dataset is a benchmark dataset that contains 70,000 grayscale images of fashion items from 10 different classes, such as T-shirts, dresses, sneakers, etc.

4	Image anomalies detection using transfer learning of ResNet-50 convolutional neural network	2022	1-Import Necessary Libraries 2-load Pre-trained ResNet-50 Model 3-Add Custom Layers for Anomaly Detection 4-Compile the Model 5-Load and Preprocess Image Data 6-Train the Model 7-Evaluate the Model	ResNet-50, CNN, SVM	95%	<p>We assess the implementation of the planned strategy on the overall normal picture irregularity site dataset: a dataset consisting of three types of industrial elements namely: bottle ,carton and spoon.</p> <p>The total number of atypical images is 127, where the pictures were taken locally by Apple's versatile 12-megapixel camera. The image size is 4032*3024 with a depth of 24 bit, the image type is jpg and the images are RGB color.</p>
5	Fashion Recommendation System Using Resnet50	2023	1-Data Collection and Preprocessing 2-Load Pre-trained ResNet50 Model 3-Extract Features 4-Build Recommendation Engine 5-Train and Fine-tune Recommendation Model 6-Evaluation 7-Deployment 8-Monitoring and Maintenance	ResNet-50, CNN TensorFlow Lite, OpenCV	98%	<p>We trained our model using a large image dataset, including the multiple category labels, descriptions and high-res images of fashion products, which consists of 44k+ images.</p> <p>The results have been promising, indicating potential real world applications such as searching for a product using its digital copy in large sets of images.</p>
6	A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation	2023	1-Data Collection 2-Data Preprocessing 3-Taxonomy Development Data Analysis	ResNet-50, CNN TensorFlow Lite, OpenCV	86%	<p>The dataset that is generated or analysed during this study are included in the published article</p>

7	Fashion image classification on mobile phones using layered deep convolutional neural networks	2016	1-preparing the data 2-defines the CNN architecture 3-compiles the model 4-trains the model 5-evaluates the trained model	Layered Deep Convolutional Neural Networks (LDCNNs)	96.9%	we collected two kind of dataset: (I) 120K datasets of clothes images on EC sites to train the classifier and (II) the clothing image set composed of photos taken by participants with their mobile phones.
8	Research on Clothing Image Classification Models Based on CNN and Transfer Learning	2021	1-Data Preparation 2-Define CNN architectures 3-Model Training 4-Model Evaluation 5-Experimentation 6-Analysis 7-Visualization 8-Conclusion	CNN Inception-v3 and pre-trained EfficientNet-B0	97.2% and 97.8%	These CNN models are trained on the dataset named Street-FashionData having eight categories of clothing images, and their performances are evaluated and compared by conducting classification tests.
9	A comparative research on clothing images classification based on neural network models	2021	1-Data Preparation 2-Define multiple neural network architectures 3-Model Training 4-Model Evaluation 5-Experimentation 6-Analysis 7-Visualization 8-Conclusion	CNN, MobileNet V1, and MobileNet V2	93.3% and 91.2%	the classification of clothing images in the Fashion-MNIST dataset and the results are compared.
10	Hierarchical convolutional neural networks for fashion image classification	2018	1-Data Preparation 2-Define CNN architectures 3-Model Training 4-Model Evaluation 5-Experimentation 6-Analysis 7-Visualization 8-Conclusion	HCNN VGG16 and VGG19, SVM	99.9% and 92.8%	In this paper, we use Fashion-MNIST dataset, which consists of 50,000 images of training set and 10,000 images of test set (Xiao, Rasul, & Vollgraf, 2017). Each grayscale image is sized 28×28 pixels and classified into 10 classes including 't-shirt', 'trouser', 'pullover', 'dress', 'coat', 'sandals', 'bag', 'shirt', 'sneaker', and 'ankle boots'

3.1. Paper 1

Enhancing Ship Classification with CNNs and Transfer Learning

Introduction

Welcome to an in-depth exploration of ship classification using Convolutional Neural Networks (CNNs) with the Analytics Vidhya hackathon dataset. CNNs are a cornerstone of image-related tasks, known for their ability to learn hierarchical representations of images. In this project, we dive into understanding the power of CNNs to classify ships based on their visual features.

This project aims to demonstrate deep learning application in image categorization and compare CNNs built from scratch and those enhanced through transfer learning. It explores ship classification, from preprocessing to evaluation and comparison.

3.2. Paper 2

Image Prediction Using a Pre-trained Model

Introduction

Researchers from all over the world compete to create the most precise and effective picture recognition systems. Therefore, it often makes sense to employ an existing neural network design as a starting point for your own projects rather than having them bend their own neural network designs from begin. Even better, researchers shared the trained neural network versions of these network architectures after training them on sizable data sets. Therefore, we can use such already-trained neural networks either directly or as a jumping point for our training.

3.3. Paper 3

An Evaluation of Machine Learning Models For Deep Learning Image Classification With Fashion-MNIST Dataset

Introduction

This project uses the Fashion-MNIST dataset to assess and compare various machine-learning models and classifiers for image classification. The objective is to compare different models' performance and identify their shortcomings. By analyzing the data, this project illuminates the best methods for achieving outstanding accuracy in image classification.

Accurate image categorization is essential in many applications; Consequently, computer vision academics, professionals, and developers will benefit from this project's findings. When it comes to medical diagnosis, for instance, good image classification is essential for helping healthcare professionals make correct diagnoses quickly. Facial recognition systems use it for both access control and surveillance.

It is hoped that evaluating several machine learning models presented in this project would aid developers in selecting an appropriate model for their specific use cases. This can help developers save time and money by allowing them to quickly focus on the best model without having to try out many different ones. Overall, this project's findings aid in creating more precise and effective image classification systems with wide-ranging applications.

3.4. Paper 4

Image anomalies detection using transfer learning of ResNet-50 convolutional neural network

Introduction

With the quick advancement of keen fabricating, information-based blame determination has pulled in expanding attention. As one of the foremost prevalent strategies of diagnosing errors, deep learning has accomplished exceptional comes about. Be that as it may, due to the truth that the estimate of the seeded tests is little in diagnosing mistakes, the profundities of the deep learning (DL) models for fault conclusion are shallow compared to the convolutional neural network in other regions (including ImageNet), which limits the accuracy of the final prediction.

In this paper, ResNet-50 with a 25 convolutional layer depth has been proposed to diagnose anomalous images. Trained ResNet-50 applies ImageNet as a feature extractor to diagnose errors. It was proposed on three sets of data which are the bottle, the spoon, and the carton, and the proposed method was achieved. The prediction accuracy of the data set was 99%, 95% and 90%, respectively.

3.5. Paper 5

Fashion Recommendation System Using Resnet50

Introduction

Fashion recommendation systems are pivotal for enhancing online shopping using deep learning, specifically ResNet-50. Our Fashion Recommendation System (FRS) employs ResNet-50 to process and represent fashion images' key attributes like color and style. Content-based recommendations suggest visually similar items based on user preferences. The FRS evolves through machine learning, delivering improved user engagement and conversion rates in e-commerce fashion. This paper reviews and explores fashion recommendation systems and filtering techniques, filling a gap in academic literature.

It serves as a valuable resource for machine learning, computer vision, and fashion retailing professionals. We trained our model using a large image dataset, including the multiple category labels, descriptions and high-res images of fashion products, which consists of 44k+ images. The results have been promising, indicating potential real world applications such as searching for a product using its digital copy in large sets of images.

3.6. Paper 6

A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation

Introduction

E-commerce recommendation systems facilitate customers' purchase decision by recommending products or services of interest (e.g., Amazon). Designing a recommender system tailored toward an individual customer's need is crucial for retailers to increase revenue and retain customers' loyalty. As users' interests and preferences change with time, the time stamp of a user interaction (click, view or purchase event) is an important characteristic to learn sequential patterns from these user interactions and, hence, understand users' long- and short-term preferences to predict the next item(s) for recommendation.

This paper presents a taxonomy of sequential recommendation systems (SRecSys) with a focus on e-commerce product recommendation as an application and classifies SRecSys under three main categories as: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation (matrix factorization and Markov models) and (iii) neural network-based approaches (deep neural networks, advanced models).

This classification contributes towards enhancing the understanding of existing SRecSys in the literature with the application domain of e-commerce product recommendation and provides current status of the solutions available alongwith future

research directions. Furthermore, a classification of surveyed systems according to eight important key features supported by the techniques along with their limitations is also presented.

A comparative performance analysis of the presented SRecSys based on experiments performed on e-commerce data sets (Amazon and Online Retail) showed that integrating sequential purchase patterns into the recommendation process and modeling users' sequential behavior improves the quality of recommendations.

3.7. Paper 7

Fashion image classification on mobile phones using layered deep convolutional neural networks

Introduction

Toward implementation of fashion recommendation system based on photos taken with mobile phones, we propose a framework to recognize hierarchical categories of a fashion item. To classify an arbitrary photo of clothes robustly, (1) we collected two kind of dataset: (I) 120K datasets of clothes images on EC sites to train the classifier and (II) the clothing image set composed of photos taken by participants with their mobile phones.

(2) we proposed Layered Deep Convolutional Neural Networks (LDCNNs) which is specialized in classifying images into hierarchical categories: hoodie is a lower in hierarchy in tops category. Experimental result shows proposed LDCNNs obtained mean accuracy of 92.7% for datasets from EC sites and 96.9% for those from mobile phones. This results are better than those (84.9%, 90.6 % respectively) for MLR+CNN in classification accuracy.

3.8. Paper 8

Research on Clothing Image Classification Models Based on CNN and Transfer Learning

Introduction

Traditional clothing image classification method, which involves artificially extracting features of clothing images, are not effective owing to images having complex background and clothing distortion. With the purpose of improving the classification accuracy, eight models are designed based on convolutional neural network (CNN) and transfer learning. These CNN models are trained on the dataset named Street-FashionData having eight categories of clothing images, and their performances are evaluated and compared by conducting classification tests.

The experimental results show that model ICM-C and model ECM-C have high classification accuracies of 97.2% and 97.8%, respectively. These two models have the same convolutional bases as pre-trained Inception-v3 and pre-trained EfficientNet-B0, respectively, moreover, their entire networks need to be trained. The experimental results have proved the efficiency of classification models described in this paper.

3.9. Paper 9

A comparative research on clothing images classification based on neural network models

Introduction

To make deep learning really work in real life, it is better for the model to be used with neural networks of small memory requirements and small computations. In this paper, four neural network models, i.e. Fully Connected Neural Network, CNN, MobileNetV1, and MobileNetV2 are applied to deal with the classification of clothing images in the Fashion-MNIST dataset and the results are compared.

MobileNetV2, which performs best, is a deep learning network in the field of image classification and recognition. With the development of robots, smart auto, and augmented reality in real-world application, it is better to use a network such as MobileNet in a computationally limited platform.

The experimental results show that 1) MobileNet is a more effective method of classification of clothing images compared to Fully Connected Neural Network and CNN, with an accuracy rate of 91%. 2) MobileNetV2 has greater improvement in accuracy than the previous model, which accuracy has reached to 93%. 3) MobileNet is simpler than Fully Connected Neural Network and CNN model structures, although training time has increased, it ensures higher accuracy.

3.10. Paper 10

Hierarchical convolutional neural networks for fashion image classification

Introduction

Large amount of image data has been flooded with the help of search engines and social network services. Unstructured image data can now be used in statistics and data mining due to its large enough amount for training and the advanced computational power with multiple cores of CPU and the usage of GPU. In earlier studies, conventional machine learning methods such as Support Vector Machine (SVM), Principal Component Analysis

(PCA) and Linear Discriminant Analysis (LDA) have been used in image data analysis (Taigman, Yang, Ranzato, & Wolf, 2014).

