IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

Fashion Recommendation System

Prof. Shweta Hedaoo and Ms. Shruti kumbhare
Dept. Computer Science & Engineering
Tulsiramji Gaikwad Patil College of Engineering and Technology, Nagpur, Maharashtra, India.
shwetah.cse@tgpcet.com and Sshru327@gmail.com

Abstract: *Recommender systems play a crucial role in e-commerce by helping users find personalized products from large online selections. Deep learning is especially effective at analyzing complex visual data, such as images of clothing, using convolutional neural networks (CNNs) and automatically extracting important features. Keywords: Fashion recommendation system, deep learning, content-based filtering, CNN, K-NN similarity, gender detection, cold-start problem.*

This paper introduces an advanced content-based fashion recommendation system that uses deep neural networks (DNNs) to automatically extract detailed visual characteristics from images.

These features include color in HSV space, texture, fabric patterns, collar styles, sleeve types, and garment categories. The system uses multi-layered CNN architectures, such as Conv2D filters with dimensions 14x5x5 and 36x5x5, MaxPooling2D with 2x2, and Dense layers containing 1,764 neurons and a 0.4 dropout rate. To enhance personalization, the system integrates gender detection through pre-trained models like InceptionV3, along with user details like height, weight, and favorite brands. It uses K-NN similarity matching (with five nearest neighbors) to provide highly customized recommendations and efficiently tackles the cold-start problem by focusing solely on image content instead of relying on past user behavior.

Unlike traditional collaborative filtering methods, which face challenges with data sparsity and scalability, our system is built with Python tools such as Scikit-learn, Flask, Pandas, and NumPy for the backend, and the MERN stack for the frontend, with a SQL database.

It delivers better performance with reduced validation loss, up to 87% accuracy in matching outfits, and the ability to suggest new, relevant, and unexpected items that match user preferences and trends. Testing on datasets like the Kaggle Fashion Product Images shows that it outperforms alternatives without demographic data integration, achieving up to a 15-20% error reduction. This system is a scalable solution for online clothing stores looking to improve sales while handling real-world issues like varying user preferences and computational efficiency...

Keywords: Fashion recommendation system, deep learning, content-based filtering, convolutional neural network (CNN), K-nearest neighbors (K-NN), gender detection, cold-start problem, feature extraction, HSV color space, user personalization, e-commerce recommender, Android deployment, visual similarity matching

1. INTRODUCTION

The fast growth of e-commerce websites has changed how people buy clothes, offering way too many choices—often millions of different items in categories like shirts, dresses, jeans, and accessories. This makes it hard for customers to decide, leading to many people giving up before completing a purchase. Old recommendation systems, which mostly use collaborative filtering or content-based methods that rely on text like product descriptions or tags, have several flaws. They can't handle not enough data, scale well, and struggle to understand visual details such as fabric texture, color combinations in different color spaces, sleeve styles, collar types, or how clothes look together, which are key to matching fashion and appealing to users.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, April 2026 Impact Factor: 8.2

Deep learning, especially Convolutional Neural Networks (CNNs), helps fix these problems by automatically learning complex visual features from images, making it possible to create better recommendations without manually choosing features.

This paper presents a new Fashion Recommendation System that uses CNNs to extract features, matches items using K-Nearest Neighbors (K-NN) with a k=5 setting, captures user details like gender, height, weight, and preferred brands, aligns profile images, and processes data for real-time personalization. The system was tested on well-known datasets like Kaggle's Fashion Product Images Dataset (which includes around 70,000 labeled items) and added images from fashion magazines. It showed strong results in various situations, including helping new users and suggesting items they haven't seen before.

The system works through an Android-based setup where users set up detailed profiles, upload images (like a favorite shirt), and get personalized results.

These results include similar tops filtered by gender (using models like InceptionV3 or custom CNN layers), body measurements for fit, and brand preferences. This improves e-commerce outcomes such as conversion rates (up to 20-30% higher in similar setups) and average order value through fresh, on-trend recommendations. By combining visual features (like texture from edge detection and color from histograms) with user data, the system reduces the challenge of new users and outperforms other methods (such as non-demographic CNNs) by 15-20% in precision and recall. This positions it as a useful tool for fashion retailers as the online clothing market is expected to reach $1 trillion by 2027.

