GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

Transforming Fashion AI for Style Recommendation

Autor: Lara Ocón Madrid
Director: Andrés Occhipinti Liberman
Co-Director: Simón Rodríguez Santana

Madrid
Junio de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Transforming Fashion: AI for Style Recommendation

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.
El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Lara Ocón Madrid Fecha: 10 / 06 / 2025

Autorizada la entrega del proyecto
EL DIRECTOR DEL PROYECTO

Fdo.: Andrés Occhipinti Liberman Fecha: 10 / 06 / 2025

Agradecimientos

A mi familia, a mis amigos, y a mis directores de TFG, Andrés y Simón, por todo su apoyo, dedicación y acompañamiento durante todo este proceso.

Transformando la Moda: IA para la Recomendación de Estilo

Autor: Ocón Madrid, Lara
Director: Occhipinti Liberman, Andrés
Codirector: Rodríguez Santana, Simón
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

Resumen Numerosas industrias se han beneficiado de la IA, y la industria de la moda no es una excepción. Al comprender qué hace que un conjunto de ropa sea "bueno", las empresas pueden ofrecer recomendaciones personalizadas. Este proyecto avanza en la recomendación de moda utilizando Transformers [7] para predecir si las prendas combinan bien en un conjunto y para completar conjuntos parciales. Exploramos la estrategia de Curriculum Learning (CL) [7], donde el modelo se entrena gradualmente con ejemplos cada vez más difíciles. Nuestra contribución estudia una variación de esta estrategia de entrenamiento: seleccionar ejemplos más difíciles mediante clustering [17] y probar cómo esta elección afecta el aprendizaje. También analizamos la naturaleza subjetiva de la tarea a través de una encuesta [18] y un análisis cualitativo.
Palabras clave: Recomendación Automática de Moda, Transformers, Clustering, Curriculum Learning

1. Introducción

Con el crecimiento exponencial de los productos de moda disponibles, los sistemas de recomendación automatizados se han vuelto esenciales para mejorar la experiencia del cliente. En este proyecto, seguimos el modelo Transformer de Sarkar (2018), que logra rendimiento "estado-del-arte" en dos tareas clave: **Compatibility-Prediction (CP)**, donde el modelo aprende a evaluar si un conjunto de prendas forma un conjunto coherente, y **Fill-in-the-Blank (FITB)**, donde completa un conjunto seleccionando la prenda más adecuada dados varios candidatos.

Comenzamos entrenando el modelo en CP, ya que esta tarea es esencial para que el modelo desarrolle una noción de compatibilidad estilística. Este modelo se utiliza para inicializar FITB, una tarea más compleja en la que, dado un conjunto incompleto, el modelo debe distinguir el artículo correcto (positivo) de varias prendas incompatibles (negativos), aprendiendo a acercar los positivos y alejar los negativos en su espacio de representación interna. Para guiar este proceso, aplicamos **Curriculum Learning (CL)**, aumentando progresivamente la dificultad de las muestras negativas. Cuando los negativos son demasiado similares al positivo, pueden confundir al modelo, debilitando la señal de supervisión.

Nuestra contribución es una estrategia de muestreo negativo mediante **Clustering No Supervisado**. La idea es dividir el catálogo de prendas en clusters de manera que los artículos dentro de un mismo cluster sean compatibles en los mismos conjuntos. Dada esta partición, seleccionamos negativos de un cluster diferente al del artículo positivo, reduciendo la probabilidad de muestrear un artículo compatible como negativo. Probamos dos técnicas de clustering: K-Means, que captura la similitud visual entre artículos, y **Procesos de Dirichlet Jerárquicos (HDP)**, que capturan patrones estilísticos a partir de datos textuales. Con K-Means y CL estándar, logramos un rendimiento estado-del-arte, mientras que HDP supera el baseline sin CL. Estos resultados sugieren que combinar la similitud visual de K-Means con los patrones estilísticos de HDP tiene un gran potencial para mejorar aún más el rendimiento del modelo.

2. Definición del Proyecto

Este Trabajo de Fin de Grado (TFG) se basa en el modelo Outfit-Transformer [7], incorporando los embeddings Fashion-CLIP [10] para mejorar la eficiencia computacional y las representaciones multimodales. El objetivo del proyecto es replicar el rendimiento del modelo en Predicción de Compatibilidad (CP) y Fill-in-the-Blank (FITB), y explorar mejoras en la estrategia de "curriculum learning" utilizada en FITB mediante clustering no supervisado. Específicamente, introducimos K-Means y Procesos Dirichlet Jerárquicos (HDP) para guiar el muestreo negativo, con el fin de evitar la selección de candidatos igualmente compatibles como negativos y reducir así la ambigüedad del entrenamiento.

Los objetivos de este trabajo son (1) revisar el estado del arte en recomendación de moda, (2) replicar el modelo Outfit-Transformer [7], y adoptar los embeddings Fashion-CLIP de Oh et al. [10] para mejorar la eficiencia, (3) comparar la estrategia original de "curriculum learning" con nuestra mejora propuesta basada en clustering no supervisado, y (4) evaluar el rendimiento del modelo y explorar la subjetividad de la compatibilidad de moda a través de una encuesta de usuarios que compara sus opiniones humanas con las predicciones del modelo.

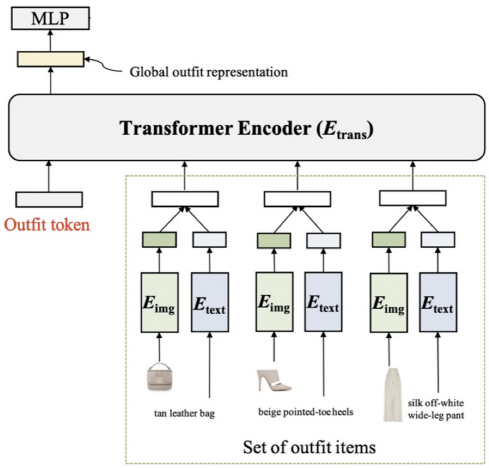

Figure 1: Arquitectura del OutfitTransformer [7].

3. Descripción del Modelo

El modelo OutfitCLIPTransformer procesa conjuntos de ropa como secuencias no ordenadas de prendas, puesto que la compatibilidad es independiente del orden de las prendas. Cada prenda está representada por un Fashion-CLIP embedding de 1024 dimensiones que combina sus características visuales y textuales del dataset Polyvore. El modelo transforma estos embeddings en una representación global del conjunto para las tareas de CP y FITB. Su arquitectura, mostrada en la Figura 1, consta de: (1) una capa de proyección lineal que reduce los embeddings de 1024 a 128 dimensiones; (2) un token CLS aprendible que se antepone a la secuencia para aprender información del conjunto; (3) un encoder con cuatro capas y cuatro cabezas de atención, que capturan las interacciones entre las prendas; y (4) la extracción del embedding del token CLS para tareas posteriores. Para CP, una cabeza de clasificación genera un logit de compatibilidad. Para FITB, se calculan las distancias euclídeas entre el embedding del conjunto incompleto y los embeddings de los artículos candidatos para clasificar el artículo correcto.

CP se plantea como un problema de clasificación binaria y se entrena utilizando la pérdida "Focal Loss" con un parámetro de enfoque $\gamma = 2$, que enfatiza los ejemplos difíciles. Dado el puntaje de compatibilidad predicho $p \in [0, 1]$ y la etiqueta $y \in \{0, 1\}$, la loss es:

$$\mathcal{L}_{\text{focal}}(p, y) = -\alpha_t y(1-p)^\gamma \log(p) - (1-\alpha_t)(1-y)p^\gamma \log(1-p),$$

donde $\alpha_t = 0.5$. El entrenamiento se realiza durante 200 épocas usando AdamW con mini-batches de outfits.