However, these methods still have limitations when processing large-scaled image data. Deep learning, also known for Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), is introduced to solve the issue of lack of capacity to handle large-scaled and unstructured data (Tsantekidis et al., 2017). Deep learning methods have been widely used in estimation and classification problems such as stock price prediction, speech recognition, image captioning, face recognition systems, objects detection, age recognition, and adult content filtering (Iliukovich-Strakovskaia, Dral, & Dral ; Tsantekidis et al., 2017).

Regards to deep learning models applied on image data, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has contributed to introduce deeper CNN architectures with better results.

Chapter 4

IMPLEMENTATION:

1. Image pre-processing:

The aim of image processing is to enhance the quality of image and later on to perform features extraction and classification. It is most commonly used in computer vision, medical imaging, meteorology, astronomy, remote sensing and another related field. Tools used in image processing:

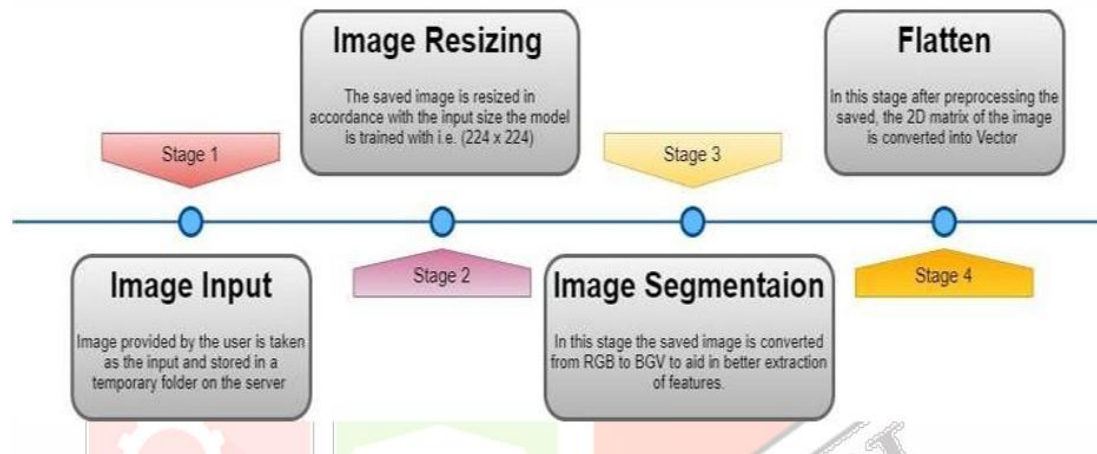


Fig. 12 Flow-chart for Image Preprocessing

Following tools have been used in project for image processing

1. **TensorFlow and Keras:** TensorFlow and its high-level API Keras are used for deep learning tasks, including loading a pre-trained ResNet-50 model, creating a neural network model, and making predictions.
2. **ResNet-50 :** The ResNet-50 architecture is employed for feature extraction from images. It's a pre-trained deep convolutional neural network (CNN) model commonly used for image classification and feature extraction.
3. **PIL (Pillow):** The Python Imaging Library (PIL), specifically the Pillow library, is used for image manipulation and loading images from files.
4. **NUMPY:** NumPy is used for numerical operations and data manipulation, including working with arrays and matrices.
5. **scikit-learn (sklearn):** scikit-learn is used for implementing the k-nearest neighbors (KNN) algorithm, which is used for finding similar fashion items based on image features.
6. **OpenCV (cv2):** OpenCV is used for image manipulation and display. It's used to show the recommended fashion items as images.

7. **TQDM:** TQDM is used for creating progress bars to track the progress of iterating through fashion images when extracting features.
8. **os:** The os module is used for file operations and directory handling. It's used to manage the storage and retrieval of image files.
9. **pickle:** The pickle library is used for serializing and deserializing Python objects. In the code, it's used to save and load image features and filenames.

4.1 RECOMMENDATION ENGINE:

Recommendation Engine may be treated as a black box which analyze some set of users and shows the items which a single user may like

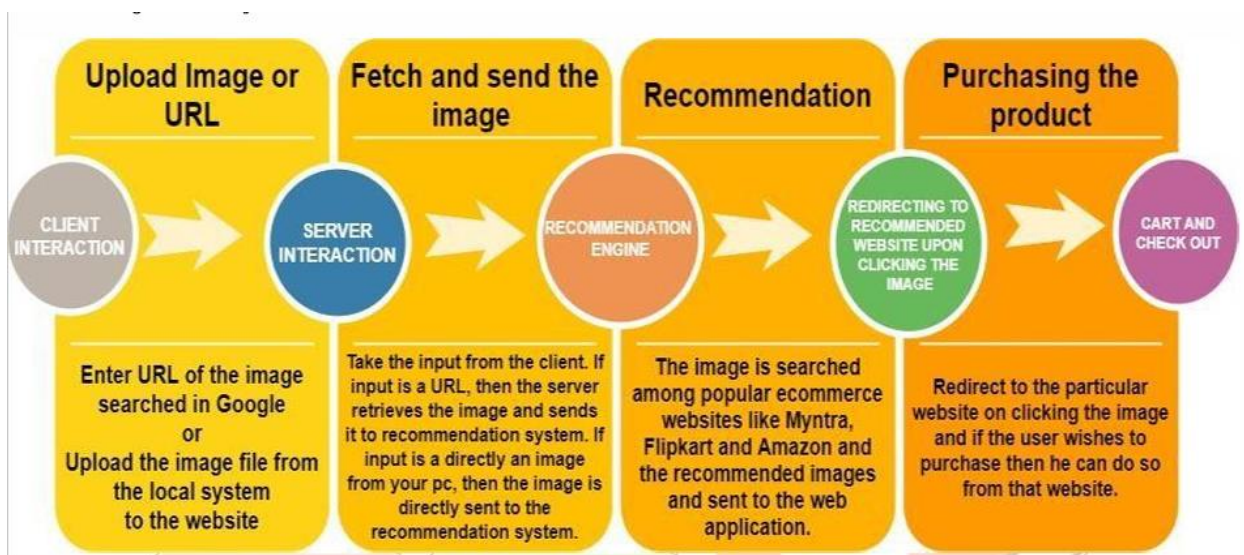


Fig. 13 Diagram for Recommendation Engine

The major benefits of using a recommendation engine are

1. Increased Sales
2. Cross-selling and Upselling
3. Reduced Abandoned Carts
4. Improved Customer Retention
5. Enhanced User Experience
6. Optimized Inventory Management
7. Dynamic Homepage and Landing Pages
8. Seasonal and Trend-Based Recommendations
9. Personalized Email Marketing
10. Customer Segmentation
11. Data-Driven Insights

12. Competitive Advantage

In this project, we use algorithms like nearest neighbours, pretraining the ResNet model and making a deployable streamlit app for the same.

The motivation behind this project is to work on an awesome ML use case we see all around us. Every time we open a shopping app and open a product, especially from the clothing section, we get a lot of similar recommendations to what we are looking.

If you are looking at a red and black striped shirt, you'll get suggestions of shirts with similar colours that are red and/or black and which match the pattern of stripes in this case. This subtle feature provides a good customer experience as customers can glance over similar products to make a better choice and get a closer shop-like experience. Here we build a primary fashion recommender to replicate the same.

Pre-requisites and Dataset

Throughout the tutorial, we will work with tools and libraries like streamlit, scikit-learn, tensorflow, PIL etc. So before proceeding with the project, we need to install these libraries and ensure the environment is set up.



Figure 14 Some samples from the dataset

We are using the “Fashion Product Images Dataset Small” dataset on Kaggle. It is a large dataset of fashion products, with over 44,000 images and multiple category labels, descriptions, and professional product images. This dataset is the smaller version of an existing high resolution dataset where images are scaled down to lower resolution making it easy to store and train for machines with limited resources. The dataset was scraped from

an e-commerce website and includes products from various categories, including dresses, tops, bottoms, shoes, and accessories.

The images are converted to low-resolution but they are professionally shot, making them ideal for faster training and evaluation for our purpose. The dataset includes a variety of product categories, making it a good choice for tasks that require a broad understanding of fashion. The comprehensive metadata provides information about the product brand, colour, size, and other details. Although we are currently not going to use metadata and only build using visual similarity for our recommendation system.

Concept of Feature Embeddings

In the world of machine learning and computer vision, feature embeddings play a pivotal role in transforming raw data into a more meaningful and compact representation. Feature embeddings are lower-dimensional vectors that capture essential characteristics of the data while reducing its dimensionality. In the context of image data, feature embeddings represent distinctive attributes of images, making them suitable for various tasks such as classification, similarity search, and recommendation.

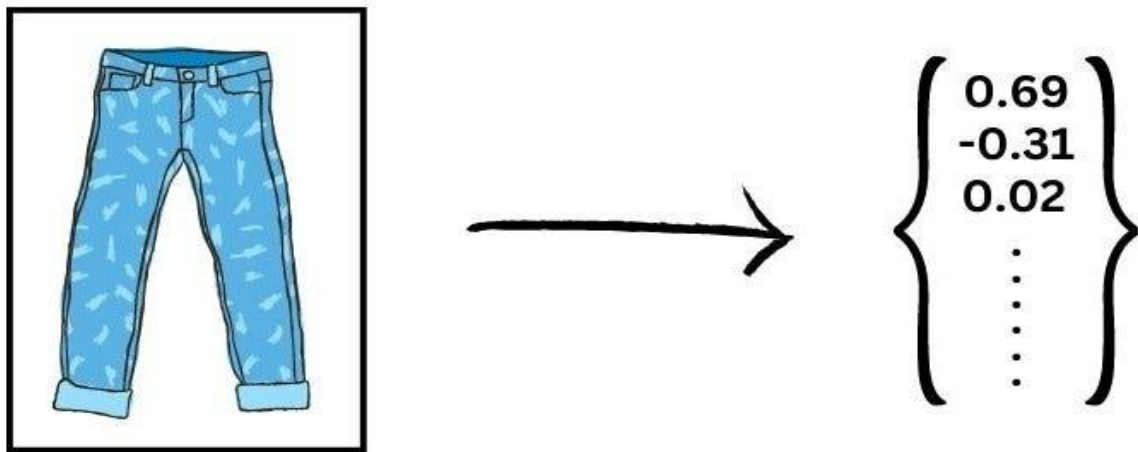


Figure 15 Representing image as a feature vector

Regarding fashion recommendations, the goal is to find similarities between fashion items based on their visual characteristics. We need a robust and discriminative representation of fashion images to achieve this. This is where pretraining using ResNet (Residual Neural Network) comes into play.

Why are we using pre-trained ResNet model?

ResNet is a deep neural network architecture that has revolutionized the field of computer vision. It introduced the concept of residual learning, which enables the training of much deeper networks without suffering from vanishing gradient problems. ResNet's deep architecture allows it to learn intricate features and patterns from images, making it an ideal candidate for extracting feature embeddings in our fashion recommender system.

Advantages of ResNet for Pretraining:

1. **Hierarchical Feature Learning:** ResNet's architecture enables the network to learn features hierarchically. This is essential for capturing both low-level details (e.g., edges, textures) and high-level semantic information (e.g., object shapes, patterns) from fashion images.
2. **Transfer Learning:** ResNet is pre-trained on massive image datasets, such as ImageNet, to recognize various objects and scenes. This pre-trained knowledge can be transferred to our fashion recommender system, allowing us to benefit from the network's understanding of complex visual features.
3. **Effective Feature Extraction:** ResNet's deep layers can extract rich and discriminative features, which is crucial for identifying subtle differences and similarities between fashion items. These features form the basis of our feature embeddings.
4. **Normalization and Regularization:** ResNet incorporates techniques like batch normalization and residual connections, leading to more stable and faster convergence during training. This helps in generating more reliable and consistent feature embeddings.

By leveraging the power of ResNet's pre-trained architecture, we can generate high-quality feature embeddings that encapsulate the visual essence of fashion items. These embeddings are a foundation for our Nearest Neighbours algorithm, enabling us to efficiently find and recommend similar fashion products based on their visual attributes.

Further on, we will delve deeper into the implementation of ResNet for feature extraction and demonstrate how these embeddings enhance the performance of our smart fashion recommender system.