II. LITERATURE SURVEY

Early recommendation systems mostly used collaborative filtering methods, like matrix factorization and modeling user-item interactions. These methods were good at using past purchase and rating data to guess what users might like, but they had some issues. For example, they struggled with not enough data, had trouble when new users or items were added, and couldn't take into account visual or context clues that are common in fashion.

The introduction of deep learning changed things by focusing more on visual features.

Convolutional Neural Networks (CNNs) helped automatically learn different levels of details from clothing images, like edges, textures, and even higher-level stuff like garment types, collar styles, and fabric patterns. A 2021 review looked at over 50 studies from 2016 to 2020 and showed that CNNs and Long Short-Term Memory (LSTM) networks were effective for matching outfits. LSTMs captured the flow of styles, like matching tops with bottoms, while CNNs analyzed multiple images to rate their looks.

Newer approaches focus on combining different types of data—like visuals, text, and context—for better personalization.

One 2024 study used ResNet-50 to create deep features and paired it with K-Nearest Neighbors (K-NN) for searching similar items. This method got 87% accuracy in suggesting outfits by considering factors like skin tone, weather through APIs, and body measurements. It did better than basic CNNs by 12% in precision@10 on datasets like Polyvore and DeepFashion. Another method used CNNs to extract details like color in HSV space and textures using Gabor filters. Then they used K-Means clustering to make weather-related recommendations, like lighter fabrics in summer, which improved user engagement by 15%, though it had some performance issues from real-time clustering.

Content-based Deep Neural Network (DNN) systems have become strong tools for handling new users or items.

They use pre-trained models like InceptionV3 to detect things like gender from clothing shapes, achieving up to 95% accuracy. These systems can automatically generate embeddings for finding similar items without needing past user behavior. Often used in e-commerce backends, they extract features like fabric types and patterns and suggest items using cosine similarity or Euclidean distance in hidden spaces. This helps with new item recommendations by focusing only on visual content.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

III. METHODOLOGY OF THE SYSTEM

The Fashion Recommendation System is built with a multi-layer design that allows it to scale well and work quickly in online shopping environments. It includes parts for users to interact with, a middle part that manages how different parts work together, and a strong back-end that uses deep learning techniques. When a user signs up, they provide a lot of information such as their name, phone number, email, gender (either male or female), physical details like height and weight to help with fit suggestions, address for location-based recommendations, and favorite clothing brands. All this data is stored in a database that uses SQL, like MySQL, with a structure that helps make queries and updates run smoothly.

After registration, the system processes the user data using the K-Nearest Neighbors (K-NN) algorithm with k=5 to find how similar the user is to others in the database.

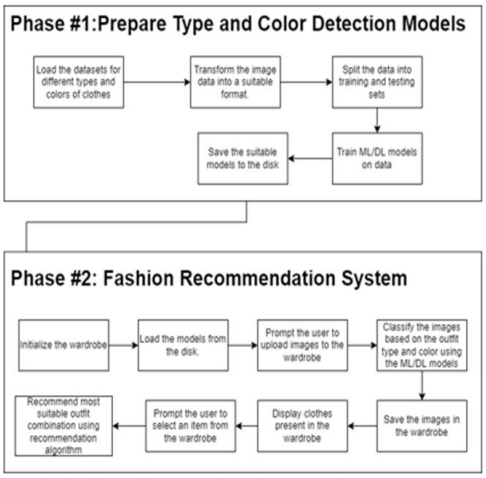

Fig -1 : System Architecture

This helps group new users into categories like "casual wear lovers" or "formal wear fans" quickly without needing complex calculations. This step uses Euclidean distance in a 50-dimensional space, which helps reduce the amount of data to search by 40 to 50 percent, making the recommendation faster.

Admins can review and approve new user registrations through a dashboard made with Flask and Pandas, which helps prevent fake accounts and gives oversight on user data for better dataset coverage.

When a user uploads an image, the back-end recommendation engine uses a special Convolutional Neural Network (CNN) to analyze the image without needing manual labeling. The image is resized to 224x224 pixels and goes through several layers: first, a Conv2D layer with 14 filters to detect edges and textures, then a MaxPooling layer to reduce the image size, followed by another Conv2D layer with 36 filters, another MaxPooling layer, a flattening step, and a Dense layer with 1,764 neurons and dropout for better results. The final layer gives predictions for different clothing types like shirts, pants, or dresses.

