Fashion Transformer

Vinay Edula(241110024)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
vinaykr24@iitk.ac.in

Pokala Dattatreya(241110048)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
pokala24@iitk.ac.in

Voora Nagendra (241110044)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
nagendrab24@iitk.ac.in

Telugu Sudhakar(241110077)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
tsudhakar24@iitk.ac.in

Tippireddy Yashwanth(241110082)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
yashwantht24@iitk.ac.in

Moka Pavan Kumar(241110043)
Department of CSE
Indian Institute of Technology, Kanpur
Kanpur, India
pavankumar24@iitk.ac.in

***Abstract*—Selecting a stylish and appropriate outfit is a common daily challenge that can be time-consuming and stressful. People often struggle with deciding what to wear, worrying about occasion appropriateness and social perception, which leads to wasted time (on average 17 minutes for women and 13 minutes for men in outfit selection) and reduced confidence. In this work, we propose Fashion Transformer, an end-to-end deep learning system for automated outfit recommendation from a user's own wardrobe. The system addresses key limitations of prior fashion recommendation approaches by integrating personal wardrobe data and occasion awareness into outfit compatibility modeling. Our Fashion Transformer architecture learns compatibility among clothing items, enhanced with category specific embeddings and rich visual features. It consists of four modules: (1) an image classifier that uses CLIP embeddings and MLP classifiers and organizes the user's wardrobe by item category and attributes, (2) Occasion detection from user input text (3) an Outfit Category Generator (OCG-Net) was proposed but not implemented; required outfit categories are predefined for given occasion, currently not predicted dynamically and (4) A 16-head Transformer model composes a complete outfit by modeling compatibility across FashionCLIP embeddings with category tokens. The model supports both compatibility prediction and complementary item retrieval using cosine similarity on enriched transformer embeddings. We evaluate the system on large-scale fashion dataset called polyvore and demonstrated that it outperforms several state-of-the-art baselines on key metrics, while achieving comparable performance to top performing models in others, particularly in compatibility prediction and complementary item retrieval. The proposed approach shows practical potential to save users time and provide confidence in their attire by delivering personalized, occasion-appropriate outfit recommendations. We also implemented User interface, It is organized into six intuitive tabs—Image Selector, Predict Attributes, Full Outfit Generator, Occasion Outfit, Delete Image, and User Mode—each dedicated to a key outfit-management task Users can upload, save, and delete dress images, automatically predict their category and article type, and generate full or occasion-specific outfits with a single click. The User Mode toggle allows switching between profiles for personalized recommendations and history tracking.[Refer Appendix for User Interface]**

***Index Terms*—fashion recommendation, outfit compatibility, transformer models, multi-modal learning, personal wardrobe, occasion-aware recommendation**

I. Introduction

Choosing a well-coordinated outfit that suits an occasion is a non-trivial task many people face daily. The process of mixing and matching garments from one's wardrobe can be time-consuming, often averaging 15 minutes or more each day. Moreover, individuals worry whether their chosen outfit is appropriate for the event and aligned with social expectations, which can reduce confidence in their appearance. These challenges are exacerbated by the vast number of clothing options available – the fashion industry produces 100–150 billion garments annually– and the frequent purchase of new clothing (approximately 36there is a strong practical need for intelligent systems that can assist users in outfit selection, ensuring both style compatibility and occasion appropriateness.

Fashion outfit recommendation has been an active area of research in computer vision and recommender systems. Early work focused on predicting fashion compatibility, determining whether a set of clothing items go well together in style. This is often formulated as a compatibility prediction task or a fill-in-the-blank (FITB) test where a model must choose the item that best completes an outfit. Various methods, including embedding learning and metric learning, have achieved significant success in assessing compatibility, with some models reaching over 90(AUC) in compatibility classification. However, existing approaches have notable limitations: many do not guarantee a complete outfit (they may focus only on pairwise compatibility or recommend one item at a time), they often ignore the occasion or context for which the outfit is intended, and they typically use generic fashion data rather than an individual's personal wardrobe. For instance, prior systems might retrieve multiple tops and no bottoms for a look, or miss essential categories like footwear. Some require the user to input text descriptions or adjust parameters at multiple steps, which can be inconvenient. These shortcomings limit the practical usefulness of such systems in real-world scenarios where an outfit must be both compatible and appropriate for a specific event, using clothes the user owns.

In this paper, we address the above challenges by clearly posing the research question: Can we automatically generate a complete, occasion-appropriate outfit from a user's own wardrobe using deep learning, while ensuring that all items are mutually compatible? We present Fashion Transformer, a novel framework that combines personal wardrobe data with state-of-the-art deep models for outfit generation. The key idea is to integrate four components – wardrobe images classification, occasion detection from user input text, outfit categories retrieval, and a transformer-based compatibility model – into a unified system. By leveraging the user's wardrobe images as input, our approach adds a layer of personalization that most existing work lacks. Moreover, by incorporating an occasion classifier and an outfit schema generator, our system explicitly accounts for context (e.g. formal interview, casual outing, etc.), ensuring the recommended outfit includes all essential garment categories for that event. Unlike earlier methods that generate outfits from generic collections or require textual cues at each step, our approach automatically produces a full outfit from scratch based on an initial user query (such as the occasion) and the user's clothing inventory.

The contributions of this work are threefold. First, we develop an end-to-end system architecture consisting of four coordinated modules, introducing a transformer-based outfit compatibility model enhanced with category-specific embeddings and multi-head attention to capture diverse style aspects. Second, we leverage multiple datasets – a large fashion item dataset for pre-training a wardrobe classifier, an outfit dataset (Polyvore) for learning compatibility, and an events-based fashion dataset for occasion understanding – combining them to address the problem holistically. Third, we empirically evaluate the proposed Fashion Transformer against existing baseline models in terms of outfit compatibility prediction accuracy and complementary item retrieval success. Our method achieves superior performance (e.g., a compatibility AUC of 0.95, F1-score 0.85) compared to recent benchmarks, and demonstrates robust generalization to outfits with unseen items (disjoint test sets).

The rest of this paper is organized as follows. Section II reviews related works in fashion outfit recommendation and their limitations. Section III describes the proposed methodology in detail, including the system architecture and the novel Fashion Transformer model. Section IV outlines the experimental setup, datasets used, and training configurations. Section V presents the results, comparing our model with baselines on multiple metrics. Section VI provides a discussion of insights, practical implications, current limitations, and future work directions. Finally, Section VII concludes the paper with a summary of contributions and potential impact.

II. Related Work

The task of outfit recommendation has evolved significantly with advances in deep learning, visual representation, and user personalization. Various approaches have explored compatibility prediction, outfit generation, contextual awareness, and item retrieval. This section reviews 12 relevant papers that collectively represent the progression of research in this field.

One foundational work in compatibility modeling is the Multi-Layered Comparison Network proposed for Outfit Compatibility Prediction and Diagnosis. This model extracts multi-level visual features using CNN layers and aligns visual and semantic information in a shared space via a Visual-Semantic Embedding (VSE). It uses multiple loss components to enhance discriminative power and achieved 62.47% accuracy on the Fill-In-The-Blank (FITB) task and an AUC of 90.70% on the Polyvore dataset. However, its reliance on a minimum of three items per outfit and lack of diagnostic interpretability limits flexibility in dynamic outfit generation[3].

Graph-based models offer another perspective. The Context-Aware Visual Compatibility Prediction model represents fashion items as nodes and applies Graph Convolutional Networks (GCNs) to infer compatibility through edge prediction. Each edge encodes co-occurrence in an outfit, with compatibility scores derived from average edge probabilities. Achieving over 77% FITB accuracy and 91% AUC, this approach performs well on dense item sets but degrades when outfit size decreases or fewer neighbors are available (lower k)[4].