Configuring the model

After understanding the motivation behind feature embedding and ResNet, let's get into implementing it using code. First, we start by defining a function for configuring the model, which we will use as a sequence of pre-trained ResNet50 model followed by a GlobalAveragePooling2D layer.

```
Product Recommendation > main.py > ...
1  import streamlit as st
2  import os
3  from PIL import Image
4  import numpy as np
5  import pickle
6  import tensorflow
7  from tensorflow.keras.preprocessing import image
8  from tensorflow.keras.layers import GlobalMaxPooling2D
9  from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
10 from sklearn.neighbors import NearestNeighbors
11 from numpy.linalg import norm
12
13 feature_list = np.array(pickle.load(open('embeddings.pkl', 'rb')))
14 filenames = pickle.load(open('filenames.pkl', 'rb'))
15
16 model = ResNet50(weights='imagenet', include_top=False, input_shape=(224,224,3))
17 model.trainable = False
18
19 model = tensorflow.keras.Sequential([
20     model,
21     GlobalMaxPooling2D()
22 ])
23
```

Figure 16 Configuring the model

We begin by importing the necessary libraries. TensorFlow and Keras provide the tools required for model creation and manipulation. The function is designed to provide us with the ResNet50-based model architecture. This model will be responsible for extracting feature embeddings from our fashion images.

We load the ResNet50 model with pre-trained weights from the ImageNet dataset. These weights enable the model to recognize a diverse range of objects and features, which is advantageous for our task. To prevent the pre-trained weights from being modified during our fashion recommender system training, we set the trainable attribute of the base ResNet50 model to False.

We create a sequential model by adding the base ResNet50 model to it. This establishes a flow for data to pass through the model's layers. We follow the base ResNet50 model with a GlobalAveragePooling2D layer. This layer averages the spatial dimensions of the feature maps, resulting in a global representation of the image's features. This compact representation is essential for generating meaningful feature embeddings.

Importance of adding GlobalAveragePooling2D layer

The GlobalAveragePooling2D layer in a convolutional neural network (CNN) computes the average value of each feature map in the previous layer, resulting in a single value for each feature map. This process reduces the spatial dimensions of the feature maps to a fixed size, effectively collapsing the spatial information and retaining only the channel-wise (feature) information.

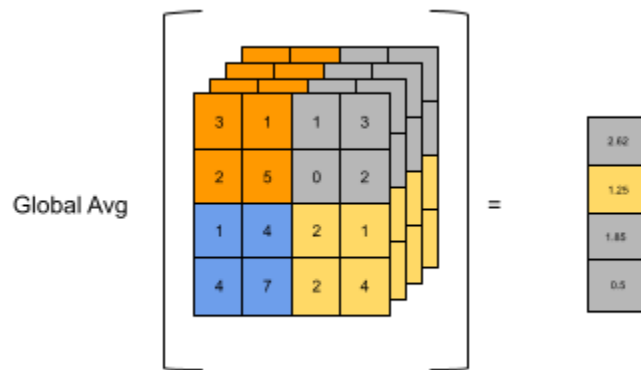


Figure 17 Working of GlobalAveragePooling2D¹

As demonstrated in the previous code, the GlobalAveragePooling2D layer is employed after the base ResNet50 model in the context of the fashion recommender system. It helps transform the complex feature maps generated by the convolutional layers into a compact and informative feature vector that can be further used for similarity calculations and recommendation purposes.

Here's how the GlobalAveragePooling2D layer works:

1. **Input:** The input to the GlobalAveragePooling2D layer is a set of feature maps (also known as activation maps) generated by the previous convolutional layers in the

network. Each feature map represents the presence of certain features in different regions of the input image.

2. **Pooling Operation:** For each feature map, the GlobalAveragePooling2D layer computes the average value of all the elements within that feature map. This operation effectively summarizes the feature map by capturing the average response of that particular feature across the entire spatial extent of the input image.
3. **Reduced Spatial Dimensions:** Unlike traditional max pooling or average pooling layers, which operate on local patches of the feature map, the GlobalAveragePooling2D layer computes a single average value for the entire feature map. This results in a drastic reduction in spatial dimensions.
4. **Flattening:** The computed average values from all the feature maps are typically flattened into a 1D vector. This vector serves as a compact representation of the input image's features.

Extracting Features

Now let's write a function for extracting features from our images and generate image embeddings. The `extract_features` function takes an image path and the pretrained ResNet model as inputs, and returns a normalized feature vector that represents the image.

```
34 def feature_extraction(img_path,model):
35     img = image.load_img(img_path, target_size=(224, 224))
36     img_array = image.img_to_array(img)
37     expanded_img_array = np.expand_dims(img_array, axis=0)
38     preprocessed_img = preprocess_input(expanded_img_array)
39     result = model.predict(preprocessed_img).flatten()
40     normalized_result = result / norm(result)
41
42     return normalized_result
```

figure 18 Extract features

After loading the necessary libraries, the function loads the image from the specified path and resizes it to the input size required by ResNet (224x224 pixels). The image is converted into a NumPy array, which can be processed by the neural network.

The array is preprocessed to ensure it is compatible with the ResNet model. This includes mean subtraction and scaling to bring the pixel values within the expected range which is done using `preprocess_input` function which is a `resnet` library function.

The preprocessed image array is passed through the pretrained ResNet model to obtain a feature vector. This vector captures the image's visual characteristics and semantic information. The feature vector is normalized to ensure that its magnitude does not influence the similarity calculations. This step helps in making the feature embeddings more robust and comparable.

Nearest Neighbours Algorithm

In the context of our smart fashion recommender system, the Nearest Neighbours algorithm helps us discover fashion items that closely resemble the input image. By finding the “nearest neighbors” in terms of feature similarity, we can suggest fashion items that align with the user's visual preferences. This approach creates a personalized and intuitive shopping experience, enhancing user engagement and satisfaction.

The `NearestNeighbors` class is a part of `scikit-learn`, a versatile machine learning library in Python. It provides an efficient implementation of the Nearest Neighbours algorithm, making it an ideal choice for various recommendation and similarity-based tasks. The key parameters of the `NearestNeighbors` class are:

- **n_neighbors:** The number of neighbors to consider when making recommendations. This parameter affects the granularity of the recommendations. In our case, we may set it to a value like 5 to suggest a handful of visually similar fashion items.
- **algorithm:** The algorithm used to compute nearest neighbors. Common options include 'brute' (brute-force search), 'kd_tree', and 'ball_tree'. In our case, 'brute' is suitable for small datasets.
- **metric:** The distance metric used to measure similarity. 'euclidean' is a common choice, but other metrics like 'cosine' and 'manhattan' can also be used depending on the nature of the data.

In the code provided, the Nearest Neighbours algorithm is implemented using the NearestNeighbors class. It takes the extracted feature vectors as input and calculates the nearest neighbors based on the specified distance metric. Here's the relevant part of the code:

```
43
44 def recommend(features, feature_list):
45     neighbors = NearestNeighbors(n_neighbors=6, algorithm='brute', metric='euclidean')
46     neighbors.fit(feature_list)
47
48     distances, indices = neighbors.kneighbors([features])
49
50     return indices
51
```

Figure 19 choosing model

By leveraging the Nearest Neighbours algorithm, we can efficiently locate and recommend fashion items that closely match the visual attributes of the user's selected image.

Uploading the image

The `save_uploaded_file` function ensures that the uploaded image is saved in a specific location for further processing. It takes the uploaded file as input and saves its contents to the specified directory.

```
26 def save_uploaded_file(uploaded_file):
27     try:
28         with open(os.path.join('uploads', uploaded_file.name), 'wb') as f:
29             f.write(uploaded_file.getbuffer())
30         return 1
31     except:
32         return 0
33
```

Figure 20 upload the image

After saving the uploaded image, the application displays the chosen image to the user, offering a visual confirmation of their selection. Here's the relevant code snippet:

```
52 # steps
53 # file upload -> save
54 uploaded_file = st.file_uploader("Choose an image")
55 if uploaded_file is not None:
56     if save_uploaded_file(uploaded_file):
57         # display the file
58         display_image = Image.open(uploaded_file)
59         st.image(display_image)
```

Figure 21 upload the image

Once the user's selected image is displayed, the recommender system generates and displays similar fashion product images. The Nearest Neighbours algorithm aids in finding the most visually similar images to the uploaded one. The code snippet below demonstrates this process:

```
61 features = feature_extraction(os.path.join("uploads",uploaded_file.name),model)
62 #st.text(features)
63 # recommendation
64 indices = recommend(features,feature_list)
65 # show
66 col1, col2, col3, col4, col5 = st.columns(5)
67
68 with col1:
69     st.image(filenamees[indices[0][0]])
70 with col2:
71     st.image(filenamees[indices[0][1]])
72 with col3:
73     st.image(filenamees[indices[0][2]])
74 with col4:
75     st.image(filenamees[indices[0][3]])
76 with col5:
77     st.image(filenamees[indices[0][4]])
78 else:
79     st.header("Some error ocurred in file upload")
80
```

Figure 22 upload the image

In this code, the `get_model` function retrieves the ResNet50 model, and the `get_inference` function calculates the nearest neighbors for the uploaded image. The loop

then iterates through the indices of similar images, displaying them in a visually appealing grid format using Streamlit's columns layout.

4.2 SIZE PREDICTION

Fashion retailers lose billions each year to product returns and abandoned carts, fueled by the complexity around finding the right size.

E-commerce accounts for 23% of global fashion sales, according to Common Thread, up from just 14% five years ago. In key markets such as the UK, online clothing sales are predicted to overtake physical retail in 2022, with the pandemic prompting many consumers to discover the convenience of shopping from anywhere at any time.

But despite its advantages, online fashion shopping comes with a considerable challenge: navigating the complex and often erratic state of sizing. With a lack of standardization and no access to a dressing room, the small differences between retailers force customers to search through complicated size charts or place an order and hope for the best. But with the help of technology, retailers can provide an effective solution to customers' sizing struggles.

What is a size recommendation solution?

Size recommendation tools are digital solutions that analyze each customer's unique body and fit preferences to provide tailored guidance on which size a customer should purchase. Often, modern clothing fit recommendation tools are powered by artificial intelligence algorithms, which compare the customer's inputted data against product data to determine which size would fit best.

These tools come in a variety of shapes and sizes, but, essentially, they remove the complexity around sizing and provide an efficient way for customers to find the right product for their body.

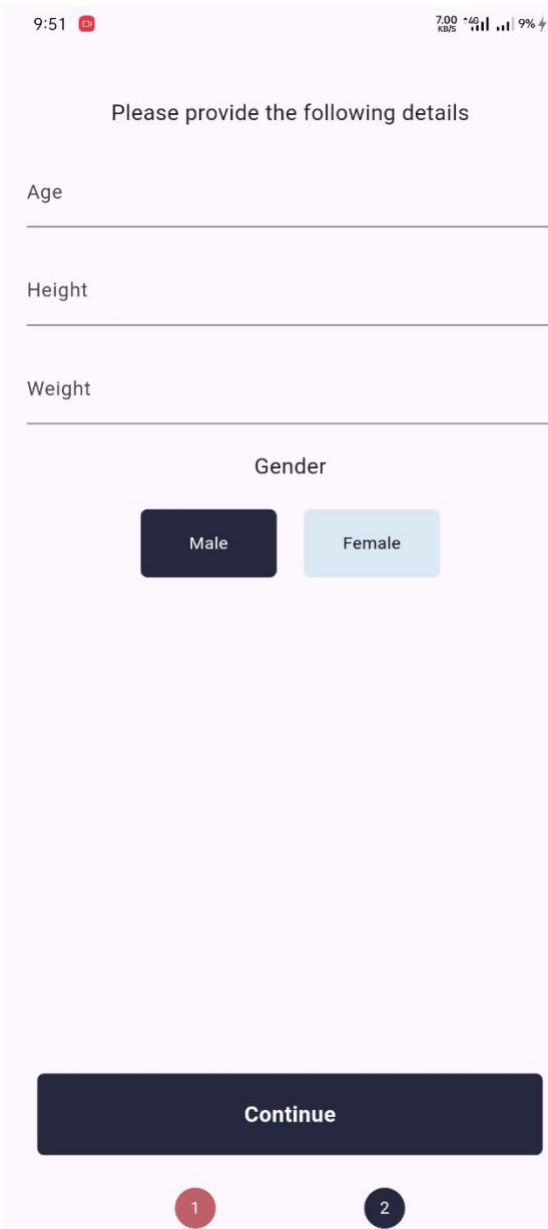
Size recommendation tools in fashion:

1. Quiz-based solutions

Quiz-based size recommendation solutions rely on the customer providing answers to a series of questions regarding their body measurements and fit preferences. These quizzes also often take into account how the customer views their own body, asking them to provide an objective assessment of their shape and physique. This data is fed into an algorithm, which compares the information provided against product data to determine the size that best meets the customer's needs.