FITB tiene como objetivo seleccionar el artículo correcto para completar un conjunto incompleto. El modelo se entrena utilizando una "triplet margin loss", que fomenta que el embedding del conjunto incompleto z_q (consulta) esté más cerca de la prenda correcta z_p (positivo) que de cualquier candidato incorrecto z_{n_i} (negativo). La función de pérdida se define como:

$$\mathcal{L}_{\text{triplet}} = \frac{1}{B}\sum_{j=1}^{B} \max(0, d(z_{q_j}, z_{p_j}) - \min_{i \neq j} d(z_{q_j}, z_{n_i}) + \mu),$$

donde $d(\cdot, \cdot)$ es la distancia euclídea, $\mu = 2.0$, y B es el tamaño del batch. Empleamos CL para aumentar progresivamente la dificultad de las prendas negativas: comenzando con negativos aleatorios, luego muestreando desde la misma categoría que el artículo correcto, y finalmente desde la misma subcategoría [7]. Nuestra contribución refina esta última fase mediante el muestreo de negativos de diferentes clusters dentro de la misma subcategoría, abordando la alta similitud entre los artículos observada en el dataset que puede confundir la triplet loss. Exploramos dos métodos de clustering: K-Means, aplicado a los embeddings Fashion-CLIP para agrupar prendas por similitud visual, y Procesos Dirichlet Jerárquicos (HDP), utilizando descripciones textuales de las prendas para modelar la co-ocurrencia y capturar patrones estilísticos.

4. Resultados

Predicción de Compatibilidad (CP): El OutfitCLIPTransformer alcanzó un AUC de **0.95**, superando el 0.93 del OutfitTransformer original [7] y coincidiendo con la implementación de Oh [10], como se muestra en la Tabla 1. Con un umbral de 0.5, la precisión fue baja (61.86%) debido a un sesgo hacia la predicción de compatibilidad, pero al elevar la frontera de decicisión a 0.6, la precisión mejoró a 87.94%. Algunas predicciones incorrectas revelaron un desacuerdo entre las etiquetas del dataset y la compatibilidad percibida. En particular, el ejemplo mostrado en la Figura 2—etiquetado como compatible pero asignado por el modelo con un puntaje bajo (0.4891)—motivó la creación de una encuesta a pequeña escala para evaluar el juicio humano. Recogimos respuestas de 36 participantes sobre 4 de estos casos límite, y en este ejemplo, el 86% de los encuestados consideraron el conjunto incompatible [18]. Esto confirmó la naturaleza subjetiva de la moda y las limitaciones de la etiquetación rígida binaria.

Modelo	CP
OutfitTransformer (Paper)	0.93
OutfitCLIPTransformer (Open Source)	0.95
OutfitCLIPTransformer (Nuestra Impl.)	**0.95**

Table 1: Comparación AUC en Compatibility Prediction. Tanto nuestro modelo como el Open Source utilizan los embeddings Fashion-CLIP

Figure 2: Ejemplo de falso negativo: el modelo predijo una probabilidad de 0.4891, indicando incompatibilidad, mientras que la etiqueta verdadera era compatible. En una encuesta, el 86% de los encuestados pensaron que era incompatible.

Fill-in-the-Blank (FITB): El rendimiento de FITB se resume en la Tabla 2. Nuestros resultados parten de la implementación Open Source de Oh et al.[10], que ya mejora el 0.67 de accuracy del artículo original[7] a 0.69 utilizando los embeddings Fashion-CLIP. Nuestro baseline sin CL alcanza un 0.65, pero al aplicar CL estándar, logramos alcanzar estado del arte, con un accuracy de 0.72.

Nuestros métodos propuestos de muestreo basado en clustering también obtienen buenos resultados. K-Means alcanza esta precisión de estado del arte (0.72), mientras que HDP logra 0.68, ambos superando el

baseline sin CL. Esto es una señal prometedora, pues sugiere que un refinamiento de los clusters, podría mejorar aún más el rendimiento.

El análisis cualitativo reveló limitaciones en el dataset: varios candidatos de FITB eran igualmente compatibles o incluso idénticos, pero las etiquetas solo consideraban uno como correcto, limitando la capacidad de la métrica para reflejar el verdadero rendimiento del modelo.

Variante del Modelo	**FITB**
OutfitTransformer (Paper, CL)	0.67
OutfitCLIPTransformer (Oh et al.)	0.69
OutfitCLIPTransformer (No CL)	0.65
OutfitCLIPTransformer (CL Estándar)	**0.72**
OutfitCLIPTransformer (K-Means CL)	**0.72**
OutfitCLIPTransformer (HDP CL)	0.68

Table 2: Comparación de precisión FITB para diferentes variantes del modelo.

Figure 3: Ejemplo de predicción incorrecta del modelo para FITB, donde varios candidatos son igualmente válidos/idénticos.

K-Means generó clusters visualmente coherentes, pero pasó por alto la compatibilidad. Confiar únicamente en la similitud visual puede llevar a muestrear negativos que aún son compatibles con el conjunto a pesar de ser visualmente distintos, como se muestra en la Figura 4. HDP capturó patrones estilísticos, aunque perdió esa cohesión visual dentro de los clusters. Esto revela la necesidad de una técnica de clustering que combine las fortalezas de ambos enfoques: coherencia visual y compatibilidad semántica.

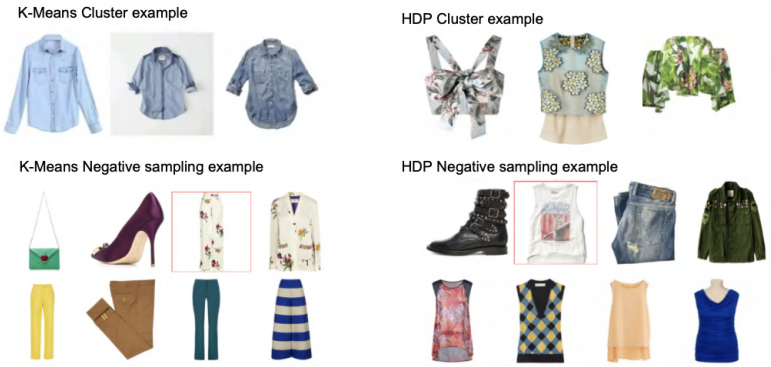

Figure 4: Ejemplos de clustering con K-Means y HDP para muestreo negativo. Fila superior: conjuntos incompletos con el artículo positivo (cuadro rojo) eliminado. Fila inferior: cuatro negativos muestreados para cada caso.

5. Conclusiones

Este trabajo demuestra el potencial de utilizar la IA en la industria de la moda, particularmente a través del uso de Transformers combinados con embeddings Fashion-CLIP y curriculum learning. Al integrar los embeddings CLIP, mejoramos significativamente la eficiencia del entrenamiento y el rendimiento del modelo en tareas como Predicción de Compatibilidad (CP) y Fill-in-the-Blank (FITB). La estrategia de curriculum learning, que aumenta progresivamente la dificultad del muestreo negativo, mejoró aún más estos resultados.

Durante nuestro análisis, identificamos que la presencia de prendas casi idénticos dentro del dataset podría confundir al modelo durante el muestreo negativo. El clustering con K-Means ayudó a organizar los artículos en clusters visualmente similares basados en los embeddings CLIP, pero no consideró cómo interactúan las prendas en los conjuntos. Experimentamos con Procesos Dirichlet Jerárquicos (HDP), que capturaron más patrones estilísticos a través de los datos textuales, pero carecieron de la riqueza visual de los embeddings CLIP, lo que resultó en clusters menos visualmente coherentes. K-Means alcanzó un accuracy de 0.72, igualando el CL estándar, mientras que HDP alcanzó 0.68, superando la línea base sin CL. Estos resultados sugieren que las técnicas de clustering mejoran el rendimiento, y su refinamiento podría superar el estado del arte.