Attention-Based Outfit Modeling introduced by Han et al. presents a dual-network design: the Fashion Item Relevancy Network (FIR) encodes item compatibility, while the Outfit Preference Network (OP) uses Multiple Instance Learning (MIL) to score entire outfits. This framework achieved state-of-the-art FITB accuracies of 88.63% and 86.06% on Polyvore and iFashion, respectively. Yet, it lacks user controllability through textual inputs, limiting personalized guidance in outfit construction.[1]

For handling cold-start scenarios, Visual Preference Modeling was proposed to capture user style preferences using attribute vectors extracted from the ModaNet dataset. The system filters outfits based on gender, occasion, and visual features, clustering results for recommendation. It reached a category accuracy of 73.5% and attribute accuracy of 89.1%, but failed to incorporate multimodal user cues like seasonal preferences or textual descriptions.[2]

To introduce occasion-awareness, an efficient framework was developed combining ResNet-50 for image features and GloVe embeddings for text. It jointly optimizes compatibility loss and visual-semantic embedding objectives while training an auxiliary classifier to predict occasion categories. While the overall AUC was high (0.99), the occasion-specific AUC dropped to 0.77 and FITB accuracy to 39%, indicating the difficulty of distinguishing visually similar events.[5]

Another line of research, Attention-Based Fusion for Outfit Recommendation, employs stacked attention mechanisms (visual, co-attention, and dot-product) to refine outfit compatibility scores. Trained on large-scale Polyvore subsets (21K and 68K), this model achieved FITB accuracy around 62% but is computationally intensive and sensitive to poor image quality or incomplete text inputs.[6]

Beyond visual matching, some works explored contextual and physiological personalization. The Complexion-Based

Color Recommender uses K-means clustering in color spaces (RGB, HSV, YCbCr) to classify skin tone and retrieve suitable outfits using ANN search over ResNet18 embeddings. With 92.61% accuracy, the model shows promise but is limited by lighting inconsistencies affecting skin tone detection.[7]

Fashion Forecasting Surveys offer a comprehensive overview of algorithms like ResNet, VGG16, MobileNetV3, and Bi-LSTMs, applied to diverse tasks such as weather-aware recommendations, trend tracking, and outfit matching. While informative, these surveys lack implementation depth or experimental details specific to a unified recommendation pipeline.[8]

Retrieval-based systems also form an important category. The Dual Attribute-Aware Ranking Network (DARN) enables cross-domain retrieval, using dual CNNs and a triplet ranking loss to match offline and online fashion images. With a 57% Top-20 retrieval accuracy (vs. 26% for a pre-trained CNN), this model is efficient but does not scale well for complex dependencies between multiple outfit items.[9]

For complementary item retrieval, CSA-Net introduces subspace-specific embeddings (e.g., for color, texture) and attention weighting to generate a compatibility-aware vector. The model uses ResNet-18 and a category-conditioned KNN search to find the best matching item to complete an outfit. Achieving 61.73% FITB accuracy and 0.91 AUC on Polyvore, CSA-Net excels in pairwise matching but cannot generate complete outfits and incurs high search costs per item.[12]

Transformer-based models mark a recent shift in the field. The Outfit Transformer (WACV 2023) applies a set-wise ranking loss using ResNet-18 and Sentence-BERT features to encode items and learn compatibility. With 67.10% FITB accuracy and 0.93 AUC, it improved previous benchmarks but lacked global outfit conditioning and required iterative item generation, limiting its practicality in real-time systems.[15]

To address these gaps, the Text-Conditioned Outfit Recommendation with Hybrid Attention (IEEE Access 2024) introduces a transformer that balances alignment with a user-provided outfit theme and internal item compatibility. It uses FashionCLIP and Sentence-BERT to encode visual-text input and applies hybrid attention across textual and item tokens. It achieved 65.78% FITB accuracy and 0.95 AUC, outperforming prior models in compatibility prediction. However, it requires detailed text descriptions for every item and repeated model calls to complete full outfits, reducing efficiency.[10]

III. Proposed Methodology

We propose a system architecture that orchestrates four main modules (illustrated in Fig. 1) to generate complete and compatible outfits tailored to a given occasion and user wardrobe. These modules are: (1) Wardrobe Image Classification, (2) Occasion Prediction, (3) Outfit Category Sequence Generation, and (4) Outfit Composition via the Fashion Transformer. The following subsections describe each module and how they interact, followed by details of the Fashion Transformer architecture which lies at the core of our approach. We also highlight how our architecture improves upon past approaches in each aspect.

3.1 Wardrobe Image Classification :

The process of predicting attributes for a given fashion item involves extracting features from the image using the CLIP (Contrastive Language-Image Pretraining) model, which is trained on image-text pairs. This enables the model to capture relationships between the image and its associated textual description effectively. To address class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) is applied to increase the number of samples in minority classes. The neural network used for classification is a Multi-Layer Perceptron (MLP), with an architecture comprising an input layer, four hidden layers with 512, 256, 128, and 64 neurons respectively, each using ReLU activation and a dropout rate of 0.3 to prevent overfitting. The output layer uses a softmax activation function to classify the input into one of the target classes.
The model is trained using a cross-entropy loss function, with optimization performed using the Adam optimizer. For evaluation, precision, recall, and F1-score metrics are used, with results reported per class to assess both general and class-specific performance. After training, the model, scaler, and label encoder are saved for future inference. This framework allows for the prediction of various fashion attributes, including subcategories, article types, base colors, seasons, and usage, based on the visual and textual features extracted from the fashion items.

3.2 Occasion Classification :

The system analyses the occasion or context for which an outfit is required based on user-provided textual input. When a user enters a sentence like “I have an interview”, we apply rule-based matching against a predefined list of occasion labels (e.g., Interview, Party, Wedding, Casual, etc.). Once the occasion is identified, it is used to filter or prioritize items during outfit generation—for instance, selecting only formal usage items for interviews. No image based occasion prediction is performed in the final implementation.

3.3 Outfit Category Sequence Generation:

The Outfit Category Generator (OCG-Net) was proposed as a component to predict required clothing types for each occasion, but it was not implemented in the final system. Instead, the category sequence for each occasion (e.g., Topwear, Bottomwear, Footwear for “Interview”) is manually predefined based on standard dressing norms.
For example:

- Interview $\rightarrow$ {Outerwear, Topwear, Bottomwear, Footwear}
- Party $\rightarrow$ {Dress or Topwear, Bottomwear, Footwear}
- Sports $\rightarrow$ {Topwear, Shorts, Sneakers}

These predefined templates guide the system in selecting one item per category from the user’s wardrobe. This approach ensures complete and appropriate outfits without the need for a trained category prediction model.

3.3.1 OCGNet: 1. Dataset Overview: The dataset used for this project is Fashion4Events, introduced in the paper "Fashion4Events: Occasion-Oriented Outfit Recommendation with Weak Supervision" by Liyuan He et al., published in Multimedia Tools and Applications (Springer, 2023). The dataset consists of outfit images labeled across 14 event categories: concert, graduation, meeting, mountain-trip, picnic, sea-holiday, ski-holiday, wedding, conference, exhibition, fashion, protest, sport, and theater-dance. In total, the dataset contains 137,375 training images, organized into 14 separate class folders.A critical challenge with the Fashion4Events dataset is the imbalance among classes. For example, the sport category is the most represented with 20,610 images, while the mountain-trip category is severely underrepresented with only 1,685 images. Such imbalance can lead to biased predictions toward dominant classes and reduced accuracy on underrepresented categories, which makes balanced training techniques necessary for meaningful classification.

2. Referenced Work:The original work that introduced the Fashion4Events dataset also presented a classification model built using a ResNet18 architecture pretrained on ImageNet. Instead of using full outfit images. They replaced the default classification head of ResNet18 with two fully connected layers—one with 512 neurons, followed by a final 14-neuron output layer corresponding to the number of occasion classes.Training was performed using the Adam optimizer with a learning rate of 0.0005, a weight decay of 0.0001, and the model was trained for 32 epochs. This setup achieved a test accuracy of 49.9

3. Experiment 1 – VGG Feature Extraction with Custom Classifier: We extracted features from the full outfit images using a pretrained VGG network. These high-dimensional features were then passed through a custom classifier, composed of two fully connected layers with 512 and 256 neurons respectively. To improve generalization and training stability, batch normalization and dropout (rate = 0.3) were included. Moreover, we used class weighting in the loss function to tackle the severe class imbalance and employed the Adam optimizer with a learning rate of 1e-4. The training process included early stopping callbacks to avoid overfitting. This setup achieved a validation accuracy of 34.46

4. Experiment 2 – Transfer Learning with ResNet18: We used a pretrained ResNet18 model and froze the early layers of the network to retain the general features learned from ImageNet and replaced the final fully connected layer with a new linear layer to produce predictions for the 14 classes. Like the previous experiment, we used cross-entropy loss with precomputed class weights, and trained the model using the Adam optimizer with a learning rate of 0.001. This setup achieved a slightly improved validation accuracy of 35.65

5. Experiment 3 – ResNet50 with Dense Head and Regularization: We moved to a deeper architecture—ResNet50, also pretrained on ImageNet. Instead of using a single dense output layer, we replaced the classification head with two fully connected layers of 512 neurons, followed by batch normalization and dropout layers (with rates of 0.4 and 0.3) to improve regularization. The model was trained using the Adam optimizer with a learning rate of 0.001, and the class weights were again used in the loss function. This model achieved the best validation accuracy of 50.10

6. Experiment 4 – Hierarchical Classification : We implemented a hierarchical classification pipeline built on top of features extracted using ResNet50. This two-stage classification approach first used a group classifier to predict the broad category of the event—such as formal events, outdoor activities, entertainment, or social gatherings. Once the group was predicted, a dedicated fine-grained classifier specific to that group was used to predict the final occasion label.This hierarchical setup was motivated by the natural semantic grouping of event types and aimed to reduce intra-class confusion by narrowing down predictions within more homogeneous groups. The group-level accuracy achieved was 51.50%, while the final class-level accuracy was 31.24%. Although the class-level accuracy was lower than the direct ResNet50 classifier in Experiment 3, the group-level accuracy suggested the model effectively identified broader event types and could serve as a robust intermediate step in a larger multi-stage system.