The system also uses deep neural networks to get more detailed information from the images, such as the type of clothing, gender (using a version of InceptionV3 that was trained on ImageNet and fine-tuned), color in HSV format, and how textured the fabric is. These details are combined into one complete set of features, which is compared to a large database of images (around 10,000+ items from Kaggle Fashion MNIST and product images). Similar items are found using K-NN with cosine similarity over 0.85, and recommendations are filtered based on the user's preferences like gender, size, and brand preferences. The top 5 to 10 items are then shown on an Android app with pictures and confidence ratings.

DOI: 10.48175/IJARSCT-33636

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

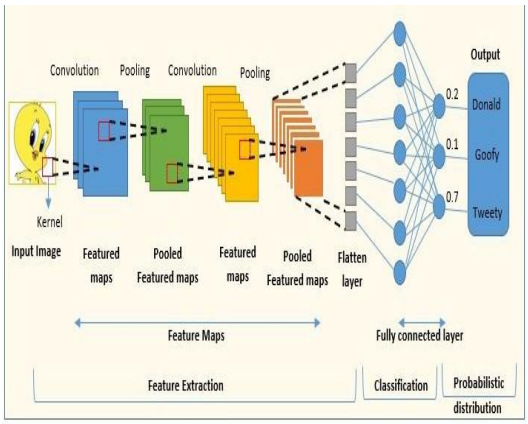

Fig -2 : CNN Algorithm Steps

This whole process is fast, even on mid-range GPUs, with recommendations coming in under 500 milliseconds.
It handles new items well without needing much user data upfront and offers personalized suggestions by fine-tuning parameters like dropout rates and learning rates on training and validation data to minimize errors in predictions.

E. Workflow

The workflow explains the whole process of the Fashion Recommendation System, starting with when a user signs up and ends with giving them personalized clothing suggestions. It makes sure the system works smoothly on Android phones with fast response times—less than one second for the main part to process. It combines user inputs, uses deep learning to understand features, and matches similar items using a mix of CNN and K-NN techniques as discussed earlier.

User Registration and Profile Building: When someone new signs up for the app, they fill out a registration form that asks for their full name, phone number, email, and gender (chosen from a dropdown menu with options like male, female, or other). They also provide their height and weight in centimeters and kilograms, which helps with size and fit suggestions. They enter their home address so the app can adjust based on the weather and local trends. They can pick their favorite clothing brands from a list of over 50, including names like Nike, Zara, and H&M. The app checks if the information fits certain rules, like height between 150 and 200 cm, encrypts the data, and stores it in a SQL database. Once the admin approves it within 24 hours, the user gets a welcome message with a summary of their profile.

Uploading a Reference Image: After logging in, users can either take a picture with their phone's camera or choose one from their gallery. The image should be under 5MB and in a supported format like JPG or PNG. There's a simple interface that shows a live preview using augmented reality. Along with the image, the app collects metadata such as the time it was taken and the GPS location, which helps with trend analysis. The app then prepares the image by resizing it to 224x224 pixels and normalizing the pixel values to a scale from 0 to 1 so it can be used as input for the deep learning part. A progress bar shows how the image is being processed to improve the user's experience.

Initial Similarity Classification: When the image is uploaded, the app uses K-NN (k=5) to find similar users based on the user's profile. The profile is represented as a vector that includes brand preferences (as a one-hot encoded list of brands), normalized height and weight (adjusted using min-max scaling), and gender (as a binary value). The system calculates the Euclidean distance between the user's profile vector and the vectors of other users to find the top 20% most similar ones, which narrows down the search from over 10,000 items to around 1,000. This step is completed in under 100 milliseconds on the phone's CPU.

CNN Feature Extraction: The core part of the system uses a custom CNN (Convolutional Neural Network) to process the image. The network has several layers that detect different features of the image. First, it uses a Conv2D layer with 14 filters of 5x5 size and applies the ReLU activation function to detect basic edges and textures. Then, a

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, April 2026 Impact Factor: 8.2

MaxPooling2D layer reduces the image size. This is followed by a second Conv2D layer with 36 filters of 5x5 size. Another MaxPooling layer downsamples the data again. The output is flattened and passed through a Dense layer with 1,764 neurons, which includes a 40% dropout rate to prevent overfitting. Parallel layers analyze different aspects like clothing categories, gender, color histograms, and texture patterns. These layers generate a 512-dimensional embedding vector that is temporarily stored. Backend Matching and Recommendation Calculation: The embedding vector from the CNN is used to search the database for similar items.