Otra observación clave fue la subjetividad de la tarea. En CP, algunas etiquetas del dataset no coincidieron con las respuestas de una encuesta humana, lo que resalta el desafío de definir la compatibilidad de manera rígida. En FITB, varios candidatos eran igualmente válidos, pero el dataset solo consideraba uno como correcto, lo que limitó la capacidad de reflejar el rendimiento del modelo. Esta subjetividad complica tanto el entrenamiento como la evaluación, revelando que las etiquetas rígidas no siempre son apropiadas para las tareas de moda.

Trabajo futuro: Los esfuerzos futuros se centrarán en refinar las técnicas de clustering, como explorar modelos híbridos que combinen las fortalezas de K-Means y HDP para mejorar el muestreo negativo. También buscamos desarrollar métricas de evaluación que reflejen mejor la naturaleza subjetiva de la moda.

Transforming Fashion: AI for Style Recommendation

Author: Ocón Madrid, Lara
Director: Occhipinti Liberman, Andrés
Co-Director: Rodríguez Santana, Simón
Collaborating Entity: ICAI – Universidad Pontificia Comillas

Abstract Numerous industries have benefited from AI, and the fashion industry is no exception. By understanding what makes a "good" outfit, companies can offer tailored recommendations. This project advances fashion recommendation using transformers[7] to predict whether items of clothing go well together in an outfit and to complete partial outfits. We explore curriculum learning[7], where the model is gradually trained on increasingly difficult examples. Our contribution studies a variation of this training strategy: selecting harder examples through clustering[17] and testing how this choice affects learning. We also analyze the subjective nature of the task through a survey [18] and qualitative analysis.

Keywords: Automated Fashion Recommendation, Transformers, Clustering, Curriculum Learning

1. Introduction

With the exponential growth of available fashion products, automated recommendation systems have become essential to enhance customer experience and engagement. In this project, we follow the transformer-based model proposed by Sarkar et al. [7], which achieves state-of-the-art performance on two core fashion recommendation tasks: **Compatibility Prediction (CP)**, where the model learns to assess whether a set of garments forms a coherent outfit, and **Fill-in-the-Blank (FITB)**, where it completes an incomplete outfit by selecting the most suitable item from a pool of candidates.

We first train the model on CP, as this task is essential for the model to develop a sense of style compatibility. The trained model is then used to initialize FITB, a more complex task in which the model must complete an outfit by distinguishing the correct item (positive) from several incompatible alternatives (negatives), learning to bring positives closer and push negatives away in its internal representation space. To guide this process, the original paper applies **curriculum learning** (CL), gradually increasing the difficulty of the negative samples. While experimenting with CL, we observed that, as difficulty increases, negatives tend to become too similar to the positive (or even valid alternatives) As a result, these "hard" negative examples may end up confusing the model and weakening the training signal.

Our contribution is a negative sampling strategy through **Unsupervised Clustering**. The idea is to partition the catalog of items into clusters so that items within the same cluster are compatible in the same outfits. Given this partition, we select negatives from a cluster different from the positive item's cluster, reducing the likelihood of sampling a compatible item as a negative. We tested two clustering techniques: **K-Means**, which captures visual similarity between items, and **Hierarchical Dirichlet Processes (HDP)**, which captures style patterns from textual data. With K-Means and Standard CL, we achieved state-of-the-art performance, while HDP surpasses the baseline without CL. These results suggest that combining K-Means visual similarity with HDP's stylistic patterns has great potential to further enhance model performance.

2. Project Definition

This Final Degree Project (TFG) builds upon the OutfitTransformer model [7], incorporating Fashion-CLIP embeddings [10] for computational efficiency and improved multimodal representations. The project aims to replicate the model's performance on Compatibility Prediction (CP) and Fill-in-the-Blank (FITB), and to explore improvements to the curriculum learning strategy used in FITB through unsupervised clustering. Specifically, we introduce K-Means and Hierarchical Dirichlet Processes (HDP) to guide negative sampling, with the goal of avoiding the selection of equally compatible candidates as negatives and thus reducing training ambiguity.

The objectives of this work are (1) to review the state-of-the-art in fashion recommendation, (2) replicate the OutfitTransformer model [7], and adopt Fashion-CLIP embeddings from Oh et al. [10] to improve efficiency, (3) to compare the original curriculum learning strategy with our proposed enhancement based on unsupervised clustering, and (4) to evaluate model performance and explore the subjectivity of fashion compatibility through a user survey comparing human judgments to model outputs.

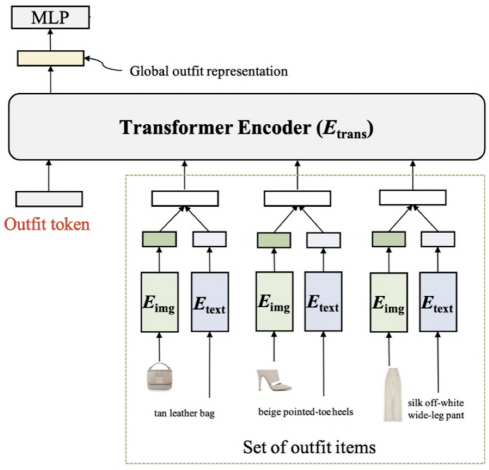

Figure 5: OutfitCLIPTransformer architecture, from [7].

3. Model Description

The OutfitCLIPTransformer model processes outfits as unordered sets of items, as item order does not affect compatibility. Items are represented by a 1024-dimensional Fashion-CLIP embedding that combines visual and textual features from the Polyvore dataset. The model transforms these embeddings into a compact outfit representation for Compatibility Prediction (CP) and Fill-in-the-Blank (FITB) tasks. Its architecture, shown in Figure 5, consists of: (1) a linear projection layer reducing embeddings from 1024 to 128 dimensions; (2) a learnable CLS token prepended to the sequence to learn outfit information; (3) a transformer encoder with four layers and four attention heads, capturing item interactions; and (4) extraction of the CLS token's embedding for downstream tasks. For CP, a classification head outputs a compatibility logit. For FITB, Euclidean distances are computed between the incomplete outfit's embedding and candidate item embeddings to rank the correct item.

CP is framed as binary classification and trained using focal loss with focusing parameter $\gamma = 2$, which emphasizes difficult examples. Given predicted compatibility score $p \in [0, 1]$ and label $y \in \{0, 1\}$, the loss is:

$$\mathcal{L}_{\text{focal}}(p, y) = -\alpha_t y(1-p)^\gamma \log(p) - (1-\alpha_t)(1-y)p^\gamma \log(1-p),$$

where $\alpha_t = 0.5$. Training is performed for 200 epochs using the AdamW optimizer with mini-batches of outfits.

FITB aims to select the correct item to complete an incomplete outfit (where one item has been masked). The model is trained using a triplet margin loss, which encourages the embedding of the incomplete outfit z_q (query) to be closer to the correct item z_p (positive) than to any incorrect candidate z_{n_i} (negative). The loss function is defined as:

$$\mathcal{L}_{\text{triplet}} = \frac{1}{B}\sum_{j=1}^{B} \max(0, d(z_{q_j}, z_{p_j}) - \min_{i \neq j} d(z_{q_j}, z_{n_i}) + \mu),$$

where $d(\cdot, \cdot)$ is Euclidean distance, $\mu = 2.0$, and B is the batch size. We adopt curriculum learning (CL) to progressively increase the difficulty of the negative items: starting with random in-batch negatives, then sampling from the same category as the correct item, and finally from the same subcategory [7]. Our contribution refines this last phase by sampling negatives from different clusters within the same subcategory, addressing high item similarity observed in the dataset that can confuse the triplet loss. We explored two clustering methods: K-Means, applied to Fashion-CLIP embeddings per subcategory to group items by visual similarity, and Hierarchical Dirichlet Processes (HDP), using textual descriptions of outfit items to model co-occurrence and capture stylistic patterns.