Fit quizzes are utilized by the size recommendation tool, for instance. Customers are asked to enter their height, weight, body shape and age . This data is used to calculate a size recommendation and a percentage chance that the customer will be happy with the choice.

Quizzes are perfect for customers with privacy concerns or body image issues that make them uncomfortable about sharing photos, but this approach does come with a glaring limitation — results depend on the customer providing accurate information, and even minor inaccuracies can lead to a poor outcome.



9:51 7.00 KB/s 4G 9%

Please provide the following details

Age

Height

Weight

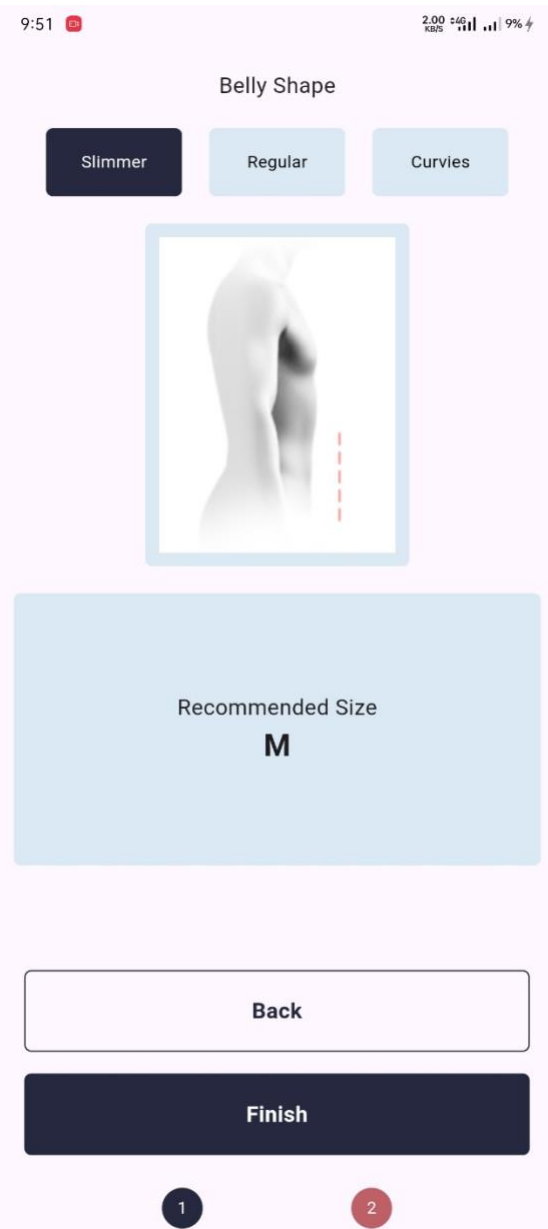
Gender

Male Female

Continue

1 2

Figure 24



9:51 2.00 KB/s 4G 9%

Belly Shape

Slimmer Regular Curves

Recommended Size

M

Back

Finish

1 2

Figure 23

2. Traditional purchases

Like purchase analysis, review analysis solutions rely on historic purchase data to determine whether a particular size is likely to fit the customer. However, in this case, the solution uses data from other customers that have purchased the same product and their opinion on its fit.

The Amazon size recommendation tool, for example, uses information provided in reviews to indicate which size the customer should order. Each product page indicates whether the item is too small, somewhat small, true to size, somewhat large, or too large, and offers guidance on whether customers should order their normal, smaller or larger size.

This offers a simple way for retailers to point customers towards an approach size with no customer input required. However, it also fails to account for differences in fit preference.

But implementing size recommendation technology won't just reduce returns. It can also attract new customers and additional purchases by removing uncertainty around sizing and appealing to the modern consumer. With digital (32%), sustainability (12%) and engagement (11%) ranked as the three biggest opportunities for fashion in 2022, the right size recommendation tool can help retailers to meet all of these consumer demands.

Advantages Clothing size recommendation technology for fashion Fit finder by AI:

Size recommendation: helping customers to buy clothing in the size that fits them best, utilising AI powered technology that matches customers' body shape . 100% customisable solution for single and multi-brand retailers.

This project predicts the clothing size of a customer using the dataset to solve the problem of customers purchasing clothes online, only to find that the size does not fit them when they receive it.

Features & Benefits size prediction:-

Help customers find the best fitting size and visually similar items instantly. Make eCommerce operations more efficient with automated product detail page descriptions. Customers ordering the wrong size? Help your website visitors to find their sizes! Convert the customer who doesn't know their size.

Decision Tree:-

Decision trees are a popular [machine learning algorithm](#) that can be used for both regression and classification tasks. They are easy to understand, interpret, and implement, making them an ideal choice for beginners in the field of machine learning.

In this comprehensive guide, we will cover all aspects of the decision tree algorithm, including the working principles, different types of decision trees, the process of building decision trees, and how to evaluate and optimize decision trees. By the end of this article, you will have a complete understanding of decision trees and how they can be used to solve real-world problems.

A decision tree is a **non-parametric supervised learning algorithm for classification and regression tasks**. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models.

A decision tree is a hierarchical model used in decision support that depicts decisions and their potential outcomes, incorporating chance events, resource expenses, and utility. This algorithmic model utilizes conditional control statements and is non-parametric, supervised learning, useful for both classification and regression tasks.

The tree structure is comprised of a root node, branches, internal nodes, and leaf nodes, forming a hierarchical, tree-like structure. It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Some tree and algorithm combinations support [missing values](#).
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. However, the scikit-learn implementation does not support categorical variables for now. Other techniques

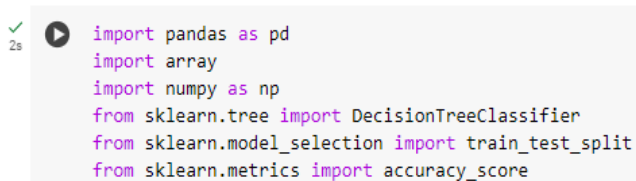
are usually specialized in analyzing datasets that have only one type of variable. See [algorithms](#) for more information.

- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The Model:

decision tree models will be used in this project to predict the dress size of customers in order to alleviate the problem of online purchases being not fitting. By alleviating this issue, the business will not only receive better reviews on their site but it will also save on costs associated with returns. The customers will also save on time and effort to raise return requests and following up on such requests

Importing Libraries:



```
import pandas as pd
import array
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Figure 25 import library

It imports necessary libraries and modules for data handling, model creation, and evaluation.

loaded and preparing the data:

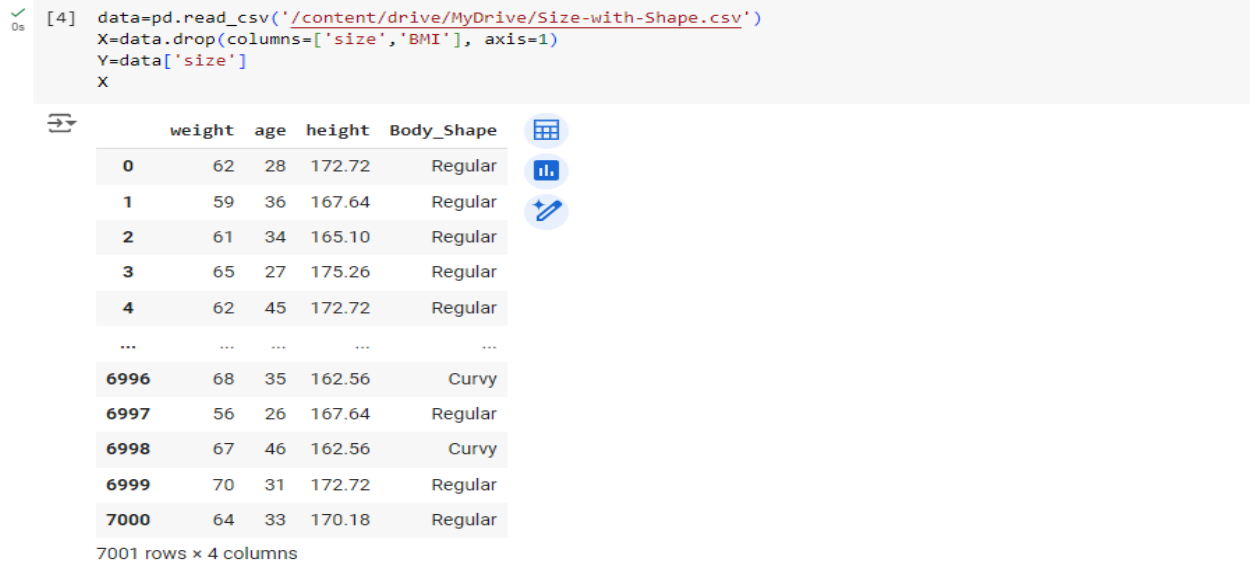


Figure 26 loaded the data

creates a new DataFrame **X** containing the features for your machine learning model. It drops the columns named 'size' and 'BMI' from the original **data** DataFrame using the **drop** method along with the **axis=1** argument, which specifies that columns should be dropped. The resulting Data Frame contains all columns except 'size' and 'BMI', presumably used as input features for your model.

creates a Series **Y** containing the target variable for your machine learning model. It selects the column named 'size' from the original **data** DataFrame, presumably to be predicted by your model.

Replacing Categorical Values with Numerical Values:

```
[5] X['Body_Shape']=X['Body_Shape'].replace(['Slim', 'Regular', 'Curvy'],[0,1,2])
Y=Y.replace(['XXS', 'S', 'M', 'L', 'XL', 'XXL', 'XXXL'],[0,1,2,3,4,5,6])
print(X)
```

	weight	age	height	Body_Shape
0	62	28	172.72	1
1	59	36	167.64	1
2	61	34	165.10	1
3	65	27	175.26	1
4	62	45	172.72	1
...
6996	68	35	162.56	2
6997	56	26	167.64	1
6998	67	46	162.56	2
6999	70	31	172.72	1
7000	64	33	170.18	1

[7001 rows x 4 columns]


Figure 27 replace value




- **X['Body_Shape']=X['Body_Shape'].replace(['Slim', 'Regular', 'Curvy'], [0, 1, 2]):** This line replaces the categorical values 'Slim', 'Regular', and 'Curvy' in the 'Body_Shape' column of the DataFrame **X** with numerical values 0, 1, and 2 respectively. This conversion is often necessary when working with machine learning algorithms as they usually require numerical input.
- **Y = Y.replace(['XXS', 'S', 'M', 'L', 'XL', 'XXL', 'XXXL'], [0, 1, 2, 3, 4, 5, 6]):** This line replaces the categorical sizes 'XXS', 'S', 'M', 'L', 'XL', 'XXL', and 'XXXL' in the target variable **Y** with numerical values 0, 1, 2, 3, 4, 5, and 6 respectively.
- prints the modified DataFrame **X** after replacing the categorical values in the 'Body_Shape' column with numerical values.

Train-Test Split:

```
[6] X_train, X_test, Y_train, Y_test = train_test_split (X,Y,test_size=0.9, stratify=Y,random_state=12)
```