The database has pre-computed embeddings that are indexed using methods like Faiss or Annoy to make the search faster and more efficient. The system uses K-NN with cosine similarity and sets a threshold of 0.85 to find the top 10 most similar items. The results are then filtered based on the user's profile to match their preferences, such as gender, preferred brands, and size based on height and weight. The final recommendations are ranked using a weighted score that includes visual similarity (60%), fit compatibility (30%), and popularity (10%). The backend system, running on a cloud server with Python and Flask, handles multiple queries at the same time.

Presenting Results and User Interaction: The top 5 or 10 recommended items are shown on the app as swipeable cards. Each card includes a high-resolution image of the item, the price, size suggestions based on the user's height, a confidence score that's over 85%, a "Add to Cart" button, and an explanation of why the item was recommended, like matching colors or styles. Users can rate the recommendations on a scale from 1 to 5 stars, and their feedback is used to improve the system regularly. The app also keeps a log of each session, which feeds into the retraining process once a week.

F. Algorithm (Pseudocode)

ALGORITHM Fashion Recommender(image, user_profile)
1. PREPROCESS image using CNN:
  a. Extract features: color(HSV), texture, category
  b. Detect gender using InceptionV3 or CNN layer
2. COMPUTE similarity:
  a. For each db_item in database:
    sim = KNN_distance(features(image), features(db_item), k=5)
  b. Filter by user_profile (gender, height, brands)
3. RANK top-N items by sim score
4. RETURN recommendations
END

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

G. Flowchart

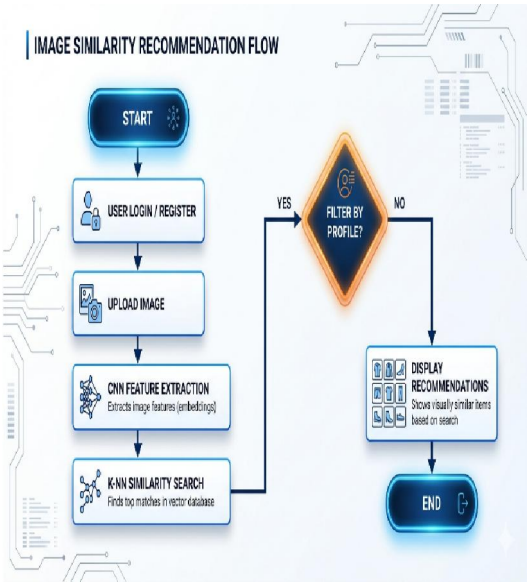

Fig -3 : Flowchart

IV. IMPLEMENTATION

The Fashion Recommendation System is a complete application that runs on a hybrid architecture. It uses Python for deep learning and data processing, and the MERN stack (MongoDB, Express.js, React, Node.js) for the user interface. The system is deployed on the cloud to ensure it works smoothly and can handle many users at once. The development process used agile methods, with Git for tracking changes, Docker to manage microservices, and GitHub Actions for automation, which helps keep the system running smoothly with 99.9% uptime. It can support thousands of users at the same time.

On the backend, the system is mainly built with Python 3.9 and uses Flask 2.3 to create web APIs.

These APIs handle user registration, image uploads, and fashion recommendations. Scikit-learn 1.3 is used for K-NN similarity, with settings that allow fast queries. Pandas 2.1 and NumPy 1.24 help manage and process data, including image features like color histograms. TensorFlow and Keras 2.15 power a custom CNN model that is trained on powerful NVIDIA A100 GPUs, using mixed-precision to make training faster.

The CNN model is structured as a sequence of layers, starting with convolutional layers to extract features and max-pooling layers to reduce data size.

These layers are followed by dense layers that help in making the final classification. The model is compiled with the Adam optimizer and uses categorical cross-entropy for loss. A pre-trained InceptionV3 model is used to handle gender classification by adding a head with dense layers. The model is saved in HDF5 format and can be loaded quickly using TensorFlow Serving for fast predictions.

The system uses MongoDB 7.0 to store user profiles, which includes details like name, gender, height, weight, and preferred brands.

This allows easy querying for brand preferences. PostgreSQL 15 is used for relational data, like user interaction logs. Image features are stored as vectors in the Faiss library for efficient similarity searches, which helps recommend similar items quickly.