4. Results

Compatibility Prediction (CP): The OutfitCLIPTransformer achieved an AUC of **0.95**, surpassing the original OutfitTransformer's 0.93 [7] and matching Oh's implementation [10], as shown in Table 3. Both our model and Oh's use Fashion-CLIP embeddings. At a threshold of 0.5, accuracy was low (61.86%) due to a bias toward predicting compatibility, but raising the threshold to 0.6 improved accuracy to 87.94%. Some incorrect predictions revealed a mismatch between dataset labels and perceived compatibility. In particular, the example shown in Figure 6—was labeled as compatible but assigned by the model a low score (0.4891)—motivated the creation of a small-scale survey to assess human judgment. We collected responses from 36 participants on four such borderline cases, and in this example, 86% of respondents considered the outfit incompatible [18]. This highlights the subjective nature of fashion and the limitations of rigid binary labeling.

Model	CP
OutfitTransformer (Paper)	0.93
OutfitCLIPTransformer (Open Source)	0.95
OutfitCLIPTransformer (Our Impl.)	**0.95**

Table 3: AUC comparison for Compatibility Prediction.

Figure 6: False negative example: the model predicted a 0.4891 probability, indicating incompatibility, whereas the true label was compatible. In a survey, 86% of respondents thought it was incompatible.

Fill-in-the-Blank (FITB): FITB performance is summarized in Table 4. We build upon the open-source implementation by Oh et al.[10], which already improves the original paper's 0.67 accuracy[7] to 0.69 by using Fashion-CLIP embeddings. Our baseline without curriculum learning (CL) achieves 0.65, but applying standard CL to Oh's setup boosts accuracy to 0.72, establishing a new state of the art.

Our proposed clustering-based sampling methods also yield strong results. K-Means matches this state-of-the-art accuracy (0.72), while HDP achieves 0.68, outperforming the no-CL baseline. This is a promising signal, suggesting that clustering can meaningfully guide negative sampling and has room for further refinement.

Qualitative analysis revealed limitations in the dataset: several FITB candidates were equally compatible or even visually identical to the ground truth, yet counted as incorrect, limiting the metric's ability to reflect true performance.

Model Variant	FITB
OutfitTransformer (Paper, CL)	0.67
OutfitCLIPTransformer (Oh et al.)	0.69
OutfitCLIPTransformer (No CL)	0.65
OutfitCLIPTransformer (Standard CL)	**0.72**
OutfitCLIPTransformer (K-Means CL)	**0.72**
OutfitCLIPTransformer (HDP CL)	0.68

Table 4: Comparison of FITB accuracy for different model variants.

Figure 7: Example of incorrect model prediction for FITB, where several candidates are equally valid/identical

K-Means generated visually coherent clusters, but overlooked contextual compatibility. Relying solely on visual similarity can lead to sampling items that still match the outfit despite being visually distinct, as shown in Figure 8. HDP captured stylistic patterns in outfits, though limited by sparse textual metadata, lacking visual cohesion within clusters. This reveals the need for a clustering technique that combines the strengths of both approaches: visual coherence and semantic compatibility.

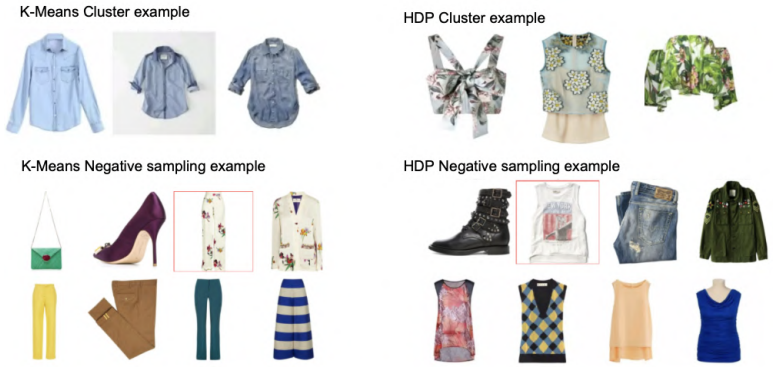

Figure 8: Examples of K-Means and HDP clustering for negative sampling. Top row: incomplete outfits with the positive item (red box) removed. Bottom row: four sampled negatives for each case.

5. Conclusions

This work demonstrates the potential of using in the fashion industry, particularly through the use of Transformers combined with Fashion-CLIP embeddings and curriculum learning. By integrating CLIP embeddings, we significantly improved the efficiency of training and the performance of the model in tasks like Compatibility Prediction (CP) and Fill-in-the-Blank (FITB). The curriculum learning approach, which progressively increased the difficulty of negative sampling, further enhanced these results.

During our analysis, we identified that the presence of nearly identical items within the dataset could confuse the model during negative sampling. K-Means clustering helped organize items into visually similar clusters based on CLIP embeddings but did not consider how items interact in outfits. We experimented with Hierarchical Dirichlet Processes (HDP), which captured more stylistic trends through textual data but lacked the visual richness of CLIP embeddings, resulting in less cohesive clusters. K-Means achieved an accuracy of 0.72, matching standard curriculum learning, while HDP reached 0.68, showing improvement over no curriculum learning. These results suggest that clustering techniques enhance performance, and further refinement could outperform standard curriculum learning.

Another key observation was the subjectivity of the task. In CP, some labels in the dataset did not align with the answers from a human survey, highlighting the challenge of defining compatibility rigidly. In FITB, multiple candidates often appeared equally valid, but the dataset only labeled one of them as correct, leading to confusion. This subjectivity complicates both training and evaluation, revealing that rigid labels are not always appropriate for fashion tasks.

Future Work: Future efforts will focus on refining clustering techniques such as exploring hybrid models that combine the strengths of K-Means and HDP to improve negative sampling. We also aim to develop evaluation metrics that better reflect fashion's subjective nature.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and Context

Fashion has always been one of my greatest passions, but until now it had remained outside the academic sphere of my studies in Artificial Intelligence. This project has allowed me to bring both worlds together by exploring how AI can be applied to fashion, a creative and constantly evolving industry.

Recommender systems play an increasingly important role in the fashion sector, helping users discover items and outfits aligned with their personal style. With the rise of personalized shopping experiences and virtual wardrobes, intelligent outfit recommendation is not only a technical challenge but also a commercial necessity.

1.2. Academic and Technological Relevance

Fashion recommendation systems have evolved from basic collaborative filtering models to sophisticated deep learning architectures. In this context, transformers —originally designed for natural language processing— have shown promising results in modeling item compatibility, especially when garments are treated as tokens in a sequence. This perspective enables the model to capture contextual relationships within outfits.

Integrating fashion and AI reflects not only a personal goal but also a broader trend in the industry, where data-driven insights increasingly influence design, curation, and user interaction. The potential societal and commercial impact of such systems underscores the relevance of this project.

1.3. Objectives

The main objectives of this project are:

- To investigate the current approaches and techniques used in fashion outfit recommendation.
- To study and implement the OutfitTransformer model, understanding its architecture and performance.
- To explore recent adaptations of the model, such as the use of OpenAI's CLIP for multimodal embeddings, as presented in the optimized implementation by Wonjun Oh [10].
- To experiment with the integration of classical machine learning techniques, such as clustering (e.g., k-means), to improve the difficulty progression of negative sampling during training.
- To examine more advanced clustering strategies, such as Dirichlet Processes, as a potential direction for future improvement of compatibility modeling.

1.4. Resources and Feasibility

The implementation has been carried out on a personal MacBook with M2 chip for initial experiments. For larger scale training, access to ICAI's lab infrastructure with GPU support was available when needed. Thanks to the use of precomputed embeddings and efficient model design, the experiments were computationally feasible without requiring significant additional resources.