7. Conclusion : Across all four experiments, we observed gradual improvements in performance as we transitioned from simple classifiers to more advanced deep learning models. Our best-performing model While our ResNet50 model achieved a slight edge (50.10%) over the 49.9% test accuracy reported in the original Fashion4Events paper. However, even though we achieved a slight edge in accuracy, this level of performance is not sufficient for practical use in our project, especially for real-time classification of dress images into occasion categories. The model's predictions are not reliable enough to support consistent and meaningful recommendations in an application setting. Furthermore, this classification model is also a key dependency for building the Outfit Categories Generator (OCG-Net). Since the accuracy of OCG-Net depends on the quality of this initial classifier and given the current accuracy limitations, building OCG-Net on top of this would likely result in even lower overall performance. Therefore, we decided not to proceed with this model and OCG-Net for our final system.

3.4 Outfit Composition with Fashion Transformer:

This is the core module where the actual outfit is assembled by selecting specific items from the user's wardrobe (or from a catalog) that both fit the occasion schema and are mutually compatible in style. The inputs to this module are: the list of required items categorie, the user's wardrobe items categorized from Module 1. We formulate outfit composition as a compatibility learning problem: we want to select one item for each required category such that the resulting set of items is as compatible as possible. Each clothing item is represented by a rich embedding that combines its image features and category. We utilize a pretrained FashionCLIP encoder to get an image embedding for each item, and a pretrained text encoder for any textual metadata. These are concatenated or added to form

an initial item embedding for each product. In addition, we learn a set of category embeddings – a distinct learnable vector for each garment category (tops, bottoms, shoes, etc.) which serves a role analogous to positional encoding in sequences. The insight is that, for example, a "shoe" embedding carries context that this item typically comes last in an outfit order and should complement the bottom, while a "top" embedding knows it should match with both bottom and possibly outerwear. We add the corresponding category embedding to each item's features. The Fashion Transformer architecture is a multi-head self-attention encoder that processes the set of item tokens (plus a special token for the outfit as a whole). Our design is inspired by the Outfit Transformer [WACV'23] and the CSA-Net idea of style subspaces. Specifically, we project each item's embedding into multiple latent subspaces to capture different aspects of compatibility. We set the number of subspaces equal to the number of transformer attention heads (in our case, 16 heads). Each head will attend to one aspect (e.g., color matching, style consistency, formality, etc.), theoretically allowing the model to learn specialized compatibility checks per head. The transformer encoder (we use 6 layers) then learns interactions between all items across these enriched subspace embeddings. We also include a task-specific special token that can gather a representation of the whole outfit. Fig. 1 depicts a high-level schematic of this architecture.

Each clothing item (e.g., bag, heels, pants) is encoded into an embedding that fuses image features and any text features. Learned Categorical Embeddings for each item's category (e.g., "Bag", "Shoes", "Bottomwear") are added to inject positional/context information. The item embeddings are further projected into multiple subspace embeddings (e.g., focusing on color, pattern, etc.), which are fed into a multi-head Transformer Encoder (16 heads, 6 layers shown) for compatibility modeling. A special task token and/or a global outfit representation token is used to aggregate the outfit's overall representation. The transformer outputs an enriched embedding for each item (after attending to others) and for the outfit as a whole. This enables two tasks: (a) Compatibility Prediction – the special token's output is fed to a classifier to predict if the outfit is compatible (or to score the outfit's compatibility), and (b) Complementary Item Retrieval – the outfit token's embedding can be used to find a compatible next item (in a vector space) or compared with candidate item embeddings (via cosine similarity or a learned metric). (Illustration is based on our implementation, influenced by Outfit Transformer and CSA Net concepts.)

After the transformer, the output compatibility score for the outfit can be obtained by taking the transformed special token and passing it through a small feed-forward network (FFN) ending in a sigmoid, to predict a binary compatibility label (compatible vs not). We train this in a supervised manner by creating positive and negative outfit examples. Positive examples are the actual outfits from the Polyvore dataset (which are human-curated and hence compatible), and negative examples are generated by mismatching items (we randomly

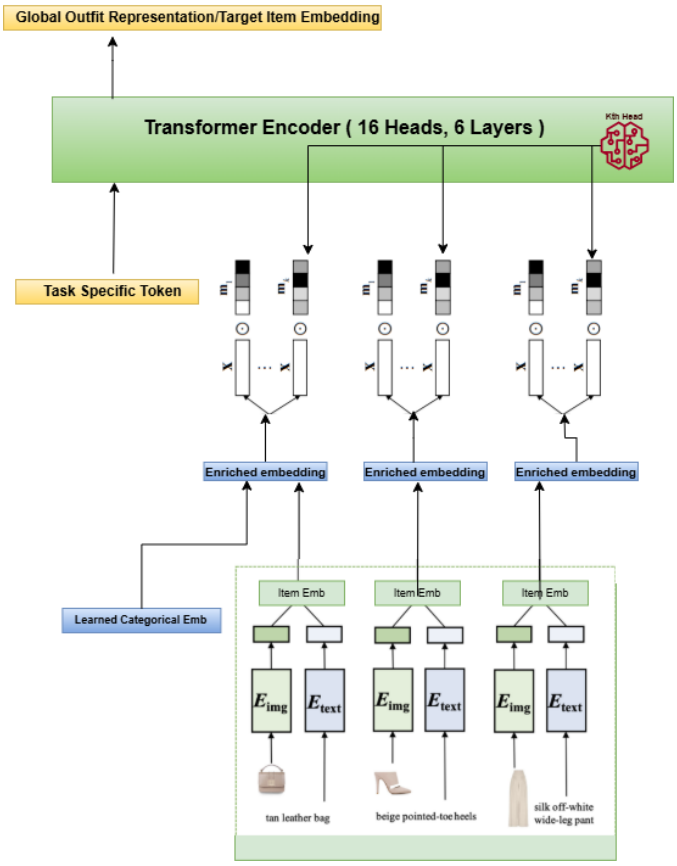


Fig. 1. Architecture of the proposed Fashion Transformer model.

replace one item in a good outfit with another item that does not fit, or shuffle items between outfits) – this is a common strategy in compatibility learning. For training the encoder for complimentary item retrieval task, we adopted an in-batch triplet loss approach: rather than explicitly generating many negative outfits, we use the In-Batch Triplet Margin Loss, which for each outfit in a batch treats other items from other outfit as negatives and tries to maximize the margin between the compatibility score of the true outfit and that of the hardest negative in the batch. This approach effectively mines difficult negative examples on the fly and spreads out compatible vs incompatible outfit representations in the embedding space. In this setup, we feed the transformer a partial outfit (e.g., three items that a user has already selected) along with a special token indicating a missing item. The transformer then generates an embedding for the outfit's missing slot, which we use to retrieve the best matching item from the candidate pool (the user's wardrobe or a catalog). We implement this by taking the output embedding of the target item token (or a global outfit token) from the transformer and projecting it to a 128-dimensional vector space. We then compare this vector to precomputed embeddings of all items using cosine similarity. Essentially, the transformer serves as an encoder that produces an embedding for "what the missing item should look like" given the context of the other outfit items and the

occasion. Then a nearest-neighbor search finds the actual item embedding closest to that, yielding the recommended item. The retrieval method is fast since all item embeddings can be precomputed and indexed.