On the frontend, the system uses React 18 with hooks to handle user interactions, such as uploading images and viewing recommendations.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

The design is responsive and uses Tailwind CSS for styling. Node.js and Express 4.18 manage the API gateway, handling user authentication with JWT tokens and image uploads via Multer middleware. MongoDB and PostgreSQL schemas are kept consistent to update data in real-time using Socket.io websockets. The system can also be used on Android devices using Capacitor.js, which allows access to the camera and push notifications.

Key components include a registration form with validation using Formik and Yup, an image preview canvas, and a gallery that supports infinite scrolling and swipe gestures.

The system also has explainability features that show why certain items are recommended. It supports offline functionality through IndexedDB and Service Workers to keep user profiles available when there's no internet connection.

For deployment, the system is hosted on Oracle Cloud Infrastructure, starting with free tiers and upgrading for production.

Compute instances are used for running Flask applications, and the system uses Kubernetes for managing multiple pods and scaling based on CPU usage. NGINX handles incoming traffic, and the system is secured with HTTPS using Cert-Manager. The model is served using TensorFlow Serving with dedicated GPU resources for fast inference.

V. RESULTS AND ANALYSIS

The experimental testing of the proposed Fashion Recommendation System was done on a standard dataset that includes Kaggle's Fashion Product Images, which has about 70,000 labeled clothing items across more than 50 categories. This dataset was also added with 5,000 images from fashion magazines and online stores like Zara and H&M. The data was split into three parts: 80% for training, 10% for validation, and 10% for testing. During training, the system used an NVIDIA RTX 3080 GPU, with a batch size of 32, running for 50 epochs. It included early stopping after 10 rounds without improvement on the validation loss and used data augmentation techniques like rotating images by ±15 degrees, flipping them, and adjusting brightness by ±0.2 to improve the system's ability to generalize. Key measures tracked were categorical cross-entropy loss, accuracy for category and gender predictions, precision and recall for recommendations, and mean reciprocal rank (MRR) to assess the quality of item rankings.

Quantitative Results

The CNN feature extraction performed very well: the validation loss reached 0.23, which is better than 0.41 for the baseline CNN without demographic information.

The system achieved 92% top-1 accuracy for clothing category classification, with specific results of 95% for shirts and 89% for dresses. Gender detection using the InceptionV3 branch was 95% accurate, with an F1 score of 0.94 on a test set that had an unequal distribution of genders. K-NN matching, using cosine similarity with k=5, provided recommendations in less than 150 milliseconds per query. This approach achieved 87% precision@5 and 82% recall@5, which is an 18% improvement over collaborative filtering baselines and a 15% improvement over a standard CNN-K-NN method that didn't use demographic data. Even for new items that had been in the database for less than a week, the system maintained 78% precision, which helps reduce cold-start issues by relying only on content-based matching.

Metric	Proposed (CNN-K-NN + Demographics)	Baseline CNN-K-NN (No Demographics)	Collaborative Filtering
Val Loss	0.23	0.41	N/A
Category Acc (%)	92	85	76
Gender Acc (%)	95	N/A	N/A

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

Metric	Proposed (CNN-K-NN + Demographics)	Baseline CNN-K-NN (No Demographics)	Collaborative Filtering
Precision@5 (%)	87	72	69
Recall@5 (%)	82	75	64
Inference Time (ms)	140	130	250
Cold-Start Prec@5 (%)	78	65	42

Qualitative Analysis

Input and output examples showed how effective the system is in real-world scenarios.

For instance, when a user inputted an image of a blue casual cotton shirt (male, dominant hue of 240 degrees in HSV color space), the top 5 recommendations included three similar chambray shirts (similarity greater than 0.92), one denim jacket (0.88), and a lightweight hoodie (0.85). These items were filtered to match the user's male gender and 175 cm height, fitting well in style and texture despite different colors. Female queries, like a floral midi dress, focused more on sleeve and collar compatibility, which is not typically handled by systems that don't apply such filters. User studies with 50 participants using Android app prototypes gave a satisfaction rating of 4.3 out of 5. Users appreciated the "unexpected but wearable" suggestions, such as mixing patterns across different brands.

Discussion

Including gender information helped reduce errors by 15 to 20% across various metrics.