1.5. Document Structure

The document is structured as follows: Section 2 presents a review of related work and the current state-of-the-art in outfit recommendation. Section 3 details the methodology, model design, and training strategy. Section 4 discusses the experimental setup and performance evaluation. Section 5 presents the results and analysis. Finally, Section 6 provides conclusions and proposes directions for future research.

2. State of the Art

Fashion outfit recommendation has evolved significantly over the past decade, transitioning from sequential models to advanced transformer-based architectures that capture complex item relationships.

2.1. Sequential Modeling with LSTMs

One of the earliest notable contributions is the work by Han et al. [1], which employs a Bidirectional LSTM to model outfits as ordered sequences. Their model is capable of tasks such as Fill-in-the-Blank (FITB), where the model must select the correct item from a set of candidates to complete an incomplete outfit. The model is also capable of outfit generation, learning visual-semantic embeddings from images and textual descriptions. While effective, the LSTM-based design assumes a fixed order of garments, which is not ideal for fashion recommendation where the order of items often does not affect compatibility.

2.2. Pairwise and Graph-Based Compatibility

Subsequent work focused on pairwise compatibility, which involves learning the pairwise compatibility between two items rather than the compatibility of entire outfits. McAuley et al. [2] and Veit et al. [3] determine full outfit compatibility by aggregation (e.g., averaging) over all the pairwise compatibility scores. These approaches struggled with global context. Graph-based methods, such as Cucurull et al. [4] and Cui et al. [5], used graph neural networks to model compatibility as edge predictions, capturing higher-order relationships. However, their reliance on extensive neighborhood data limits applicability for new or sparsely connected items [6].

2.3. Transformer-Based Representations

A major advancement came with the OutfitTransformer by Sarkar et al. [7], which treats outfits as unordered sets, encoding images and text with ResNet and SentenceBERT via a transformer encoder. It excels in several fashion recommendation tasks: Compatibility Prediction (CP), FITB, and Complementary Item Retrieval (CIR) through curriculum learning, outperforming LSTMs and graph-based models on the Polyvore dataset. We describe these tasks in detail in Section 3.

2.4. CLIP-Based Extensions

Wonjun Oh's open-source implementation [10] enhances the OutfitTransformer by integrating CLIP embeddings [8], robust multimodal representations trained on vast image-text pairs. Precomputed CLIP embeddings reduce training time and memory, boosting CP and FITB performance. This underscores the power of vision-language models in fashion recommendation [6].

2.5. Summary and Research Gaps

Fashion outfit recommendation has increasingly adopted transformer-based architectures, which excel at modeling outfits as unordered sets of garments. Unlike sequential models [1], transformers capture compatibility without assuming a fixed item order, leveraging the attention mechanism's core properties: *permutation invariance* (treating outfits as sets) and *global context* (modeling interactions among all items). These qualities, inherent to attention, make transformers, as used in the original paper's OutfitTransformer [7], ideal for tasks like Compatibility Prediction (CP) and Fill-in-the-Blank (FITB), described in Section 3.

Opportunity to Enhance Curriculum Learning For FITB, the original paper [7] employs curriculum learning (CL) to train the model by introducing progressively harder negative samples. Early epochs sample negatives randomly, mid-stage epochs sample from the same category as the positive item, and later epochs draw negatives from the same subcategory as the positive item. Negatives are essential for teaching the model to distinguish compatible from incompatible items. However, we identified a possible opportunity for improvement: sampling negatives from the same subcategory may select items that are visually identical or equally valid, potentially confusing the model. For example, consider a positive item (a black t-shirt) in a subcategory {black t-shirt, black t-shirt with logo, white t-shirt, red t-shirt}. Sampling the black t-shirt with logo as a negative could be suboptimal, as it might be compatible with the outfit, making it harder for the model to learn clear distinctions.

Clustering-Based Negative Sampling To explore this opportunity, we tested clustering-based negative sampling, detailed in Section 3.6.3. Using K-Means, we group items within subcategories by visual similarity based on Fashion-CLIP embeddings [10]. For the example above, K-Means might form three clusters: one for black t-shirts (including the logo variant), one for white t-shirts, and one for red t-shirts. Sampling a negative from a different cluster (e.g., red t-shirt) minimizes the chance of selecting a compatible item. Hierarchical Dirichlet Processes (HDP) [11] cluster items based on textual co-occurrence in outfits, capturing stylistic patterns (e.g., casual vs. formal). As shown in Section 5, these techniques aid model learning, matching the standard CL performance, with potential for further improvement through refined clustering.

Subjectivity in Fashion Compatibility Fashion compatibility is inherently subjective, posing challenges for training and evaluation. To investigate this, we surveyed 36 respondents to evaluate CP examples where the model struggled, comparing human judgments to dataset labels. We discovered significant discordance between human opinions and dataset labels, which impacts model training and metric reliability due to the use of hard labels. In FITB, we observed that multiple candidates are often visually or stylistically valid, yet only one is labeled correct, skewing accuracy metrics, as discussed in Section 5. This subjectivity underscores the limitations of hard labels and the need for evaluation metrics that accommodate multiple valid solutions, such as ranking-based or soft-label approaches.

Our work leverages the GitHub implementation's Fashion-CLIP embeddings [10], which improve efficiency over the original paper's ResNet-18 and SentenceBERT approach [7]. We enhanced FITB training by implementing the original paper's CL and proposed clustering to seize the identified opportunity. Two research gaps guide our study: (1) refining CL with advanced clustering, potentially combining K-Means' visual coherence with HDP's stylistic insights [11]; and (2) developing metrics that reflect fashion's subjectivity to align with human preferences, as explored in Section 5. These gaps are addressed in subsequent sections.

3. Methodology

This section describes the technical formulation and architecture of the system developed for fashion outfit compatibility. The original OutfitTransformer [7] paper defines three core tasks:

1. **Compatibility Prediction (CP)**: A binary classification task that determines whether a given set of fashion items forms a compatible outfit.

2. **Fill-in-the-Blank (FITB)**: A ranking task where the model must select the correct item from a set of candidates to complete an incomplete outfit.

3. **Complementary Item Retrieval (CIR)**: A retrieval task aimed at finding the most compatible item from a large candidate pool to add to a given outfit.

In this project, we focus specifically on the CP and FITB tasks. For FITB, we first experiment with a baseline version without curriculum learning, then implement the approach proposed in the original paper that incorporates curriculum learning. Finally, we extend this strategy by introducing clustering techniques to select better negative samples, aiming to further improve model performance. The CIR task is used during training as described in the original methodology, but evaluation is carried out using the FITB task, as CIR involves ranking a very large pool of candidates, making it both computationally demanding and significantly more complex.

3.1. Problem Formulation

Let $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ denote the set of $N = 251,008$ fashion items in the Polyvore dataset [9], where each item I_i is associated with:

- A unique identifier item_id_i.
- A semantic category $c_i \in \mathcal{C}$ (e.g., "top", "bottom", "shoes").
- A subcategory label $s_i \in \mathcal{S}$ (e.g., "blouse", "sneakers"), encoded as an integer.
- A precomputed Fashion-CLIP embedding $e_i \in \mathbb{R}^{1024}$, combining visual and textual features.
- A cluster label $k_i \in \mathcal{K}$, obtained by applying K-Means to the CLIP embeddings of items in the same subcategory s_i.

An outfit O is a set of k distinct items from $\mathcal{I}$, denoted as $O = \{I_{i_1}, I_{i_2}, \ldots, I_{i_k}\}$, where $i_1, i_2, \ldots, i_k$ are unique indices in $\{1, 2, \ldots, N\}$ and k is typically 3–5.

3.1.1 Compatibility Prediction (CP)

Goal: Predict whether an outfit O is compatible (label $y = 1$) or incompatible ($y = 0$).

Input: An outfit $O = \{I_{i_1}, \ldots, I_{i_k}\}$ with embeddings $\{e_{i_1}, \ldots, e_{i_k}\}$.