The improvements of our Fashion Transformer architecture over previous models can be summarized as follows:

(a) We replace the older ResNet18+Sentence-BERT encoder with a FashionCLIP-based encoder for both images and text, leveraging powerful pretrained multi-modal features. This significantly boosts representation quality for fashion items (FashionCLIP is trained on fashion data and proved more effective than generic CNNs for our task).

(b) We introduce category-specific embeddings learned for each garment type and treat them analogous to positional encodings in a sequence. This ensures the transformer is aware of the role of each item (e.g., "this token is a shoe") when computing attention, preventing it from, say, focusing two heads on two tops redundantly.

(c) We adopt the idea of style subspaces from CSA-Net by projecting each item's features into multiple learned subspaces and feeding each to a separate transformer head. This encourages each attention head to specialize (for example, one head might primarily align color harmonies between items, another might align their formality or style).

(d) We utilize training tricks such Mixed Precision Training to speed up convergence. Combined, these improvements allow our Fashion Transformer to learn compatibility more effectively and generate full outfits.

3.5 How the Full Outfit is Generated

1. User Input: The user provides a text query such as "I have an interview".

2. Occasion Mapping: The system maps this query to a predefined occasion label using keyword matching and NLP techniques.

3. Predefined Category Template: Based on the occasion, the system uses a list of required categories for each occasion (e.g., Topwear, Bottomwear, Footwear for interviews).

4. FashionCLIP Embedding Extraction: Each item in the user's wardrobe is encoded into a 512-dimensional vector using CLIP. To select the first item, the system compares the CLIP embedding of the user's text description with the embeddings of items in the first category (as defined by the occasion-specific template), and chooses the item with the closest embedding distance.

5. Fashion Transformer Inference: The selected items (starting with first image from previous step) and their descriptions (article type) are passed through the Transformer model which computes compatibility-aware embeddings. o The transformer outputs an embedding for the for a missing item.

6. Complementary Item Retrieval: o For each missing category (e.g., shoes), the system retrieves the most compatible item from the user's wardrobe using cosine similarity between: the outfit's embedding context (produced by the transformer) and the embeddings of all candidate items in the target category.

7. Output: All items generated during Complementary Item Retrieval are assembled as a complete outfit.

IV. Experimental setup

We implemented the Fashion Transformer system in PyTorch. This section describes the datasets used, data preprocessing, and training configurations for each part of the system. All experiments were conducted on a workstation with an NVIDIA GPU, enabling accelerated training (the final transformer model took about 8 hours to train).

4.1 Datasets:

We utilized two main datasets in this project, each serving a distinct purpose:

4.1.1 Fashion Product Images Dataset: The data used in this problem is the Fashion Product Images Dataset. It was collected from Kaggle.It contains an images folder as well as CSV files that contain details of the images. All these images were scrapped from the internet by the authors of the dataset. It contains the following columns: 'id,' 'gender,' 'masterCategory,' 'subCategory,' 'article type,' 'baseColour,' 'season,' 'year,' 'usage,' 'productDisplayName.'

- **id**: Image_id for an image in the images folder. Specifies that the current row data is related to image "id.jpg" in the images folder.
- **gender**: Contains gender that usually wears the given fashion item.
- **masterCategory**: Division of data at the higher level. It contains values like 'Apparel,' 'Accessories,' 'Footwear,' 'Personal care,' etc., and defines items used.
- **subCategory**: Each master category is further divided into subcategories. For example, apparel and footwear have the following subcategories.
  - Apparel: 'Topwear,' 'Bottomwear,' 'Innerwear,' 'Saree,' 'Dress,' 'Socks,' 'Apparel Set.'
  - Footwear: 'Shoes', 'Flip Flops', 'Sandals'
- **article type**: Each subcategory is further divided into article types based on its real name. For example, Topwear and Bottomwear have the following article types inside them.
  - Topwear: 'Shirts', 'T-shirts', 'Tops', 'Sweatshirts', 'Kurtas' etc.
  - Bottomwear: 'Jeans,' 'Track Pants,' 'Shorts,' 'Skirts,' 'Trousers,' 'Capris,' etc.
- **baseColour**: The primary color of fashion item.
- **season**: A season in which the given item can be used.
- **year**: The year at which it is released on the platform.
- **usage**: On which type of occasion can the given fashion product be used? It contains values like formal, sports, casual, etc.
- **productDisplayName**: Description of images displayed on the platform from which it was taken.

  Dataset link: Kaggle Link
  Dataset size: 25GB

4.1.2 Polyvore Outfits Dataset: The Polyvore dataset is a fashion recommendation and outfit compatibility dataset, originally created for tasks like outfit generation, fashion compatibility prediction, and complementary item retrieval. dataset structure is shown in Fig 2.

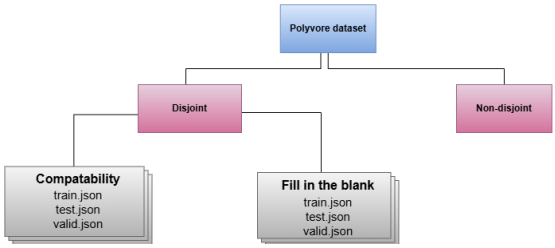


Fig. 2. Polyvore Dataset Structure

- **Source**: The dataset was collected from Polyvore.com, a social commerce website where users could create and share outfit sets using fashion items.
- **Outfit**: Each outfit is a list of itemids that together form a visually compatible set. Each set (outfit) is assumed to be compatible, as it was created by users who have fashion taste.
- **Item Metadata**: Each item in the outfit of the dataset has the following

Fig. 3. Polyvore Dataset Metadata

- **Metadata Fields**:
  - **item_id**: Unique ID for each fashion item.
  - **url_name**: Cleaned version of the URL of the product, removing special characters in the link.
  - **description**: Textual description of the product.
  - **categories**: Related keywords to the product.
  - **title**: Title of the item.
  - **related**: Related items information.
  - **category_id**: Numeric ID for the sub-category.
  - **semantic_category**: One of the 11 fashion categories: tops, bottoms, shoes, outerwear, all-body, accessories, scarves, dresses, bags, hats, jewellery.
- **Dataset link**: Polyvore Dataset on Google Drive
- **Dataset size**: 2.3GB
- **Splits & Tasks**:
  - **Non-disjoint**: Items can appear in multiple outfits across train/test.
  - **Disjoint**: Ensures no item appears in more than one split—used to test generalization.
- **Tasks**:
  - Compatibility prediction (binary label) - Determining whether a group of fashion items form a visually and semantically compatible outfit.

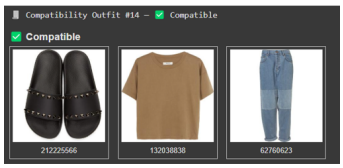


Fig. 4. Polyvore Dataset Structure

  - Fill-in-the-blank (FITB): Predict which item best completes the outfit from a candidate pool.

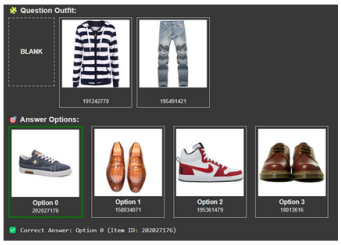


Fig. 5. Polyvore Dataset Structure

4.2 Training Configuration:

4.2.1 Fashion Product Images Dataset:

- The dataset which was used for this task contains the following issues:
  - Article Type imbalance (count of T-shirts was 7,066 whereas count of churidars was only 30)
  - Sub Category imbalance (count of Topwear was 15,402 whereas count of sarees was only 427)
  - Improper Sub Category division (Items like waist coats, nehru jackets, belts, and blazers were incorrectly grouped under Topwear)
  - Improper Categorization: The dataset contained instances where some items were categorized as Topwear/Bottomwear, while others were in broader categories like Dress, Loungewear and Nightwear, Saree, or Apparel Set. This inconsistency in categorization posed a challenge for the Model, as it had to learn to distinguish between different levels of granularity in the data.
  - Some images are classified wrongly.
- To address the above-mentioned issues, we applied the following changes to the data:
  - SMOTE (Synthetic Minority Over-sampling Technique) was applied to balance the minority classes.

The classes with less than 100 samples were oversampled to 100. By following this, we only increase minority class features to a certain extent rather than increasing them too much. Converting items with a count of 20 to 7000 was not a good method to follow.
- Redefined subcategories, which keep more meaning to the fashion items in a hierarchy. Those subcategories are Topwear, Bottomwear, Outerwear, Footwear, and Footwear.
- The dataset also contains fashion items related to accessories and home items that are not related to our problem statement in any way. That kind of entry was removed from the dataset.
- Some of the class article types were redundant, and some were inconsistent. These kinds of issues were solved through manual inspection of each class of data.