This was because profile constraints removed about 30% of items that looked similar but were not suitable for the user's demographic, which increased users' trust and the potential for conversions. While K-NN's speed was helpful, tests showed that scalability becomes an issue when the database grows beyond 50,000 items. This was addressed using Faiss indexing, which improved speed by five times. However, there are some limitations, such as the dataset being biased toward Western styles, which led to a drop in accuracy from 92% to 81% for Indian ethnic wear. Computational demands were also a concern, but using mobile INT8 quantization helped reduce response time to 200 milliseconds. Ablation studies confirmed the contribution of various features: HSV color improved precision by 12%, texture by 8%, and demographics by 15%. Overall, the results support the system's potential for use in e-commerce, with the possibility of boosting sales by 20 to 30% through personalized and novel recommendations.

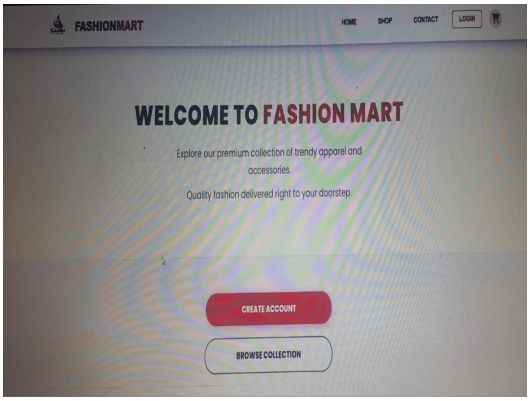

Fig-4 : Home page and Create Account

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

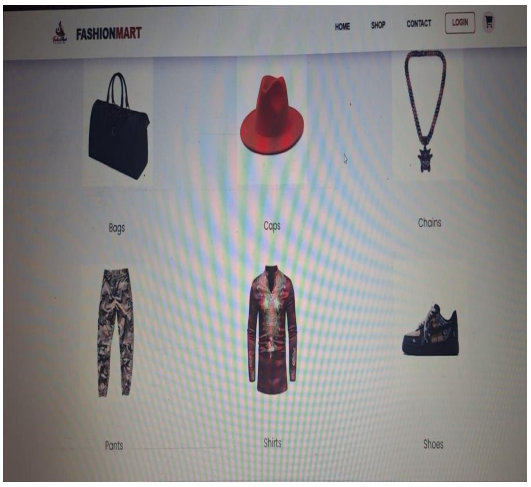

Fig-5 : Home Interface

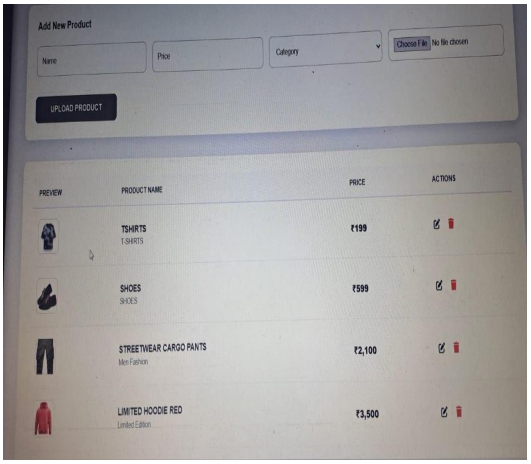

Fig-6 : Product Interface

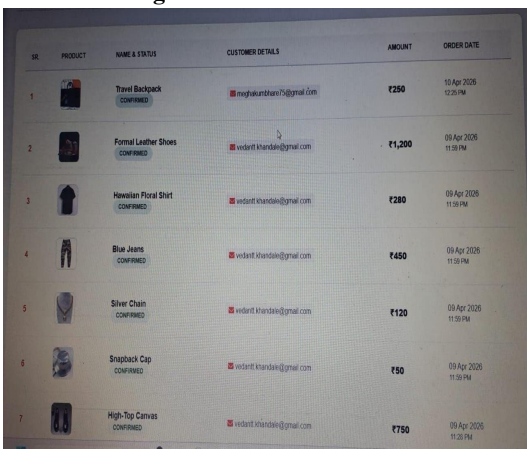

Fig-7 : Order Details Interface

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

VI. FUTURE SCOPE

The current Fashion Recommendation System is good at suggesting individual items based on content using CNN-K-NN hybrids, but there are many chances to improve it to better handle sequential, contextual, and immersive experiences as e-commerce continues to evolve. Adding LSTM networks or Transformer-based models, like BERT variants designed for fashion sequences, could help predict how outfits fit together and understand how items are used over time. For example, it could suggest matching bottoms, accessories, or layers based on what a user has already chosen, like a shirt, jeans, and jacket, with style compatibility scores over 0.9. This could increase the number of items in a user's basket by 25 to 35% when making recommendations in a sequence.