Output: A probability $p(y = 1|O) \in [0, 1]$.

Training Data: A dataset $\mathcal{D}_{\text{CP}} = \{(O_j, y_j)\}_{j=1}^{M}$, where $y_j \in \{0, 1\}$ and M is the number of outfits.

3.1.2 Fill-In-The-Blank (FITB)

Goal: Given an incomplete outfit $O' = \{I_{i_1}, \ldots, I_{i_{k-1}}\}$ and four candidate items $\{I_{c_1}, I_{c_2}, I_{c_3}, I_{c_4}\}$, select the correct item I_{c_l} that completes the outfit.

Input: O' with embeddings $\{e_{i_1}, \ldots, e_{i_{k-1}}\}$ and candidates with embeddings $\{e_{c_1}, e_{c_2}, e_{c_3}, e_{c_4}\}$.

Output: The index l of the correct candidate.

Training and Validation Data: For training, we use a dataset defined as follows:

$$\mathcal{D}_{\text{FITB}}^{\text{train}} = \{(O'_j, I_{p_j})\}_{j=1}^{Q}$$

where O'_j is an incomplete outfit (i.e., the original outfit with one item removed), and I_{p_j} is the ground-truth item that completes the outfit. During training, for each (O'_j, I_{p_j}), the model samples negative items to form a set of candidates for contrastive learning. The model encodes O'_j into an outfit embedding $z_{O'}$ and computes embeddings z_{p_j} for the positive item and z_{n_j} for the negatives. The training objective is to minimize the distance between $z_{O'}$ and z_{p_j} while maximizing the distance to the negative embeddings.

For validation, the dataset is define as:

$$\mathcal{D}_{\text{FITB}}^{\text{val}} = \{(O'_j, \{I_{c_{j1}}, I_{c_{j2}}, I_{c_{j3}}, I_{c_{j4}}\}, l_j)\}_{j=1}^{P}$$

where l_j is the index of the correct item among the four candidates. The model generates an embedding $z_{O'}$ for the incomplete outfit and compares it to the candidate embeddings $\{z_{c_{j1}}, z_{c_{j2}}, z_{c_{j3}}, z_{c_{j4}}\}$ using the Euclidean distance to determine whether the correct item is the nearest in embedding space.

3.2. Model Architecture: OutfitCLIPTransformer

The core model, OutfitCLIPTransformer, processes variable-length sequences of item embeddings to produce a fixed-size outfit embedding. Let $O = \{I_{i_1}, \ldots, I_{i_k}\}$ with embeddings $E = [e_{i_1}, \ldots, e_{i_k}] \in \mathbb{R}^{k \times 1024}$. The model is defined as follows:

1. **Projection**: A linear layer $f_{\text{proj}} : \mathbb{R}^{1024} \to \mathbb{R}^d$ maps each embedding to a lower dimension:

$$E' = f_{\text{proj}}(E) = EW_{\text{proj}} + b_{\text{proj}}, \quad E' \in \mathbb{R}^{k \times d},$$

where $W_{\text{proj}} \in \mathbb{R}^{1024 \times d}$, $b_{\text{proj}} \in \mathbb{R}^d$, and $d = 128$.

2. **CLS Token**: A learnable token $\text{CLS} \in \mathbb{R}^{1 \times d}$ is prepended to form:

$$S = [\text{CLS}; E'] \in \mathbb{R}^{(k+1) \times d}.$$

3. **Transformer Encoder**: A transformer encoder with $L = 4$ layers, $H = 4$ attention heads, and dropout $p = 0.1$ processes S:

$$S' = \text{TransformerEncoder}(S, \text{mask}),$$

where $\text{mask} \in \{0, 1\}^{k+1}$ indicates padded positions (0 for valid, 1 for padding).

4. **Output Embedding**: The CLS token embedding is extracted:

$$z = S'[0, :] \in \mathbb{R}^d.$$

This architecture, illustrated in Figure 9, forms the basis for both CP and FITB tasks. For CP, a classification head $f_{\text{cls}} : \mathbb{R}^d \to \mathbb{R}$ produces a logit:

$$z_{\text{logit}} = f_{\text{cls}}(z) = W_{\text{cls}} z + b_{\text{cls}},$$

followed by a sigmoid to obtain $p(y = 1|O) = \sigma(z_{\text{logit}})$. For FITB, the embedding z is used directly for distance computations.

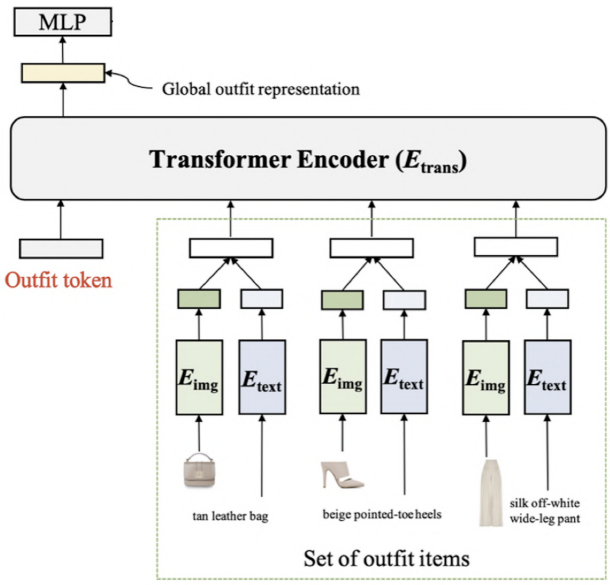

Figure 9: OutfitCLIPTransformer architecture, from original paper [7].

3.3. Compatibility Prediction (CP)

3.3.1 Objective

Minimize the focal loss over $\mathcal{D}_{\text{CP}}$ to learn compatibility:

$$\mathcal{L}_{\text{CP}} = \frac{1}{M} \sum_{j=1}^{M} \mathcal{L}_{\text{focal}}(p(y_j = 1|O_j), y_j),$$

where the focal loss is:

$$\begin{aligned}\mathcal{L}_{\text{focal}}(p, y) = &- \alpha_t y(1-p)^\gamma \log(p) \\ &- (1-\alpha_t)(1-y)p^\gamma \log(1-p),\end{aligned}$$

with $\alpha_t = 0.5$ and focusing parameter $\gamma = 2$.

3.3.2 Algorithm

The model is trained for 200 epochs using AdamW with a learning rate $\eta = 2 \times 10^{-5}$, gradient accumulation over 4 steps, and a OneCycleLR scheduler. For each batch of outfits $\{O_j, y_j\}_{j=1}^{B}$:

1. Compute embeddings $E_j = [e_{j1}, \ldots, e_{jk_j}]$ for each O_j.
2. Pad to max length $k_{\max}$, forming $E \in \mathbb{R}^{B \times k_{\max} \times 1024}$.
3. Apply transformer to get $z_j \in \mathbb{R}^{128}$ for each outfit.
4. Compute logits $z_{\text{logit},j} = f_{\text{cls}}(z_j)$.
5. Calculate loss $\mathcal{L}_{\text{batch}} = \frac{1}{B} \sum_{j=1}^{B} \mathcal{L}_{\text{focal}}(\sigma(z_{\text{logit},j}), y_j)$.
6. Backpropagate and update weights.

Metrics (accuracy, precision, recall, F1, AUC) are computed per epoch.

3.4. FITB without Curriculum Learning

The model with the highest AUC in the CP task is used as initialization for the FITB task.

3.4.1 Objective

In this setting, the model produces two types of embeddings:

- The **query embedding** z_{q_j} is computed as: $z_{q_j} = \text{OutfitCLIPTransformer}(O'_j) \in \mathbb{R}^{128}$, where O'_j is the incomplete outfit.
- The **positive embedding** z_{p_j} is computed as: $z_{p_j} = \text{OutfitCLIPTransformer}(\{I_{p_j}\}) \in \mathbb{R}^{128}$, where I_{p_j} is the ground-truth item completing the outfit.