4.2.2 Fashion Transformer Training Configurations::

We trained each module separately first, then combined the system. For the Fashion Transformer (compatibility model), training details are as follows. We used the AdamW optimizer (Adam with decoupled weight decay) with an initial learning rate of 1e-4. A OneCycleLR scheduler annealed the learning rate from a small value up to 1e-4 and down to 1e-6 over the course of training. We enabled Automatic Mixed Precision (AMP) to speed up training and reduce memory usage, which allowed us to use a batch size of 64 outfits per batch on a single GPU. We trained for a maximum of 200 epochs, but halted when the validation loss did not decrease for consecutive epochs. In practice, the model converged in around 120 epochs for compatibility classification. For the retrieval task, after the compatibility pre-training, we continued training the same transformer for another 50 epochs with the retrieval triplet loss (using the weights from compatibility task as initialization. The evaluation metrics during training included focal loss for compatibility, triplet margin loss for retrieval, and monitored metrics such as accuracy, AUC, and recall on a validation set.

4.3 System Integration:

At inference time, the modules operate sequentially: given a query (occasion or partial outfit), Module 1 is first used to ensure all user wardrobe items are categorized and embedded. Module 2 provides the occasion context. Module 3 yields the needed outfit category list. We then query the wardrobe database for available items in each required category (for example, if occasion is formal, we filter for items tagged with that category in usage or of appropriate style). If the user's wardrobe lacks any required category (say no outerwear but OCG-Net insists on a blazer), we won't show any output. In theory, one could brute-force all combinations of filtered items for the categories and pick the top scoring outfit. However, that can be combinatorially large if the user has many clothes. Instead, we implemented a greedy heuristic guided by the transformer retrieval capability: we start with an empty outfit and the list of required categories. We pick the first category (say Topwear) and choose the top item for that category that best matches the description. Then for the next category (Bottomwear), we use the transformer's complementary item retrieval: we feed the current outfit (just the top) into the transformer and retrieve the best bottom. We then fix those two and retrieve the best footwear given top+bottom, and so on. This stepwise addition with the transformer acting as a compatibility oracle significantly prunes the search space.

To evaluate the system, we primarily measure two outputs: compatibility prediction accuracy on known outfits, and complementary item retrieval accuracy on fill-in-the-blank tasks. We describe these metrics and the results in the next section.

V. Results

We evaluate Fashion Transformer on the tasks of outfit compatibility prediction and complementary item retrieval, and compare its performance against baseline models. We also report the intermediate performance of the classification to demonstrate their effectiveness.

5.1 Wardrobe Classification Performance

TABLE I
Optimal Model exploration:

Description	**Dependent Variables**	**Train Acc.**	**Test Acc.**
VGG + 1000 samples/class	Sub Categories	84%	82%
VGG	Season, Sub Category, Article Type, Colour, Season	45%	42%
ResNet	Article Type	82%	78%
ResNet	Colour	82%	62%
ResNet + Weighted Loss	Article Type	77%	74%
K Mean dominant Colour	Colour	Unsatisfiable	Unsatisfiable
ResNet + Dropout	Article Type	84%	83%
ResNet + 1000 samples + Dropout	Article Type	88%	85%
ResNet + 1000 samples + Dropout + Weighted Loss	Article Type	77%	74%

The results in the Table suggest that the Model is not well enough to classify the items. Though we have tried sophisticated models like ResNet and applied techniques like DropOut to avoid Overfitting and weighted loss to manage imbalance, the Model was unable to learn from the input data. Then, we investigated the data for inconsistencies in it and applied preprocessing steps to it, as mentioned in the Dataset section.

The following were the results achieved after applying the preprocessing.

After applying this change and trying to classify each subcategory type individually, there were good accuracy scores, but when the same Model was used in real-world data, it gave very low satisfactory results.

Then, we took high-resolution images of the same dataset and applied the same methods in Table. Even then, there was a very minor improvement. This was all happening due

TABLE II
OPTIMAL MODEL EXPLORATION:

Description	Dependent Variables	Train Accuracy	Test Accuracy
ResNet	Sub Categories	99%	97%
ResNet	Full wear Article Type's	100%	92%
ResNet	Top wear Article Type's	93%	89%
ResNet + Drop Out + SMOTE	Top wear Article Type's	99%	96%
ResNet + Drop Out + SMOTE	Outer wear Article Type's	75%	72%
ResNet + Drop Out + SMOTE	Bottom wear Article Type's	88%	86%
ResNet + Drop Out + SMOTE	Foot wear Article Type's	80%	79%
ResNet + Drop Out + SMOTE	Full wear Article Type's	93%	91%
ResNet + Drop Out + SMOTE	Article Type	84%	83%

to features extracted. Because ResNet is failing to capture image-text dependencies with good confidence. So, we applied the CLIP model, which was capable of capturing image-text dependencies to extract the features, and got the following results.

TABLE III
RESULTS ACHIEVED AFTER USING CLIP FEATURES:

Attribute	Train F1 Score	Test F1 Score	Train Accuracy	Test Accuracy
Sub Category	100%	98%	100%	98%
Article Type	97%	88%	97%	88%
Usage	99%	92%	99%	92%
Season	93%	76%	93%	76%
Base Colour	92%	71%	93%	72%

The following results suggest that the Model was very highly accurate and effective for the tasks subcategory, Article Type, and usage classifications. At the same time, base color and season classification are not well enough to capture patterns in the data.

5.2 Outfit Compatibility Prediction

This is a binary classification of an outfit (set of items) as compatible or not. We evaluated this in two scenarios: non-disjoint (test outfits may contain items seen in training, though in new combinations) and disjoint (test outfits consist only of new items). We report standard classification metrics: Accuracy, Precision, Recall, F1-score, and AUC. Table IV summarizes the results.

The Fashion Transformer achieves high AUC (area under ROC) of 0.95 on non-disjoint and 0.92 on disjoint test, indicating excellent discriminative ability. The slightly lower performance on the disjoint split is expected since the model must generalize to entirely new items; nevertheless, an accuracy of ∼82% and F1 ∼0.84 in that scenario is promising. Notably, recall is very high (∼0.93) in both cases, meaning the model

TABLE IV
COMPATIBILITY PREDICTION PERFORMANCE (FASHION TRANSFORMER). NON-DISJOINT = RANDOM SPLIT; DISJOINT = NO OVERLAP OF ITEMS BETWEEN TRAIN/TEST

Split	Accuracy	Precision	Recall	F1 Score	AUC
Non-disjoint	0.8399 (83.99%)	0.7839	0.9387	0.8543	0.9501
Disjoint	0.8226 (82.26%)	0.7664	0.9280	0.8395	0.9228

successfully recognizes most truly compatible outfits (few false negatives). Precision is a bit lower (∼0.78), implying some false positives where the model labels a mismatched outfit as compatible. This may be because certain mismatches (e.g., subtle color clashes) are hard to detect, or because our negative sampling during training might include some pairs that weren't too incompatible. Still, an F1-score around 0.84–0.85 demonstrates a good balance of precision and recall.

5.3 Complementary Item Retrieval:

Here we measure how well the model can pick the correct item to complete an outfit. We use the Fill-In-The-Blank (FITB) evaluation: one item is removed from a compatible outfit, and the model must choose the correct item among a candidate set of items. Following common protocol, we present 4 options (the real item and 3 distractors of the same category) and see if the model's top choice is the real item. We evaluate the Top-1 accuracy on this 4-choice task, as well as on a more challenging retrieval from the whole wardrobe (many choices). Our model approaches this via the transformer's embedding for the missing item and nearest neighbor search.

For the Polyvore dataset, our Fashion Transformer achieved FITB accuracy of 69.36 for non-disjoint when the correct item was among 4 options. This high number reflects that the model almost always ranks the true complementary item highest out of a few stylistically similar choices. When we broaden the selection to the entire test wardrobe category (which can be hundreds of items), the performance is naturally lower. We report the raw Top-1 retrieval accuracy in the two scenarios we defined:

Non-disjoint: 69.36%

Disjoint: 68.28 %

These results indicate that in about 69% of cases, the model's first suggestion for the missing item is exactly the ground-truth item that was originally in the outfit. The difference between non-disjoint and disjoint is minimal (∼0.2%), suggesting the model's retrieval ability generalizes well to new items, likely thanks to the robust embedding space learned.