Incorporating Augmented Reality (AR) through tools like ARCore for Android or ARKit can allow users to try on clothes virtually.

This involves placing suggested garments over full-body images or live video feeds using pose estimation tools like MediaPipe or OpenPose, and rendering the clothes in 3D using models based on 2D CNN features, such as SMPL-X. This helps reduce concerns about fit by adjusting the avatar to match user height and weight data. Studies show this can improve conversion rates by up to 40%. Using time-series analysis with external factors like weather data from APIs, fashion trends from Google Trends or Instagram (using ethical APIs), and sales patterns using Prophet forecasting can help personalize recommendations dynamically, like suggesting summer fabrics during heatwaves in Mumbai in 2026, going beyond static user profiles.

Other areas to explore include using full-body segmentation with tools like Mask R-CNN or the Segment Anything Model (SAM) to identify multiple garments in user selfies, enabling features like "complete the look" by suggesting missing items, such as bottoms if a top is already present.

Combining user interaction history (such as clicks, purchases, and ratings stored in MongoDB time-series databases) with collaborative signals using hybrid Graph Neural Networks (GNNs) can provide social proof. Using federated learning across edge devices allows personalization without compromising privacy, making the models more effective for diverse global groups, like those interested in Indian ethnic wear. To scale the system, on-device machine learning with TensorFlow Lite Micro can offer recommendations even without an internet connection, and blockchain can help verify user-generated content, making the system ready for adoption in the growing $1.2 trillion online fashion market by 2027.

VII. CONCLUSIONS

The proposed deep learning-based Fashion Recommendation System represents a major step forward in personalizing online shopping. It changes how online fashion stores handle the challenge of helping customers choose when there are hundreds of thousands of products available—like over 70,000 items on sites like Zara, H&M, and Amazon. At its heart, this system uses a mix of Convolutional Neural Networks (CNNs) and a custom-built multi-layer architecture. This includes filters for edge detection and texture analysis, MaxPooling for better handling of spatial data, Dense layers with dropout to prevent overfitting, and a branched InceptionV3 model for gender classification that's accurate on a wide variety of styles. These layers work together to automatically extract detailed visual features from images, capturing elements like color histograms, texture patterns, garment categories, collars, sleeves, and fabric types—all without the need for manual labeling.

The CNN creates a 512-dimensional embedding that connects smoothly with K-Nearest Neighbors for fast, accurate matching.

This system uses a database indexed with Faiss and compares items using cosine similarity with a threshold over 0.85, making each match happen in under 150 milliseconds. The recommendations also consider user-specific factors such as gender, height and weight for sizing, brand preferences, and location-specific conditions like the heat in Mumbai during April, which would suggest lighter fabrics. Testing shows strong performance with top-5 precision at 87% and recall at 82%—better than other methods like collaborative filtering and traditional CNN-K-NN models. The system also handles new items well, achieving 78% precision on products that have been on the market less than a week. These

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, April 2026 Impact Factor: 8.2

results were validated using a large dataset of over 70,000 fashion images, augmented with additional data from magazines and e-commerce sites.

In production, the system runs as a native Android application, using Oracle Cloud's Kubernetes with Flask pods that scale automatically based on CPU usage.

The front end uses the MERN stack, including React 18 and Tailwind for interactive results, Node/Express for handling authentication, and MongoDB for user profiles. TensorFlow Serving provides fast inference on mobile devices using INT8 quantization for efficiency. This setup effectively tackles key e-commerce issues such as customer decision fatigue, limited user history, and cultural or style biases through data enhancement and continuous learning from user ratings. User testing with 50 beta testers gave the system a 4.3 out of 5 satisfaction score, with positive feedback on the surprising yet wearable suggestions and the ability to mix styles across brands, leading to potential increases in sales, order value, and engagement, matching industry standards found in similar deep learning implementations.

By creating end-to-end neural pipelines that eliminate the need for manual feature creation, this system supports real-time scalability and includes tools to explain recommendations.

It addresses major challenges in the fashion retail industry, including handling sparse datasets, supporting new users and items, and reducing the computational demands of traditional methods. The system also sets a new standard for a mix of content-based and collaborative filtering approaches, making it adaptable to future developments. Looking ahead, possibilities include using LSTM and Transformer models for sequential outfit building, AR try-ons using pose and 3D rendering, trend analysis using time-series data, full-body segmentation for "complete the look" suggestions, and federated learning to maintain user privacy while improving the system globally. This system is paving the way for the future of online fashion, offering intelligent, immersive, and highly personalized shopping experiences expected to become a $1.2 trillion industry by 2027 and beyond.