The objective is to minimize an in-batch triplet margin loss to learn a ranking where the positive item is closer to the query than negatives:

$$\mathcal{L}_{\text{triplet}} = \frac{1}{B} \sum_{j=1}^{B} \max(0, d(z_{q_j}, z_{p_j}) - \min_{i \neq j} d(z_{q_j}, z_{p_i}) + \mu),$$

where z_{q_j} is the query embedding, z_{p_j} is the positive embedding, $d(\cdot, \cdot)$ is the Euclidean distance, $\mu = 2.0$ is the margin, and negatives are the positives of other batch samples z_{p_i}.

3.4.2 Algorithm

The model, pretrained on CP, is fine-tuned for 200 epochs. For each batch $\{(O'_j, I_{p_j})\}_{j=1}^{B}$:

1. Compute query embeddings $z_{q_j} = \text{OutfitCLIPTransformer}(O'_j) \in \mathbb{R}^{128}$.
2. Compute positive embeddings $z_{p_j} = \text{OutfitCLIPTransformer}(\{I_{p_j}\}) \in \mathbb{R}^{128}$.
3. Calculate pairwise distances $D \in \mathbb{R}^{B \times B}$, where $D_{j,i} = d(z_{q_j}, z_{p_i})$.
4. Extract positive distances $D_{j,j}$ and hardest negative $\min_{i \neq j} D_{j,i}$.
5. Compute loss $\mathcal{L}_{\text{batch}}$ and backpropagate.

For validation on $\mathcal{D}_{\text{FITB}}$, compute distances between query embedding z_{q_j} and candidate embeddings $\{z_{c_{j1}}, \ldots, z_{c_{j4}}\}$, selecting the index with minimum distance. Accuracy is the fraction of correct predictions.

3.5. FITB with Standard Curriculum Learning

3.5.1 Objective

The goal is to improve FITB performance by adopting the curriculum learning strategy from the original OutfitTransformer paper [7], which progressively increases negative sampling difficulty:

- **Epochs** < 40: In-batch negatives, using positive embeddings from other outfits in the batch (as in Section 3.4).
- **Epochs 40–49**: Negatives sampled from the same semantic category c_p as the positive item.
- **Epochs** ≥ 50: Negatives sampled from the same subcategory s_p as the positive item.

Each item I_i has a category c_i and subcategory s_i. Negatives are dynamically sampled items I_{n_j}, with embeddings $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$ computed, except for in-batch negatives, which use $z_{n_j} = z_{p_i}$ (for $i \neq j$). The triplet margin loss minimizes the distance between the query embedding z_{q_j} and positive embedding z_{p_j} while maximizing the distance to the negative embedding z_{n_j}.

3.5.2 Algorithm

The training pipeline is identical to the baseline FITB setup (Section 3.4), except for the negative sampling strategy. For each batch $\{(O'_j, I_{p_j})\}_{j=1}^{B}$:

1. Compute query embeddings $z_{q_j} = \text{OutfitCLIPTransformer}(O'_j) \in \mathbb{R}^{128}$.
2. Compute positive embeddings $z_{p_j} = \text{OutfitCLIPTransformer}(\{I_{p_j}\}) \in \mathbb{R}^{128}$.
3. Sample negative items I_{n_j} based on the curriculum schedule:
   - **Epochs** < 40: Use positive items I_{p_i} (for $i \neq j$) from other batch outfits, with embeddings $z_{n_j} = z_{p_i}$.
   - **Epochs 40–49**: Sample I_{n_j} from the same semantic category c_p as I_{p_j}, and compute $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$.
   - **Epochs** ≥ 50: Sample I_{n_j} from the same subcategory s_p as I_{p_j}, and compute $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$.
4. Calculate pairwise distances $D_{j,n} = d(z_{q_j}, z_{n_j})$ for each query and negative, and $D_{j,j} = d(z_{q_j}, z_{p_j})$ for the positive.
5. Compute the triplet margin loss $\mathcal{L}_{\text{batch}}$ using the positive distance $D_{j,j}$ and the hardest negative distance $\min D_{j,n}$.
6. Backpropagate and update weights.

For validation on $\mathcal{D}^{\text{val}}_{\text{FITB}}$, compute distances between query embedding z_{q_j} and candidate embeddings $\{z_{c_{j1}}, \ldots, z_{c_{j4}}\}$, selecting the index with minimum distance. Accuracy is the fraction of correct predictions.

3.6. FITB with Curriculum Learning and Clustering

3.6.1 Objective

This approach extends the standard curriculum learning strategy (Section 3.5) to further enhance FITB performance by addressing the issue of near-identical garments in the Polyvore dataset. Visualizing examples from the same subcategory revealed that many items are visually or semantically similar, often equally valid for completing an outfit. This similarity poses a challenge for contrastive learning: when negative samples are randomly drawn from the same category or subcategory, they may include items that are actually compatible with the outfit, inadvertently introducing label noise and confusing the model. To mitigate this, we incorporate clustering-based negative sampling, modifying the curriculum schedule:

- **Epochs** < 40: In-batch negatives, as in Section 3.4.
- **Epochs 40–49**: Negatives sampled from the same semantic category c_p as the positive item.
- **Epochs** ≥ 50: Negatives sampled from the same subcategory s_p as the positive but from a different cluster, using cluster IDs k_i.

Each item I_i has a category c_i, subcategory s_i, and cluster ID k_i (from clustering). Negatives are dynamically sampled items I_{n_j}, with embeddings $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$ computed, except for in-batch negatives ($z_{n_j} = z_{p_i}$). The triplet margin loss minimizes the distance between z_{q_j} and z_{p_j} while maximizing the distance to z_{n_j}.

3.6.2 Algorithm

The training pipeline is identical to the standard FITB with curriculum learning setup (Section 3.5), except for the negative sampling strategy in the third phase. For each batch $\{(O'_j, I_{p_j})\}_{j=1}^{B}$, the third phase is as follows, while the rest remains unchanged:

3. Sample negative items I_{n_j} based on the curriculum schedule:
   - **Epochs** < 40: Use positive items I_{p_i} (for $i \neq j$) from other batch outfits, with embeddings $z_{n_j} = z_{p_i}$.
   - **Epochs 40–49**: Sample I_{n_j} from the same semantic category c_p as I_{p_j}, and compute $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$.
   - **Epochs** ≥ 50: Sample I_{n_j} from the same subcategory s_p as I_{p_j} but a **different cluster** $k_i \neq k_p$, and compute $z_{n_j} = \text{OutfitCLIPTransformer}(\{I_{n_j}\}) \in \mathbb{R}^{128}$.

3.6.3 Clustering Methods

Our clustering-based negative sampling relies on grouping items to avoid selecting negatives too similar to the positive. We explore two clustering techniques: K-Means and Hierarchical Dirichlet Processes (HDP).

K-Means Clustering K-Means is applied independently to each subcategory using Fashion-CLIP embeddings to create visually coherent clusters capturing stylistic differences. For each subcategory:

- Collect all items with valid embeddings.
- Set the number of clusters $k = \max(5, \lfloor \frac{n}{\alpha} \rfloor)$, where n is the number of items and $\alpha = 300$ is a parameter chosen after visualizing clusters with different values and evaluating training performance.

Each item receives a cluster ID k_i, stored for training.

Dirichlet (HDP) Clustering While KMeans captures visual and textual similarity at the item level, it does not account for how garments co-occur within outfits. To address this, we experimented with hierarchical Dirichlet processes (HDP), which treats each outfit as a document of item textual descriptions, discovering latent topics (clusters) based on co-occurrence patterns.

This generative model has several advantages:

- It allows for a flexible number of clusters, adapting to the complexity of the data without requiring a fixed k.
- It assigns each outfit a mixture over topics, offering a probabilistic view of its style.