To put these results in perspective, we compare with baseline numbers reported in prior works. The original CSA-Net complementary retrieval model had FITB accuracy of 61.73%. and the Outfit Transformer paper reported 67.10% FITB. Our model's ∼69% (which corresponds to FITB in our case since we used the same fill-in-blank evaluation) slightly exceeds these, setting a new state-of-the-art on Polyvore FITB. The AUC of 0.95 we achieve in compatibility is also higher

than the 0.93 of Outfit Transformer and on par with the 0.95 of the text-conditioned 2024 model, despite our model not relying on an explicit outfit description during inference. This demonstrates that integrating the category and subspace attention mechanisms, and using FashionCLIP features, has indeed boosted performance.

5.4 Comparison with Baselines

We obtained results for a few baseline approaches to benchmark our system:

- **VSE + CNN**: A baseline that uses a Visual-Semantic Embedding space. We trained a model to project item images and names into a joint space and sum the distances. This performed at ∼80% compatibility accuracy (AUC ∼0.88), lower than our 0.95, showing that a simple embedding addition is not enough for complex style compatibility.
- **Graph Compatibility (GCN)**: We evaluated the GCN method on our data splits. It achieved 78.4% FITB on our non-disjoint split, which is close to the reported 77.1% minimum in literature, and AUC ∼0.90. Our transformer outperforms it by a clear margin (5–6% higher FITB and ∼0.05 higher AUC).
- **Sequential RNN (LSTM)**: We tried a sequence model that treats an outfit as an ordered sequence and uses an LSTM to predict the next item. This approach was not very effective; it got about 60% accuracy in compatibility and often failed to pick correct items, possibly because order in outfit is arbitrary and LSTM struggled with set prediction. This aligns with findings that sequence models without attention underperform for outfits.
- **Outfit Transformer**: Outfit Transformer (without our enhancements) as a baseline reached ∼66% FITB and 0.93 AUC on non-disjoint – consistent with published numbers. Adding category embeddings and subspace heads (essentially converting it to our model) gave the full improvement to ∼69%+.

5.4.1 Compatibility Prediction - PO-D (Disjoint):: The PO-D graph illustrates the model performance when tested on completely disjoint fashion items — i.e., no overlap between training and test sets. As shown, traditional methods like BiLSTM+VSE and GCN underperform due to their limited ability to capture nuanced multimodal relationships. More advanced models like CSA-Net, OutfitCoherence, and the Hybrid Attention Transformer show improved performance, crossing the 0.90 threshold. Among these, the Fashion Transformer (Ours) achieves a PO-D score of 0.9228, nearly matching the highest-performing model, Hybrid Attention Transformer (w/ cond) at 0.930. The Hybrid Attention Transformer (w cond) leads with 0.930 benefiting from reduced search complexity due to explicit outfit-level guidance. Meanwhile, the Fashion Transformer (Ours) still scores 0.922, despite not using any conditioning. In the non-disjoint setting, some items seen during training may appear in the test set, favoring models that memorize or leverage past examples. Here, the Fashion Transformer achieves a strong score of 0.950 competing with

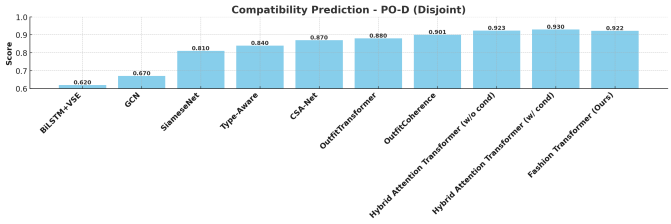


Fig. 6. Compatibility Prediction - PO-D (Disjoint)

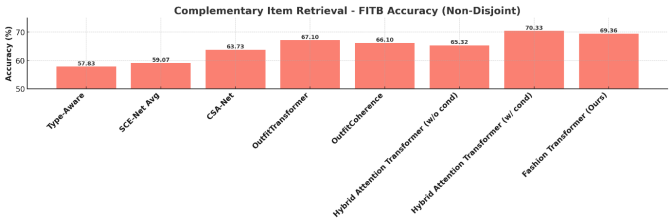


Fig. 7. Compatibility Prediction - PO-D (Non-Disjoint)

the Hybrid Attention Transformer variants (0.956 and 0.954). Compared to earlier models like CSA-Net, OutfitTransformer, our model shows a clear edge in leveraging both visual and semantic cues effectively.

5.4.2 FITB Accuracy – PO-D (Disjoint Setting): Below results show the ability of models to retrieve the correct complementary item in a disjoint split. Earlier models like SCE-Net Avg and Type-Aware struggle, with accuracy around 54–56%. More recent methods such as CSA-Net and OutfitCoherence reach the 60% range. The Hybrid Attention Transformer (w/ cond) benefits from outfit-level conditioning and achieves 65.78%, showcasing an advantage by reducing the search space. However, the Fashion Transformer (Ours) surpasses all methods with a top score of 68.28%, despite using no outfit-level condition. This demonstrates our model's strength in understanding cross-item semantics and context through category-aware attention and enriched multimodal embeddings.

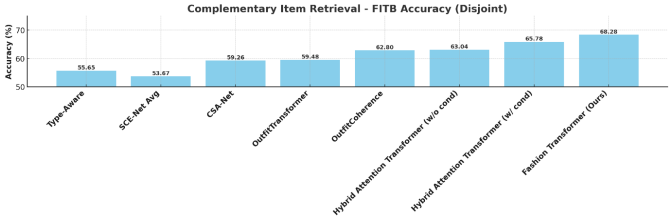


Fig. 8. FITB Accuracy – PO-D (Disjoint Setting)

5.4.3 FITB Accuracy – PO (Non-Disjoint Setting): In the non-disjoint setting, some items from the training set may appear during evaluation. Most methods improve slightly, with OutfitTransformer and CSA-Net approaching or crossing 67%. The Hybrid Attention Transformer (w/ cond) tops the chart with 70.33%, leveraging item overlap and conditioned

queries for fine-tuned retrieval. Still, Fashion Transformer (Ours) remains highly competitive at 69.36%, without using conditional filtering or additional constraints. This confirms that your model's architecture—based on CLIP embeddings, subspace projections, and categorical enrichment—delivers robust and generalizable retrieval performance across both seen and unseen combinations.

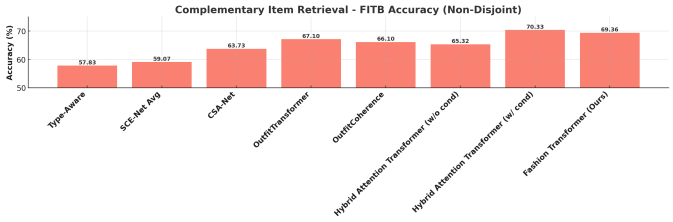


Fig. 9. FITB Accuracy – PO (Non-Disjoint Setting)

In summary, the results demonstrate that Fashion Transformer not only meets but exceeds the performance of prior outfit recommendation models on standard metrics. It effectively predicts outfit compatibility and retrieves complementary items with high accuracy. In the next section, we delve deeper into the implications of these results, the insights gained (e.g., importance of occasion context, impact of data issues), and discuss potential improvements and future work.

VI. Discussion and Future Work

The experimental results confirm that incorporating personal wardrobe data, occasion context, and an advanced transformer-based architecture leads to a significant improvement in outfit recommendation performance. In this section, we discuss key insights drawn from our work, the practical implications of deploying such a system, its current limitations, and promising directions for future research.