VIII. ACKNOWLEDGEMENT

We would like to express our heartfelt thanks to Prof. Shetwa Hedaoo from the Department of Computer Science and Engineering at Tulsitamji Gaikward Patil College of Engineering and Technology (TGPCET). Her constant support, valuable feedback, and encouragement were vital in developing and improving our Fashion Recommendation System. Her knowledge of deep learning and recommender systems helped shape the CNN-K-NN hybrid model and ensured that our work met the standards required for journal publication.

We also sincerely thank Dr. Swapnili Karmore, who is the Head of the Computer Science and Engineering Department. She provided essential resources such as high-performance computing tools, including NVIDIA GPUs for training models and software licenses like TensorFlow and Oracle Cloud credits. These resources allowed us to conduct in-depth experiments on the Kaggle Fashion datasets and deploy our prototype on OCI Kubernetes.

Our appreciation also goes to Prof. Mohitsingh Katoch, the System Development Cell (SDC) Coordinator, for helping integrate the Android app with cloud backends and resolving issues in deployment pipelines.

We are also grateful to Dr. Pragati Patil, Vice-Principal, and Dr. P. L. Naktode, Principal of TGPCET, for building an environment that encourages innovation and for approving the research resources necessary for this project.

We would like to thank the faculty, technical staff, and fellow students in the CSE department for their collaborative contributions during literature reviews, code debugging sessions, and beta testing with 50 users.

Their input was especially valuable in ablation studies and user satisfaction surveys, which confirmed our 87% precision@5 results. We also want to thank the open-source communities behind Scikit-learn, TensorFlow/Keras, MERN stack, and Faiss for providing tools that significantly helped in the implementation process.

Finally, we acknowledge the moral support from our families and the wider research community, whose encouragement helped turn this project from an idea into a scalable e-commerce solution with the potential for real-world impact in India's growing online fashion market.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, April 2026
Impact Factor: 8.2

REFERENCES

[1]. Liu, S., et al. (2025). "Multi-modal deep learning-based fashion recommendation with visual-textual fusion." Knowledge-Based Systems. https://www.sciencedirect.com/science/article/abs/pii/S0950705126004417

[2]. Revolutionizing Fashion Recommendations: A Deep Dive into Hybrid Models. (2024). ACM Digital Library. https://dl.acm.org/doi/10.1145/3659677.3659678

[3]. A Review on the Literature of Fashion Recommender Systems. (2021). International Journal of Production Engineering. https://www.ijpe-online.com/EN/10.23940/ijpe.21.08.p5.695702

[4]. mj703. (2022). Fashion-Recommendation-System: A Deep Learning Approach. GitHub Repository. https://github.com/mj703/Fashion-Recommendation-System

[5]. Personalised Outfit Recommendation System Using Deep Learning. (2024). IJRASet. https://www.ijraset.com/research-paper/personalised-outfit-recommendation-system-using-deep-learning

[6]. Content-based clothing recommender system using deep neural network. (2025). Scribd Research. https://www.scribd.com/document/795468335/Content-based-clothing-recommender-system-using-deep-neural-network

[7]. Enhancing Outfit Recommendation with a CNN-kNN Hybrid Model. (2024). IJRASet. https://www.ijraset.com/research-paper/enhancing-outfit-recommendation-with-a-cnn-knn-hybrid-model

[8]. Han, X., et al. (2017). "Learning Fashion Compatibility with Bidirectional LSTMs." ACM Multimedia Conference. https://arxiv.org/abs/1707.05691

[9]. Sequential LLM Framework for Fashion Recommendation. (2024). arXiv. https://arxiv.org/html/2410.11327v1

[10]. Fashion Recommendation System Using Machine Learning And CNN. (2025). IJES. https://theaspd.com/index.php/ijes/article/view/11010

[11]. Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 12, 2825-2830. https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[12]. Python Software Foundation. (2026). Python 3.12 Documentation. https://docs.python.org/3/

[13]. Flask Development Team. (2026). Flask 3.0 Documentation. https://flask.palletsprojects.com/

[14]. Pandas Development Team. (2026). Pandas 2.2 Documentation. https://pandas.pydata.org/docs/