```
[7] print(X.shape, X_train.shape, X_test.shape)  
X_train
```

 (7001, 4) (700, 4) (6301, 4)

	weight	age	height	Body_Shape	
2168	52	33	160.02	1	
1712	63	35	172.72	1	
4727	47	28	154.94	1	
2052	63	20	157.48	2	
4921	58	30	165.10	1	
...	
1634	55	29	162.56	1	
6519	58	27	160.02	1	
1836	56	28	160.02	1	
2498	65	24	175.26	1	
5811	63	20	175.26	1	

700 rows x 4 columns

Figure 28 train-test-split

splits the features (**X**) and target variable (**Y**) into training and testing sets.

- performs a train-test split on your feature matrix **X** and target variable **Y**. It randomly splits the data into two sets:
- **X_train**: This is the subset of features used for training your machine learning model. It contains a portion of the original dataset (10% in this case, as specified by **test_size=0.9**) that will be used to fit the model.
- **X_test**: This is the subset of features reserved for testing the trained model. It contains the remaining portion of the dataset (90% in this case) that will be used to evaluate the model's performance.
- Print the shapes of the original feature matrix **X**, the training feature matrix **X_train**, and the testing feature matrix **X_test**. It's helpful for verifying the sizes of the datasets after the split.

Training a Decision Tree classifier and Evaluate:



```
model = DecisionTreeClassifier ()
# training the Decision model with Training data
model.fit(X_train, Y_train)
```

DecisionTreeClassifier

```
DecisionTreeClassifier()
```

```
[9] X_train_prediction = model.predict(X_train)
training_data_accuracy_DecisionTreeClassifier = accuracy_score(X_train_prediction, Y_train)
```

```
[10] print(training_data_accuracy_DecisionTreeClassifier)
```

```
0.9385714285714286
```

Figure 29 training & evaluate

- initializing a Decision Tree classifier model, trains the Decision Tree classifier using the training data. The **fit()** method takes the features (**X_train**) and their corresponding labels (**Y_train**) as input and adjusts the parameters of the model to best fit the data.
- After training the model, you're using it to make predictions on the training data itself. This line predicts the target labels (**Y_train**) using the trained model and the features (**X_train**).
- calculating the accuracy of the model on the training data. The **accuracy_score()** function compares the predicted labels (**X_train_prediction**) with the actual labels (**Y_train**) and returns the accuracy score.

input data and then providing predictions:



```
input_data = (80,23,156,2)
# input_data = (68,42,165.1,1)
# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)
# XXS, 'S', 'M', 'L', 'XL', 'XXL', 'XXXL'
if (prediction == 0):
    print("your reccomend size is  XXS")
elif (prediction == 1):
    print("your reccomend size is  S")
elif (prediction == 2):
    print("your reccomend size is  M")
elif (prediction == 3):
    print("your reccomend size is  L")
elif (prediction == 4):
    print("your reccomend size is  XL")
elif (prediction == 5):
    print("your reccomend size is  XXL")
elif (prediction == 6):
    print("your reccomend size is  XXXL")
```

[6]
your reccomend size is XXXL

figure 30 input and prediction

- defines the input data, which presumably represents features such as height, weight, age, and body shape.
- the input data is converted into a NumPy array using **np.asarray()**.
- The input data is reshaped into a format compatible with the model's **predict()** function. Reshaping is necessary because scikit-learn models expect input data in a specific format. Here, **reshape(1, -1)** is used to convert the input data into a single sample with multiple features.
- The provided input data represents features related to body measurements, and the model predicts the size recommendation based on these features.
- The size recommendation is determined based on the predicted label, which corresponds to the size categories ('XXS', 'S', 'M', 'L', 'XL', 'XXL', 'XXXL').

Mobile App:



figure 31 recommendation

The Mobile app is created using the Streamlit library. Streamlit is a Python library that simplifies the process of building web applications and interactive dashboards. It allows you to create web interfaces directly from Python scripts with minimal code. When the Python script run containing the code, it initializes a Streamlit web application. This Streamlit- based web app provides a user-friendly interface for users to upload fashion images, and it uses a pre- trained deep learning model (ResNet-50) and KNN algorithm to provide fashion recommendations based on the uploaded images. Users can see the recommendations displayed in real-time on the web interface.

Deployment Flutter (Mobile App)

onboarding Screen

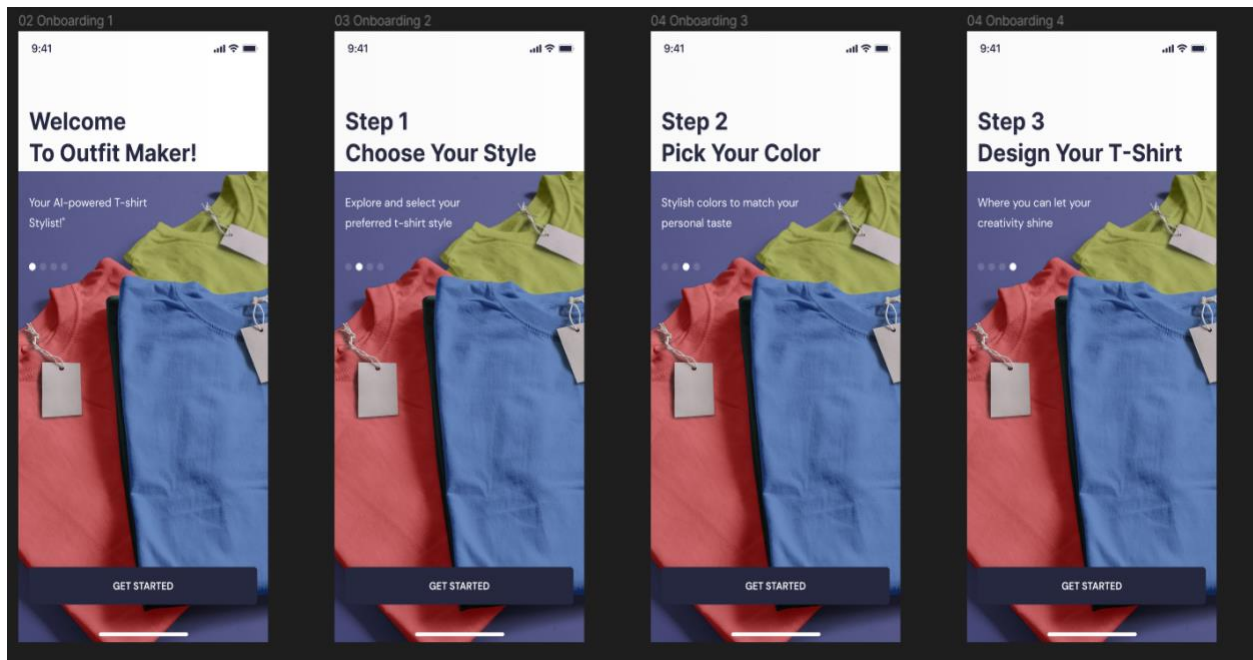


Figure 32: onboarding screen

- **Consistent Theme:** Use a consistent color scheme and style throughout all screens.
- **Navigation Indicators:** Show a progress bar or dots to indicate the current step in the onboarding process.
- **Tooltips and Hints:** Provide additional information or hints as tooltips for a more guided experience.
- This onboarding flow ensures that users are introduced to the key features of the outfit maker app in an engaging and informative manner.

Login Screen

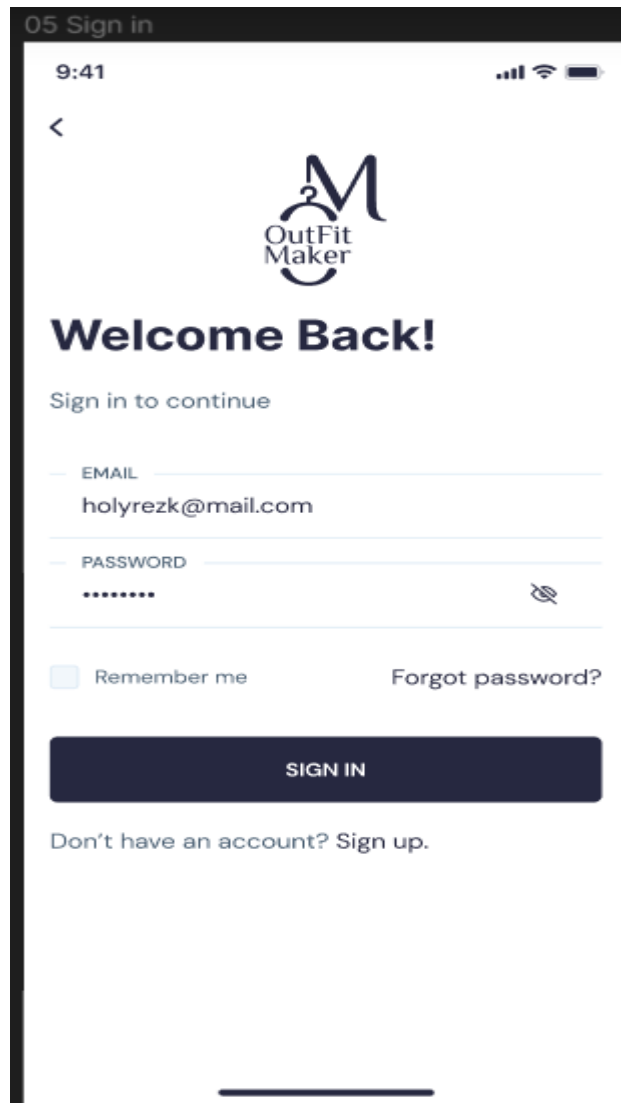


Figure 33 Login screen

On the login page, where through that page the user can register with an account to enter the home page so that the application is more secure. This page is linked to the ASP.NET server.

Whereas, after entering the user name and password, the user's data will go to the ASP.NET server in order to verify the account that was registered in the data registration page, and then come to accept the account in the event that it is correct, and if an error occurs, it will bring an error message.

There are two cases here. First, if the user does not have an account, he can go to the account registration page, through which he can create a new account. The second case is if the user already has an account and forgot the password, he can go to the forgot password page and enter the account he has, and through it he can obtain the password.

Forget password

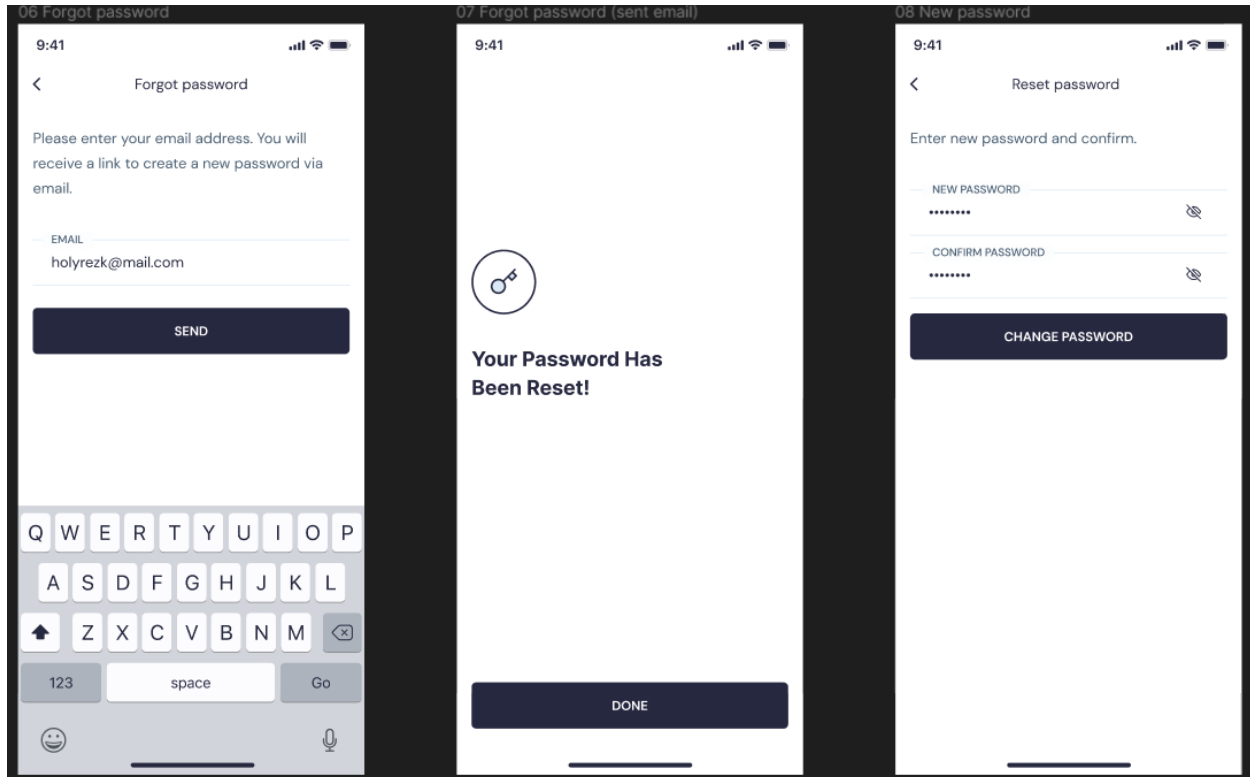


Figure 34 Forget password screen

Certainly! Here's a conceptual design for the "Forget Password" and "Reset Password" screens for an outfit maker app. The process involves the user requesting a password reset and then setting a new password.

Screen 1: Forget Password:

- Enter your email address, and we'll send you instructions to reset your password.
- Send Instructions** (Submits the email address to the system to initiate the password reset process)
- Back to Login (Navigates back to the login screen)
- After the user clicks "Send Instructions," they receive an email with a reset link.

Screen 2: Password Reset Success

- Password Reset Successful
- Your password has been successfully reset. You can now log in with your new password.

Screen 3: Reset Password:

- Reset Your Password
- Enter a new password for your account.
- New Password
- Confirm New Password
- Reset Password (Submits the new password to update the user's account)
- This flow ensures a smooth and secure experience for users who need to reset their passwords.

Register screen

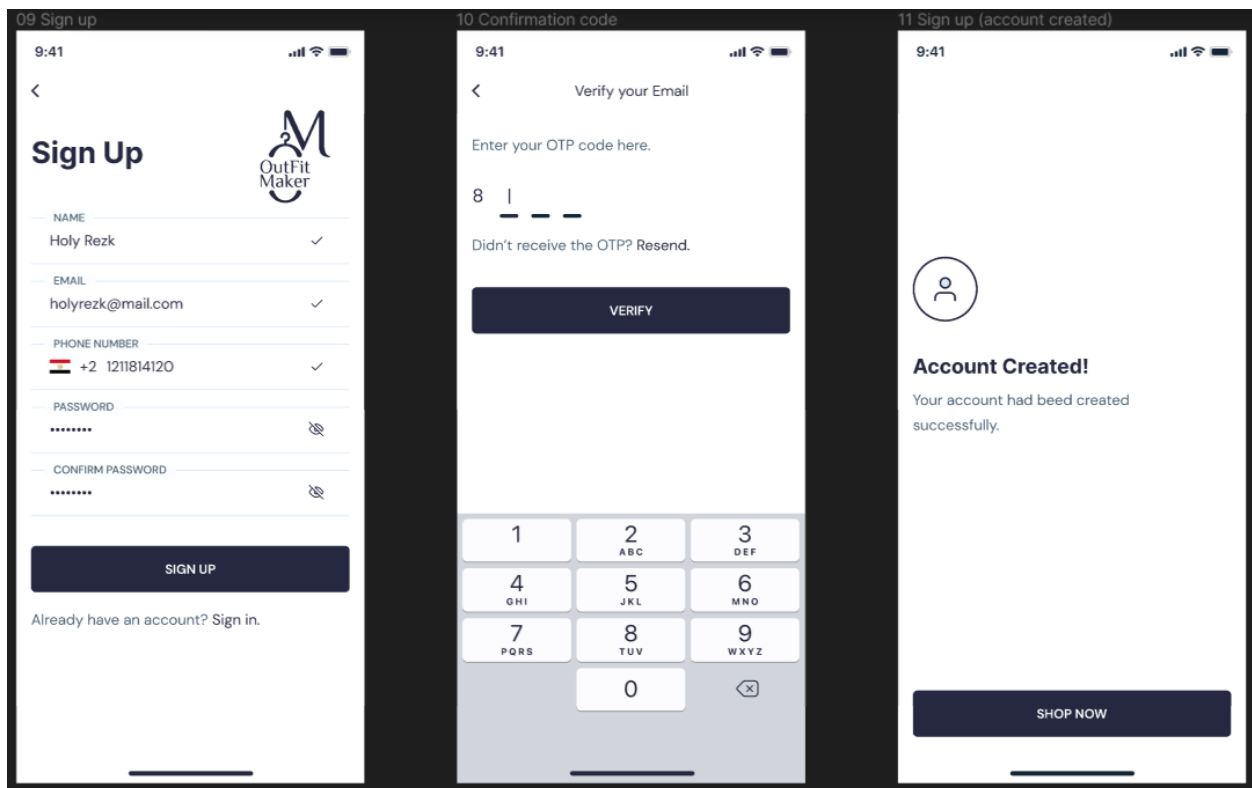


Figure 35 Register screen

Certainly! Here's a conceptual design for the "Sign Up" and "Confirm Code" screens for an outfit maker app. The process involves the user creating a new account and then confirming their email address with a code sent to them.

Screen 1: Sign Up

- Create Your Account
- Sign Up (Submits the form to create the account and sends a confirmation code to the user's email)
- After the user clicks "Sign Up," they receive an email with a confirmation code.

Screen 2: Confirm Code

- Confirm Your Email
- We've sent a confirmation code to your email. Please enter the code below to verify your account.
- Confirm (Submits the code to verify the account)
- Resend Code (Sends another confirmation code to the user's email)

Screen 3: Confirmation Success

- Email Verified
- Your email has been successfully verified. You can now log in to your account.

Home Screen



Figure 36 Home screen

The provided image shows the home screen of an outfit maker app. Based on the visual design, here's a detailed breakdown and description of the key elements:

Footer Navigation Bar

- Home (Current screen): Icon and label for the home screen.
- Browse: Icon and label to browse products.
- Create: Icon and label to create outfits.
- Favorites: Icon and label to view saved items.
- Profile: Icon and label to view and edit user profile.

User Flow

- Navigate through promotions and new collections via the hero section carousel.
- Filter products by category using the tabs (Men, Women, Kids).
- Explore best-selling items in the "Best Sellers" section.
- Reorder previous purchases from the "Order Again" section.

- Use the footer navigation to quickly switch between the home, browse, create, favorites, and profile screens.

This home screen design provides a clean and user-friendly interface that highlights key product categories and popular items, encourages user interaction with the app, and offers easy navigation to various sections.

Design studio screen

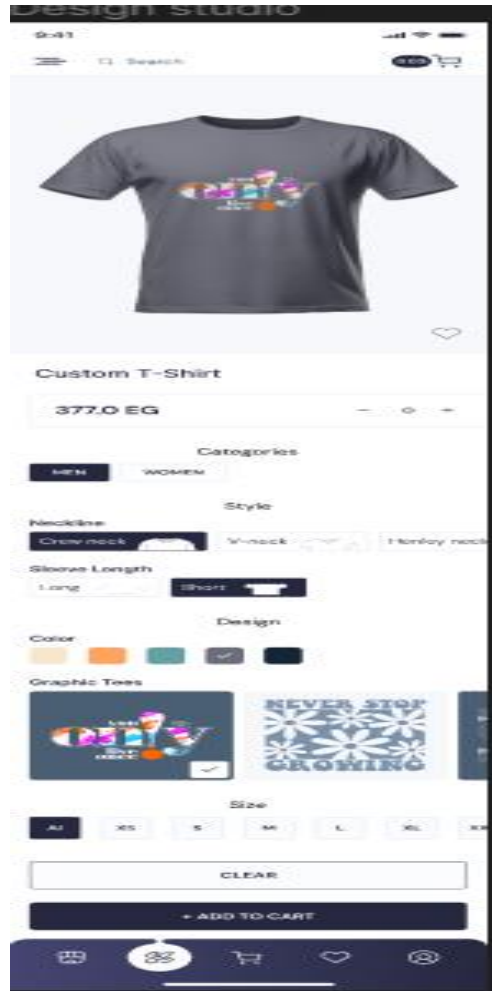


Figure 37 Design studio screen

Footer Navigation Bar

- Home: Icon and label for the home screen.
- Browse: Icon and label to browse products.
- Create (Current screen): Icon and label to create outfits or customize items.
- Favorites: Icon and label to view saved items.

- Profile: Icon and label to view and edit user profile.

General Design Elements

- Consistent Color Scheme: Light background with dark text and interactive elements, maintaining brand consistency.
- Interactive Elements: Buttons and icons are easily tappable.
- Clear Typography: Readable fonts for titles, buttons, and options.
- Visual Feedback: Highlight selected options for clarity.

User Flow

- Select Product Category: Choose between Men and Women.
- Customize Style: Select the neckline and sleeve length.
- Choose Color: Pick a color from the swatches.
- Add Design: Select a graphic design from the available options.
- Select Size: Choose the appropriate size for the t-shirt.
- Adjust Quantity: Use the "+" and "-" buttons to set the desired quantity.
- Add to Cart: Click the "Add to Cart" button to save the customized t-shirt for purchase.

quiz screen

9:51

7.00 KB/s

9%

Please provide the following details

Age

Height

Weight

Gender

Male

Female

Continue

1

2

9:51

2.00 KB/s

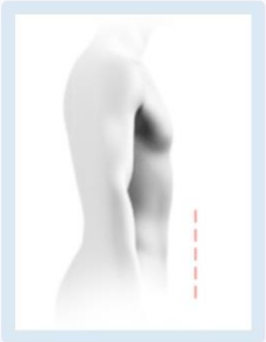
9%

Belly Shape

Slimmer

Regular

Curves



Recommended Size

M

Back

Finish

1

2

Figure 38 quiz screen

the screen displays a form for inputting personal details. Here is a summary of the elements shown:

- Instruction: "Please provide the following details"
- Input fields for:
 - Age
 - Height
 - Weight
- Gender selection with options: "Male" and "Female," with "Female" selected.
- Navigation button: "Continue"
- Step indicators: showing step 1 of a multi-step process.

This screen seems to be part of the same quiz or application process as the first image. If you need further assistance or have any specific questions, please let me know!

Category screen

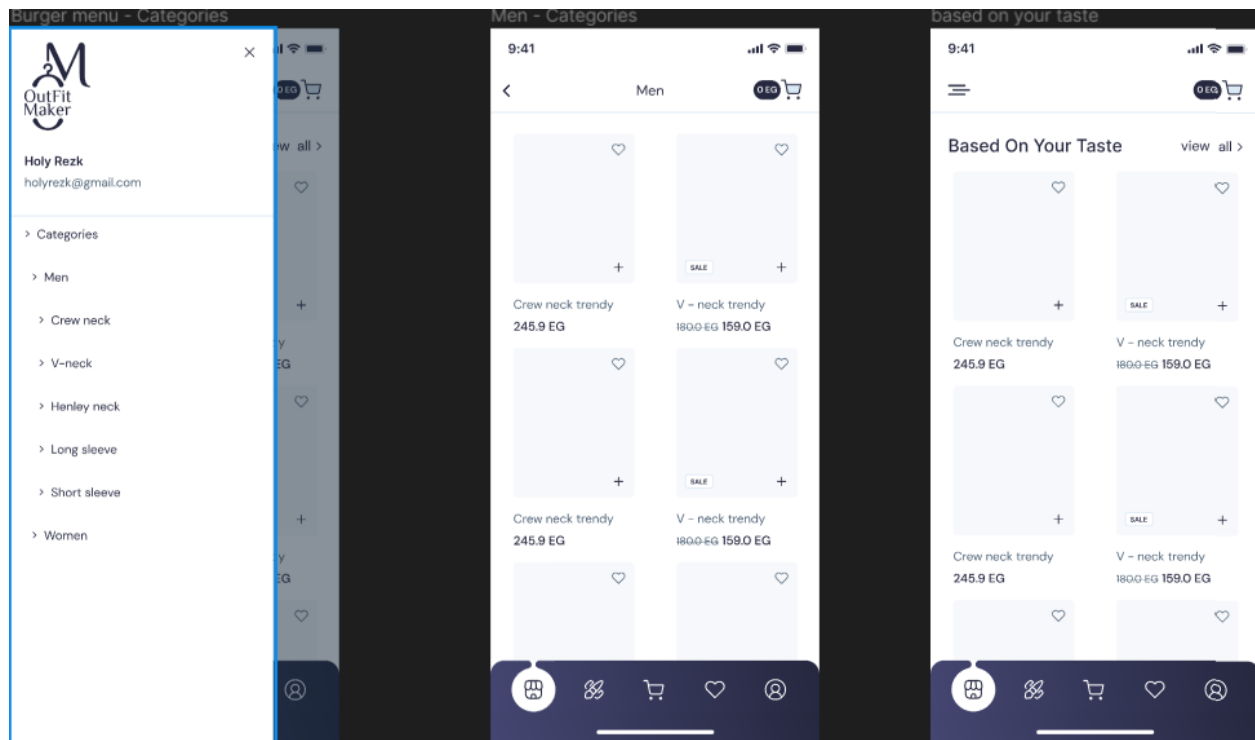


Figure 39 Category screen

Based on the provided images, here's a detailed description and breakdown of the category screen for an outfit maker app:

Footer Navigation Bar (Consistent Across All Screens)

- Home: Icon and label for the home screen.
- Browse: Icon and label to browse products.
- Create: Icon and label to create outfits or customize items.
- Favorites: Icon and label to view saved items.
- Profile: Icon and label to view and edit user profile.

General Design Elements

- Consistent Color Scheme: Light background with dark text and interactive elements, maintaining brand consistency.
- Clear Typography: Readable fonts for titles, buttons, and product details.
- Interactive Elements: Buttons and icons that are easily tappable.

- Visual Feedback: Highlight selected options for clarity.

User Flow

- Open Burger Menu: Tap the menu icon to open the category list.
- Select a Category: Choose a main category (Men/Women) and subcategories (e.g., Crew neck).
- View Products in Category: The selected category (Men) shows a grid of products.
- Based on Your Taste: Recommendations tailored to the user's preferences with an option to view all.
- Add to Wishlist: Save favorite items by tapping the heart icon.
- Add to Cart: Add items to the shopping cart using the plus icon.
- Navigate Using Footer: Switch between different sections of the app using the footer navigation bar.

This design provides a seamless and intuitive user experience, allowing users to browse categories, view product details, and manage their shopping with ease.

Order screen

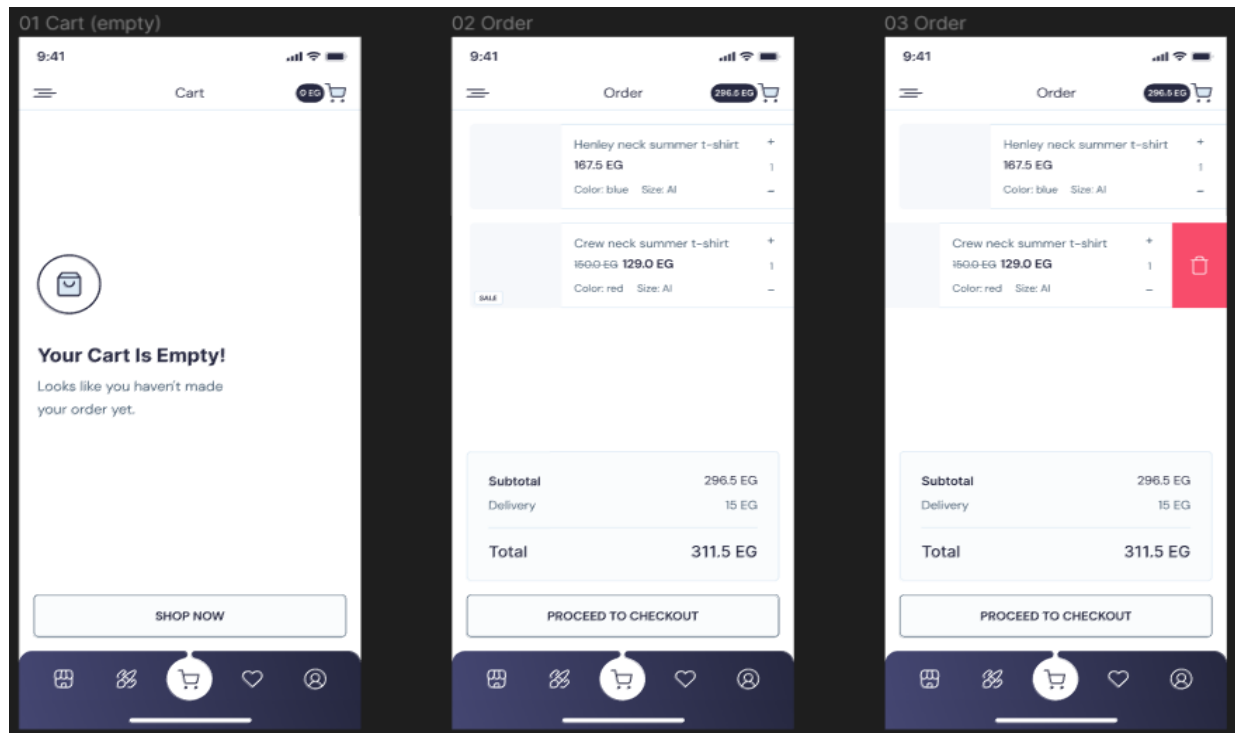


Figure 40 Order screen

The images depict different states of a shopping cart interface in a mobile app.

card (empty):

- The cart is empty.
- message "Your Cart Is Empty!" is displayed.
- prompt "Looks like you haven't made your order yet." is shown.
- There is a button labeled "SHOP NOW" to encourage the user to start shopping.
- Navigation bar at the bottom with icons for home, cart, and favorites.

Image :

- Similar to the second image but includes an option to remove an item.
- The Crew neck summer t-shirt (Color: red) has a delete icon (trash bin) next to it, indicating it can be removed from the cart.
- The total and other details remain the same.

User Flow:

- Empty Cart: User sees an empty cart and is prompted to start shopping.
- Filled Cart: User sees the items in the cart with an option to proceed to checkout.
- Modify Cart: User can remove items from the cart before proceeding to checkout.
- View Order Details: Tap on an order card to view detailed information about the order.
- Track Order: Use the "Track Order" button to see the shipping status and tracking information.
- Contact Support: Use the "Contact Support" button to get help with order-related issues.
- Navigate Using Footer: Use the footer navigation bar to switch between different sections of the app.

This order screen design provides a clear and organized interface for users to manage and track their orders effectively.

Profile screen

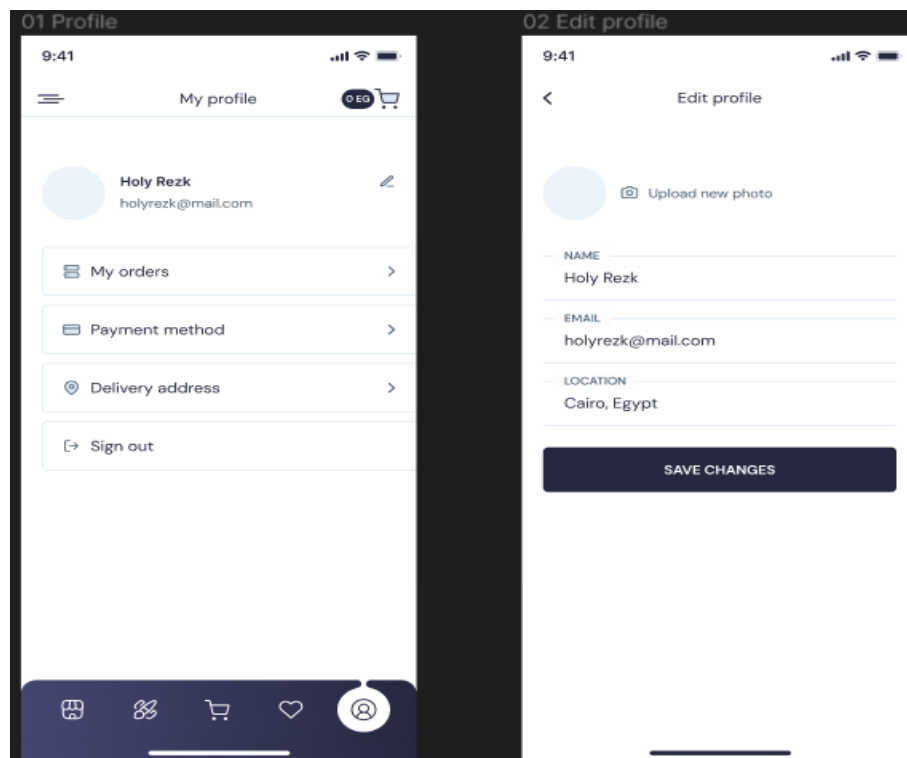


Figure 41 Profile screen

The images depict a user's profile management interface in a mobile app. Here's a detailed analysis of each screen:

- Save Changes:
- A "SAVE CHANGES" button to save any modifications made to the profile.
- Navigation:
- A back arrow at the top left corner allows returning to the previous screen without saving changes.

Key Elements:

- User Information: The profile screen provides a clear display of the user's basic information and options for managing orders, payment methods, and addresses.
- Profile Editing: Users can update their name, email, location, and profile picture. The changes can be saved using the "SAVE CHANGES" button.
- User Interaction: The design is intuitive, with clear options for navigation and actions.

User Flow:

- View Profile: Users can see their profile details and access various account management options.
- Edit Profile: Users can modify their profile information, upload a new photo, and save the changes.

These screens ensure users can easily view and update their personal information, improving the overall user experience and providing convenient access to essential account management features.

Payment screen

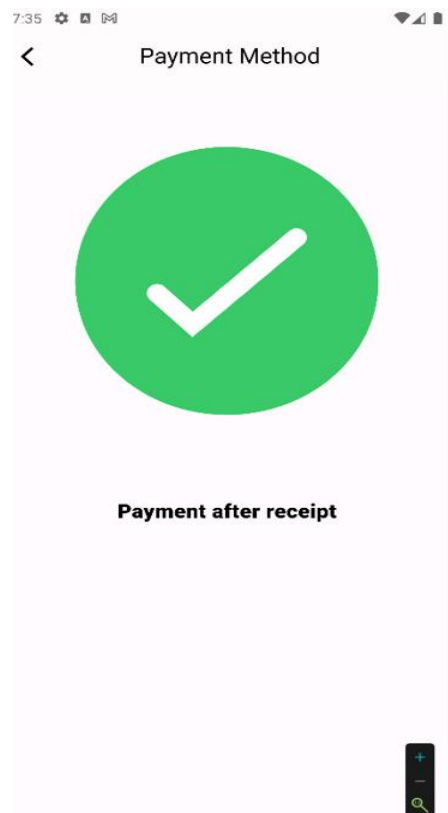


Figure 42 Payment screen

Here's a detailed description and breakdown of a payment confirmation screen specifically for the "Payment after receipt" method in an outfit maker app.

User Flow

- **Complete Payment:** After the user selects "Payment after receipt" and confirms the order, they are redirected to this screen.
- **View Confirmation:** The user sees the confirmation icon and text indicating that the payment method has been successfully recorded.
- **Navigate Back or Forward:** The user can either go back to the previous screen or proceed to other sections of the app, such as the home screen or order details.

This design provides a clear and straightforward confirmation of the "Payment after receipt" method, ensuring users are informed of their payment status and what to expect next.

Wishlist screen

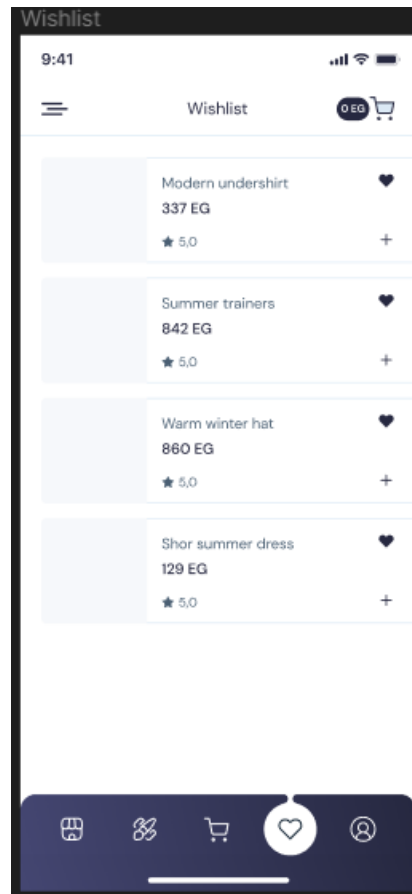


Figure 43 Wishlist screen

Based on the provided wishlist screen image, here's a detailed description and breakdown for the wishlist screen of an outfit maker app:

User Flow

- **Open Menu:** Tap the menu icon to access other sections or settings of the app.
- **View Wishlist:** See a list of products added to the wishlist.
- **Remove from Wishlist:** Tap the heart icon to remove items from the wishlist.
- **Add to Cart:** Tap the plus icon to add items to the shopping cart.
- **Navigate Using Footer:** Switch between different sections of the app using the footer navigation bar.

This wishlist screen provides a clean and organized interface for users to manage their saved items efficiently. The layout ensures easy navigation and interaction, enhancing the overall user experience.

Libraries used in the application :-

Condition builders :-Condition builders allow you to construct condition statements within your workflow. It can be used to control the flow using the Conditions or While Loop block. You can add a condition button by clicking the "Add condition" button.

Shared Preferences : Shared Preferences is used for storing data key-value pair in the Android and iOS. Shared Preferences in flutter uses NS User Defaults on iOS and Shared Preferences on Android, providing a persistent store for simple data.

Flutter Bloc :In Flutter applications, the Flutter BLoC is used to manage the state. The flutter state management feature allows handling all possible states of the application easily. It is easy to grasp the concept of the Flutter bloc. This library has excellent documentation with a lot of examples.

Dio :- Dio is a powerful HTTP client for Flutter, which provides a clean and efficient API for making HTTP requests. Dio supports a variety of features, including automatic decoding of response data, interceptors for request and response processing, and support for canceling and resuming requests.

Hexcolor :- This library is responsible for the colors inside the application .

Flutter WebView :- Flutter WebView widget displays a browser like space to show the webpage specified by URL. So, you can display a webpage just like another widget in your mobile application. In this tutorial, we will learn how to use WebView widget in Android Application.

Oktoast flutter :-A library for flutter. A pure dart toast Library. You can completely customize the style of toast

URL Launcher :- The URL Launcher is a Flutter plugin that allows your applications to launch web browsers, map applications, dialer applications, mail applications, and so on. The URL Launcher plugin works by creating intents to open applications using different URL schemes.