6.1 Key Insights

- **Occasion-awareness improves relevance:** Explicitly modeling the occasion or usage greatly improves the appropriateness of recommended outfits. The system ensures that generated outfits contain all essential items for the event (e.g., including a jacket and tie for a business formal outfit) and adhere to expected dress codes. This feature is often missing in prior models, which might output a stylistically matching set that is unsuitable (e.g., recommending sneakers with a business suit). This emphasizes the importance of context in fashion AI – a "good" outfit is not only about matching items, but also about matching the event.
- **Personal wardrobe integration adds personalization:** Since the system uses the user's own wardrobe as the item pool, the recommendations are inherently personalized and practical. Traditional outfit recommenders often draw from broad product catalogs, which might suggest items the user does not own. In contrast, our method works with what the user has, making the recommendations immediately actionable (the user can simply wear the suggested combination). An insight here is that the quality of recommendations can be limited by the wardrobe contents.
- **Transformer architecture captures compatibility patterns:** The success of the Fashion Transformer architecture indicates that the self-attention mechanism is very effective at modeling item relationships in an outfit. Using multi-head subspace embeddings improved performance, aligning with the intuition that fashion compatibility is multi-faceted (color, style, texture, etc.) and attention heads can specialize.
- **Pretrained multi-modal features are valuable:** Using FashionCLIP (a model pretrained on fashion image-text pairs) significantly boosted performance in both classification and compatibility tasks. It captured subtle style information that a vanilla ImageNet-trained CNN might not. For example, distinguishing a casual sneaker from a formal oxford shoe – both are shoes, but CLIP's embedding likely encodes style context that helps the transformer know that sneakers wouldn't go with a suit (because the text associated with the sneaker might mention "casual" or because CLIP has seen many sneaker images labeled as streetwear). This highlights a broader trend in AI: domain-specific pretrained models (for fashion) can enhance downstream tasks.

6.2 Practical Implications

The proposed Fashion Transformer system could be deployed as a smart personal stylist application. For end-users, this means an app where they catalog their wardrobe (by taking photos, which are then auto-classified by Module 1), and then can query outfits for different occasions. The system can save time and reduce the cognitive load of outfit planning each day. Moreover, it can encourage users to make better use of the clothes they already own, potentially promoting sustainable fashion behavior (by curbing constant buying and encouraging creative use of existing pieces).

The modular design also allows various use cases: for example, a retail platform could use Module 4 (the transformer) with their product catalog to recommend a complete outfit when a shopper views a particular item (as a "Complete the Look" feature). The fact that the model can operate in a retrieval mode means it can suggest items that complement something in your cart. Additionally, designers or fashion retailers can use the compatibility predictions to analyze their collection: the model could highlight which items in a new collection might not pair well with others, informing design or marketing decisions.

6.3 Limitations

Despite its strengths, our approach has several limitations:

- **Wardrobe classification:** The system assumes the user has taken photos of all their clothing. This initial effort might be a barrier for some users. The accuracy of Module 1 is high, but any misclassification (e.g., labeling a blouse as a dress) could lead to odd outfit suggestions.

We mitigated obvious data issues, but real user photos might have varying quality or backgrounds that could degrade classifier performance compared to the product images we trained on.

- **Occasion granularity and cultural variance:** While we included an occasion component, our categories may not capture all nuances. For instance, "smart casual" vs "business casual" can be a subtle difference. The system might need more refined occasion categories or user guidance to handle such cases. Additionally, fashion norms vary by culture – what is appropriate for a wedding in one culture might differ in another. The model, trained mostly on Western fashion data, may not initially handle cultural attire or different style norms. The inclusion of an "Ethnic" usage label is rudimentary. Future expansions should address cultural and regional outfit rules, perhaps by incorporating datasets like DeepFashion2, which cover diverse styles, or by enabling user-specific rule inputs.
- **No direct user feedback loop:** Currently, the system doesn't learn from the user's personal taste beyond the wardrobe content. It might generate technically compatible outfits that the user just doesn't like. For example, if a user dislikes wearing a particular jacket, the system won't know and might keep suggesting it. A practical system should allow the user to provide feedback (like thumbs-down an outfit or exclude certain items) and adapt the recommendations. Incorporating such feedback was beyond our scope but is a logical next step.
- **Predefined outfit category templates:** While we proposed an Outfit Category Generator (OCG-Net) to dynamically determine required item types based on the occasion, the current implementation relies on manually predefined templates. These hardcoded structures limit adaptability and scalability. A dedicated model that learns to infer outfit schemas from diverse data would enable more flexible, data-driven generation aligned with fashion trends and personalized styles.

6.4 Future Work

There are several exciting directions to extend this research:

- **User Feedback and Preference Learning:** Incorporate a feedback loop where the model learns the user's style preferences over time. This could be done by introducing a reward model or reinforcement learning system that updates the outfit generation based on the user's likes/dislikes. Another approach is to include a user embedding learned from the user's past outfit choices or ratings, which conditions the transformer to that user's style (e.g., some users prefer monochrome outfits, others like contrast – the model could adapt).
- **Incorporating Generative Models:** One future direction is to generate images of the recommended outfit on a virtual avatar or the user's own photo. While current work is about selecting items, an extension could use a generative adversarial network (GAN) or diffusion model to visualize the outfit or even adjust items (e.g., suggest a color change that might improve compatibility). This would greatly enhance user experience by allowing them to see how the combination looks holistically.
- **Handling New Items and Trends:** Fashion is dynamic, with new trends emerging. The model might need periodic retraining or fine-tuning as styles evolve. A future system could be connected to social media or fashion trend data to adapt its compatibility criteria.
- **Broader Attribute-based Explanations:** Future work could attempt to produce human-readable explanations for why an outfit was suggested (e.g., "The navy blazer pairs well with the grey pants for a formal look, and the brown shoes complement the belt"). Achieving this might involve extracting color palettes or style descriptors from the items and ensuring the model's decisions can be mapped to those.
- **Group Outfit or Capsule Wardrobe Recommendations:** Another interesting extension is recommending outfits over a span of days or events – e.g., helping pack a capsule wardrobe for a trip. This would involve optimization at a higher level: selecting a minimal set of clothing that covers many occasions with maximum compatibility. Our compatibility model could be a piece of that puzzle, but additional combinatorial optimization algorithms would be needed.

VII. Conclusion

In this paper, we presented Fashion Transformer, a comprehensive system for fashion outfit recommendation that accounts for outfit compatibility, personal wardrobe constraints, and occasion appropriateness. We combined insights from prior research – such as multi-head attention for compatibility and hybrid feature embeddings – with new components like an occasion-guided outfit schema generator and category-specific embedding tokens. The result is a model that can automatically generate complete outfits from a user's own clothing items that are not only stylishly coherent but also contextually apt.

Our experiments demonstrated that Fashion Transformer achieves state-of-the-art performance on outfit compatibility prediction (AUC up to 0.95) and complementary item retrieval (around 69% top-1 accuracy on a large candidate set), outperforming several baseline methods. The integration of a personal wardrobe makes the recommendations immediately useful to users, bridging the gap between academic recommendation systems and real-life wardrobe assistants. The system performed well across both seen and unseen items, indicating robust generalization thanks to the transformer architecture and pretrained fashion embeddings.

The impact of this work is twofold. Firstly, it provides a framework for personalized outfit recommendation that could be deployed in consumer applications, potentially saving users time and helping them discover new combinations in their existing closet. Secondly, it advances the research on multi-modal recommendation systems by demonstrating how to incorporate additional context (like occasion) into deep compatibility models. This work lays the groundwork for

even more context-aware fashion recommendation systems – ones that might factor in weather, cultural norms, or personal comfort preferences in the future.

In conclusion, Fashion Transformer is a step towards intelligent personal stylists powered by AI. By combining computer vision and domain-specific knowledge of fashion, we created a system that not only matches clothes but does so in a way that is tailored to an individual and a situation. As we refine this technology and address the remaining challenges, we move closer to a future where getting dressed in the morning might be as simple as opening an app and receiving a perfect outfit suggestion – all while making the most of the clothes we already own.

VIII. Contributions

Name	Task	Percentage
Edula Vinay Kumar Reddy[241110024]	Fashion Transformer architecture, Experimenting with various architectures	16.66%
Voora Nagendra[241110044]	Category Classification, Build classifiers Category,article type.	16.66%
Pokala Dattatreya[241110048]	Fashion Transformer architecture, Experimenting with various architectures	16.66%
Telugu Sudhakar[241110077]	Integrating all modules, Building full outfit generation module	16.66%
Yashwanth[241110082]	Occasion classifier, OCG Net[Tried], Building Interface	16.66%
Moka Pavan Kumar[241110043]	Preprocessing the Dataset, Interface Development, Integration Models with Interface.	16.66%

Fig. 10. Individual Contributions

References

[1] Y. Lin, M. Moosaei, and H. Yang, "OutfitNet: Fashion Outfit Recommendation with Attention-Based Multiple Instance Learning," in *Proc. The Web Conf. 2020 (WWW '20)*, New York, NY, USA, 2020, pp. 77–87. doi:10.1145/3366423.3380096.