Introduction Screen :-Introduction Screen allows you to have a screen on an app's first launch to, for example, explain your app. This widget is very customizable with a great design.

Flutter Neumorphic :- Neumorphism buttons are specially designed buttons which will give your app an aesthetic touch. Neumorphism or Soft UI is a combination of flat design and skeuomorphism. By adding intense drop/box shadows to elements, it creates an effect that the element is able to be pushed or interacted with .

Get :- Get is a very light and powerful solution for Flutter. It combines highperformance state management, intelligent dependency injection, and path management with a fast and practical application language and the conversion between them.

Image Picker :-Image Picker plugin for Flutter. A Flutter plugin for iOS and Android for picking images from the image library, and taking new pictures with the camera.

Lottie :- Lottie for Flutter. Lottie is a mobile library for Android and iOS that parses Adobe After Effects animations exported as json with Bodymovin and renders them natively on mobile! This repository is an unofficial conversion of the Lottie-android library in pure Dart.

Flutter Credit Card : - A Flutter package allows you to easily implement the Credit card's UI easily with the Card detection.

Flutter Screen Util : - A flutter plugin for adapting screen and font size. Let your UI display a reasonable layout on different screen sizes! Note: This plugin is still under development, and some APIs might not be available yet.

Asp.net :- asp.net is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files. Asp.net supports Flutter

Asp.net Auth :- Flutter plugin for asp.net Auth, enabling Android and iOS authentication using passwords, phone numbers and identity providers like Google, Facebook and Twitter.

References:

[Image Classification Using Resnet-50 Deep Learning Model \(analyticsvidhya.com\)](#)

[\(PDF\) An Evaluation of Machine Learning Models For Deep Learning Image Classification With Fashion-MNIST Dataset \(researchgate.net\)](#)

[\(PDF\) Image anomalies detection using transfer learning of ResNet-50 convolutional neural network \(researchgate.net\)](#)

[Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation | by Nitish Kundu | Medium](#)

[IJCRT2309011.pdf](#)

[Hierarchical convolutional neural networks for fashion image classification - ScienceDirect](#)

[An Analysis Of Convolutional Neural Networks For Image Classification - ScienceDirect](#)

[\(PDF\) Image classification based on RESNET \(researchgate.net\)](#)

[The Basics of ResNet50 | ml-articles – Weights & Biases \(wandb.ai\)](#)

[Top 4 Pre-Trained Models for Image Classification + Python Code \(analyticsvidhya.com\)](#)

[Enhancing Ship Classification with CNNs and Transfer Learning \(analyticsvidhya.com\)](#)

[Image Prediction Using a Pre-trained Model - Analytics Vidhya](#)

[E-Commerce Recommendation Applications | Data Mining and Knowledge Discovery \(springer.com\)](#)

[Research on Clothing Image Classification Models Based on CNN and Transfer Learning | IEEE Conference Publication | IEEE Xplore](#)

[A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation | SN Computer Science \(springer.com\)](#)

[Fashion image classification on mobile phones using layered deep convolutional neural networks | Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia \(acm.org\)](#)

[Size Recommendation Tools Guide: Best Options for Fashion \(3dlook.ai\)](#)

[A comparative research on clothing images classification based on neural network models | IEEE Conference Publication | IEEE Xplore](#)

1.10. Decision Trees — scikit-learn 1.4.2 documentation

Background References:

1- Permatasari, P. A., & Cantoni, L. (2019, July 21–26). Mapping mobile apps on batik: A journey across heritage and fashion. In Fashion communication in the digital age. FACTUM 19 fashion communication conference (pp. 166–178). Ascona: Springer. from:

https://link.springer.com/chapter/10.1007/978-3-030-15436-3_15

2- McKinsey. (2020). The state of fashion 2020. Coronavirus update. McKinsey & Company, BoF. Retrieved from:

<https://www.mckinsey.com/~media/McKinsey/Industries/Retail/Our%20Insights/Its%20time%20to%20rewire%20the%20fashion%20system%20State%20of%20Fashion%20coronavirus%20update/The-State-of-Fashion-2020-Coronavirus-Update-final.pdf>

3- Chun, J. H. (2011). A review of the characteristics of digital art expressed in contemporary fashion. International Journal of Fashion Design, Technology and Education, 4(3), 161–171. From:

<https://doi.org/10.1080/17543266.2011.585475>

4- Noris, A., Nobile, T. H., Kalbaska, K., & Cantoni, L. (2021). Digital fashion: A systematic literature review. A perspective on marketing and communication. Journal of Global Fashion Marketing, 12(1), 32–46. From:

<https://doi.org/10.1080/20932685.2020.1835522>

5- Ma, X. (2010). A framework of E-HRM information systems in fashion enterprise. In 2010 Se international conference on information technology and computer science (pp. 305–308). Kiev: IEEE. From:

<https://ieeexplore.ieee.org/document/5557124>

6- James, A. M., Roberts, B. M., & Kuznia, A. (2016). Transforming the sequential process of fashion production: Where zero-waste pattern cutting takes the lead in creative design. International Journal of Fashion Design, Technology and Education, 9(2), 142–152. From:

<https://doi.org/10.1080/17543266.2016.1167253>

7- Wang, B., & Ha-Brookshire, J. E. (2018). Exploration of digital competency requirements within the fashion supply chain with an anticipation of industry 4.0.

International Journal of Fashion Design, Technology and Education, 11(3), 333–342. from:
<http://https://doi.org/10.1080/17543266.2018.1448459>

8- Harris, S. (2008). Catwalk goes techno (wearable technologies). Engineering & Technology, 3(18), 28–30. from:

<https://doi.org/10.1049/et:20081801>

9- Evolution of Fashion Technology from:
https://www.researchgate.net/publication/352001190_A_review_of_digital_fashion_research_before_and_beyond_communication_and_marketing

10- The Impact of Personalization on the Fashion Industry from:

<https://collectid.io/the-power-of-personalization-in-fashion/>

11- Case Studies Successful AI Implementations in the Fashion Industry from:

<https://www.linkedin.com/pulse/ai-fashion-industry-ensuring-economical-reliability>

12- Studies on consumer behavior in fashion e-commerce from:

<https://www.shipbob.com/blog/online-consumer-behavior/>

13-Introduction for fashion from:

https://www.researchgate.net/publication/345895110_AI_FOR_FASHION

14-how personalization has evolved with the advent of digital technologies from:

<https://www.sciencedirect.com/science/article/abs/pii/S0306457323001711>

15- Artificial Intelligence and Machine Learning in Fashion from:

<https://www.linkedin.com/pulse/ai-fashion-industry-ensuring-economical-reliability>

16- personalization has evolved with the advent of digital technologies from:

<https://yesplz.ai/resource/trend-report-fashion-tech-and-hyper-personalization.html>

17-AI-based automated fashion design system from:

<https://fashionandtextiles.springeropen.com/articles/10.1186/s40691-023-00360-w>

18-What is personalization from:

<https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-personalization>

19-Exploration of Personalization in Digital Communication from:

https://dl.acm.org/doi/abs/10.1007/978-3-030-50341-3_35#sec-recommendations

20-The Future of Personalization from:

<https://www.shopify.com/enterprise/personalization-trends>

21-The Future of Personalized from:

<https://sarsaricreations.com/blogs/news/the-future-of-personalized-apparel-print-on-demand-trends-for-2023>

22-Conclusion

from:

<https://www.bing.com/ck/a?!&&p=69cae476beed60ddJmltdHM9MTcwMDM1MjAwMCZpZ3VpZD0xMDA5NjcyNC02M2IyLTZlZmQtMDY3YS03NTMxNjJjZTZmYWYmaW5zaWQ9NTIyMQ&pptn=3&ver=2&hsh=3&fclid=10096724-63b2-6efd-067a->

23-introduction:

<https://www.bing.com/ck/a?!&&p=b108945687fd4ba6JmltdHM9MTcwMjg1NzYwMCZpZ3VpZD0xMDA5NjcyNC02M2IyLTZlZmQtMDY3YS03NTMxNjJjZTZmYWYmaW5zaWQ9NTIzMg&pptn=3&ver=2&hsh=3&fclid=10096724-63b2-6efd-067a-753162ce6faf&psq=%09proposed+solution+for+fashion+e-commerce&u=a1aHR0cHM6Ly9ibG9nLmRpZ2l0YWxmYXNoaW9uYWVhZGVteS5jb20vZS1jb21tZXJjZS1wbGF0Zm9ybXMtZm9yLWZhc2hpb24v&ntb=1>

24-problem description:

<https://www.eshopbox.com/blog/problems-faced-by-ecommerce-fashion-brands>