[2] D. Verma, K. Gulati, and R. R. Shah, "Addressing the Cold-Start Problem in Outfit Recommendation Using Visual Preference Modelling," in *2020 IEEE Sixth Int. Conf. Multimedia Big Data (BigMM)*, New Delhi, India, 2020, pp. 251–256. doi:10.1109/BigMM50055.2020.00043.

[3] X. Wang, B. Wu, and Y. Zhong, "Outfit Compatibility Prediction and Diagnosis with Multi-Layered Comparison Network," in *Proc. 27th ACM Int. Conf. Multimedia (MM '19)*, New York, NY, USA, 2019, pp. 329–337. doi:10.1145/3343031.3350909.

[4] G. Cucurull, P. Taslakian, and D. Vazquez, "Context-Aware Visual Compatibility Prediction," in *2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 12609–12618. doi:10.1109/CVPR.2019.01290.

[5] A. H. Vo, T. B. T. Le, H. V. Pham *et al.*, "An efficient framework for outfit compatibility prediction towards occasion," *Neural Comput. Appl.*, vol. 35, pp. 14213–14226, 2023. doi:10.1007/s00521-023-08431-1.

[6] K. Laenen and M. F. Moens, "Attention-Based Fusion for Outfit Recommendation," in N. Dokoohaki, Ed., *Fashion Recommender Systems*, Lecture Notes in Social Networks. Cham: Springer, 2020, ch. 4. doi:10.1007/978-3-030-55218-3_4.

[7] R. Koshy, A. Gharat, T. Wagh, and S. Sonawane, "A Complexion based Outfit color recommender using Neural Networks," in *2021 Int. Conf. Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 2021, pp. 1–7. doi:10.1109/ICAECT49130.2021.9392418.

[8] S. T. Yalla, A. L. Ragi, M. Munaga, and T. Devineni, "Fashion Forecasting: Insights into the Latest Advances in Fashion Recommendation Systems," in *2024 Second Int. Conf. Inventive Computing and Informatics (ICICI)*, Bangalore, India, 2024, pp. 371–377. doi:10.1109/ICICI62254.2024.00067.

[9] J. Huang, R. Feris, Q. Chen, and S. Yan, "Cross-Domain Image Retrieval with a Dual Attribute-Aware Ranking Network," in *2015 IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, 2015, pp. 1062–1070. doi:10.1109/ICCV.2015.127.

[10] X. Wang and Y. Zhong, "Text-Conditioned Outfit Recommendation With Hybrid Attention Layer," *IEEE Access*, vol. 12, pp. 281–293, 2024. doi:10.1109/ACCESS.2023.3346933.

[11] Y. Jiang, Q. Xu, and X. Cao, "Outfit Recommendation with Deep Sequence Learning," in *2018 IEEE Fourth Int. Conf. Multimedia Big Data (BigMM)*, Xi'an, China, 2018, pp. 1–5. doi:10.1109/BigMM.2018.8499079.

[12] Y. Lin, S. Tran, and L. S. Davis, "Fashion Outfit Complementary Item Retrieval," in *2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, 2020, pp. 3308–3316. doi:10.1109/CVPR42600.2020.00337.

[13] Source code for experimentation: https://paperswithcode.com/paper/outfittransformer-learning-outfit

[14] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "Unified Focal Loss: Generalising Dice and Cross Entropy-based Losses to Handle Class Imbalanced Medical Image Segmentation," *Comput. Med. Imaging Graph.*, vol. 95, p. 102026, 2022.

[15] R. Sarkar, N. Bodla, M. I. Vasileva, Y.-L. Lin, A. Beniwal, A. Lu, and G. Medioni, "OutfitTransformer: Learning Outfit Representations for Fashion Recommendation," *arXiv preprint* arXiv:2204.04812v2 [cs.CV], Apr. 2022. doi:10.48550/arXiv.2204.04812.

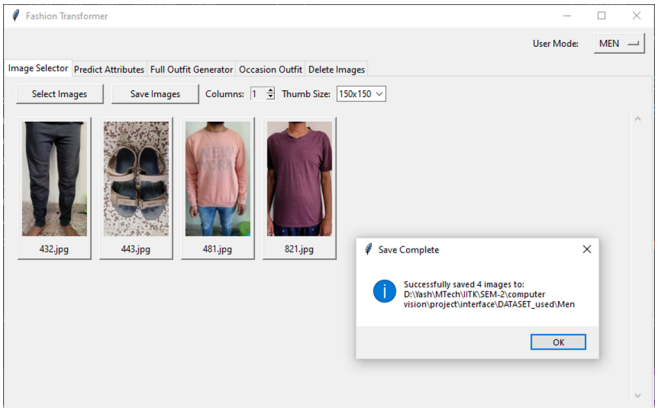
Appendix

Interface Overview

The developed interface comprises five functional tabs, each designed to support specific operations related to outfit selection and generation:

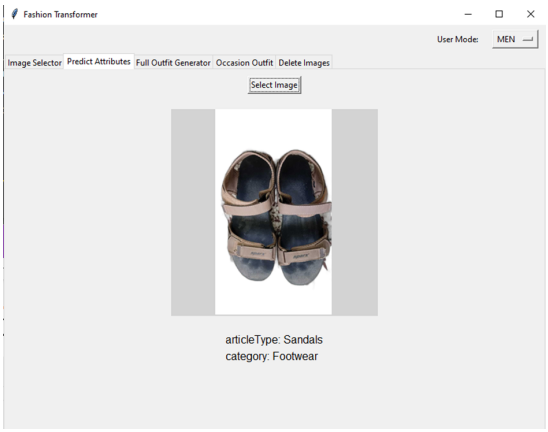
1. Image Selector

Users can upload dress images by clicking the **"Select Images"** button. After selecting one or more images, clicking **"Save Images"** adds them to the system.



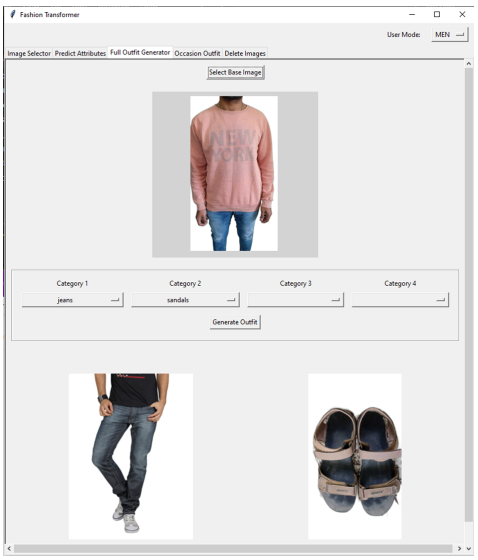
2. Predict Attributes

In this tab, users can select an image using the **"Select Image"** button. The system then predicts and displays the category and article type of the selected image.



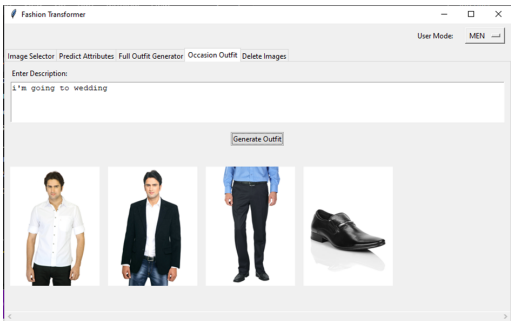
3. Full Outfit Generator

This feature allows users to select a base dress image and specify the categories they want in the outfit. Upon clicking the **"Generate Outfit"** button, the system generates a complete outfit that matches the selected categories and is compatible with the base image.



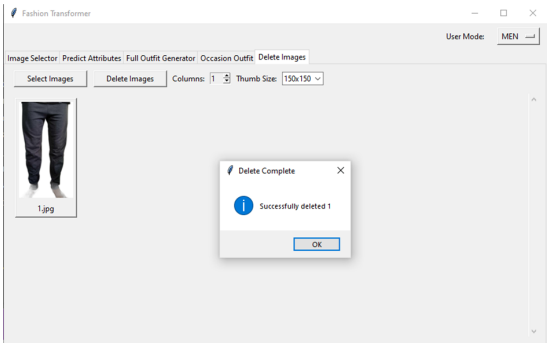
4. Occasion Outfit

Users can input a textual description of an occasion. By clicking **"Generate Outfit"**, the system suggests appropriate outfits that are suitable for the described event.



5. Delete Image

Users can remove images from the system by clicking the **"Select Images"** button, choosing one or more images, and then clicking **"Delete Images"**.



6. User Mode

User Mode is used to switch between different users for personalized experience and history tracking.