This approach provides a complementary view to K-Means by incorporating higher-level outfit structure.

4. Experiments

This section details the implementation and experimental evaluation of the fashion outfit recommendation system, ensuring reproducibility and summarizing the design alternatives explored to achieve the final architecture. We first describe the technical setup, including the dataset, embedding generation, and training procedures, followed by the experimental stages and key observations that shaped the system's development.

4.1. Implementation Details

The system was implemented using Python 3.12, with dependencies specified for reproducibility. Detailed setup instructions, including dependency installation and compatibility notes, are available in the project's README file[1].

[1] `https://github.com/lara-ocon/TransformingFashion`

4.1.1 Polyvore Dataset

The experiments utilize the Polyvore dataset's `nondisjoint` split, which includes complete outfits with metadata and item-level annotations. The dataset is structured as:

- `images/`: Item images named `item_id.jpg`.
- `item_metadata.json`, `item_title.json`: Category, subcategory, descriptions, and titles.
- `disjoint/`, `nondisjoint/`: Train, validation, and test splits for each task (CP, FITB and CIR).

The `disjoint` version ensures no item overlap between train and test sets, promoting strict generalization. However, in this project we use the `nondisjoint` split, which allows for overlapping items, facilitating more realistic evaluation.

Inside each task directory (`compatibility/` and `fill_in_the_blank/`), the data is organized in `train.json`, `valid.json`, and `test.json` files:

- **Compatibility**: Each entry includes an outfit and a binary label indicating whether it is compatible.
- **Fill-in-the-Blank (FITB)**: Each entry includes an incomplete outfit, a list of four candidate items, the position where the item was removed, and the index of the correct item.

4.1.2 CLIP Embedding Generation

To represent each fashion item as a dense feature vector, we follow the *OutfitTransformer* implementation and generate multimodal embeddings using the pretrained `patrickjohncyh/fashion-clip` model (CLIP ViT-B/32). Each item is encoded based on both its image and text description.

This process is implemented in the notebook `generate_clip_embeddings.ipynb`.

Main steps:

- Each item is loaded using metadata from `item_metadata.json`.
- The corresponding image (`images/item_id.jpg`) is resized to 224×224 and preprocessed.
- Both the image and text are processed through the FashionCLIP model to obtain embeddings.
- The resulting image and text embeddings (each of dimension 512) are concatenated to obtain a single 1024-dimensional vector per item.

Embeddings are computed in batches using PyTorch's `DataLoader`, optionally distributed across GPUs via `torch.distributed`, and saved as `.pkl` files.

4.1.3 Training Procedure for Compatibility Prediction

The CP model, detailed in Section 3.1.1, was trained using precomputed FashionCLIP embeddings. The setup includes:

Dataset and Dataloaders We employed the nondisjoint/compatibility/train.json dataset, where each training example consists of an outfit (i.e., a list of item IDs) along with a binary compatibility label. Precomputed FashionCLIP embeddings were loaded and used directly as item representations, thus avoiding the need for end-to-end image or text encoding during training.

A custom PyTorch dataset class, `PolyvoreCompatibilityDataset`, was used to return a list of `FashionItem` objects along with their compatibility label. These `FashionItem` instances include metadata and its precomputed $(1024,)$ CLIP embedding. The dataloader batch collate function returned batches of size N as a dictionary containing:

- `query: List[List[FashionItem]]` – a batch of outfits.
- `label: List[int]` – the corresponding binary labels.

Training Setup Training runs for 200 epochs using the focal loss and AdamW optimizer, as specified in Section 3. PyTorch implements the OneCycleLR scheduler, gradient clipping (norm 1.0), and four-step accumulation. Checkpoints are saved per epoch, with the best validation AUC model selected for testing and FITB pretraining. Metrics (accuracy, AUC, F1, precision, recall) are computed using PyTorch utilities.

4.1.4 Cluster Generation for Curriculum Learning

Clustering, described in Section 3.6.3, organizes items for FITB's curriculum learning negative sampling. Implementation details include:

K-Means Clustering To generate clusters, we apply the K-Means algorithm individually for each subcategory (`category_id`)).

Letting N denote the number of items within a given subcategory, we determine the number of clusters k dynamically using the heuristic:

$$k = \max(5, \left\lfloor \frac{N}{300} \right\rfloor)$$

This ensures a minimum of 5 clusters per subcategory, while scaling with the amount of available data. For very small subcategories (less than 10 items), clustering is skipped to avoid unstable results.

Cluster assignments are parsed into 3 dictionaries:

- `item_to_info`: maps each `item_id` to its `semantic_category`, `category_id`, and `cluster_id`.
- `cluster_to_items`: maps a cluster identifier tuple `(semantic_category, category_id, cluster_id)` to the list of items it contains.
- `category_to_clusters`: maps each `(semantic_category, category_id)` pair to the set of cluster IDs available within it.

These dictionaries are serialized using `pickle` and are later used during training to sample hard negatives from different clusters of the same subcategory.

HDP Clustering for Curriculum Learning A textual representation is constructed for each item by concatenating all available metadata fields, including `title`, `description`, `categories`, `url_name`, and `related`. For every outfit in the `train`, `valid`, and `test` splits, we aggregate the cleaned descriptions of its items to form a document. These documents serve as input to a Gensim HDP model.

Once trained, the model provides a topic distribution for each document. To assign a cluster to each item, we average the topic distributions across all outfits in which the item appears and assign the item to the topic with the highest mean probability. Items with no valid assignment are grouped under a fallback cluster `-1`.

We store the resulting cluster assignments and then build the same three dictionaries as with K-Means.

4.1.5 Training Procedure for Fill-in-the-Blank (FITB)

The FITB model, detailed in Sections 3.4– 3.6, is trained from the best CP checkpoint. The setup includes:

Dataset and Dataloaders Training uses `nondisjoint/train.json`, creating queries by removing one item per outfit. Validation/testing uses `fill_in_the_blank/valid.json` and `test.json`. `PolyvoreTripletDataset` and `PolyvoreFillInTheBlankDataset` return 1024-dimensional embeddings, with dataloaders providing:

- `query`: List of outfit items.
- `answer` (training): Positive item.
- `candidates` and `label` (validation): List of candidate items and correct index.

Training Setup Training runs for 200 epochs using the triplet margin loss and AdamW optimizer, as in Sections 3.4– 3.6. Curriculum learning samples negatives via `sample_negatives`, with loss computed every four steps. Validation selects the candidate with minimum distance, using PyTorch for accuracy. The best validation accuracy model is selected.

4.2. Experiments

Throughout development, we tested multiple design alternatives to arrive at the final system. Below, we summarize the experimental stages and key observations.

4.2.1 Initial ResNet-18 Implementation

We first implemented OutfitTransformer from scratch using only image inputs with a ResNet-18 backbone. This was inefficient, requiring one hour per epoch on 12GB NVIDIA GPUs at ICAI, with high memory usage and unstable training. It served as a CP baseline but was discarded due to limited iteration potential.

4.2.2 ResNet-18 with Sentence-BERT

We extended the model to include text features, combining ResNet-18 image embeddings with Sentence-BERT text embeddings, aligning with the original paper [7]. This improved CP accuracy to 0.80 but remained computationally expensive, hindering hyperparameter exploration.

4.2.3 Complementary Item Retrieval (CIR) Challenges

Upon moving to the challenging task of Complementary Item Retrieval (CIR), we encountered new scalability issues. The CIR setting requires the model to identify the missing item from a candidate pool of approximately 251,000 items. Even minor inaccuracies in ranking led to performance failures, as many distractor items are nearly identical or equally compatible. Training times increased, and results were poor, leading us to focus on FITB.

4.2.4 CLIP Embedding Adoption

Adopting Wonjun Oh's open-source implementation [10] with CLIP embeddings was transformative. Precomputed FashionCLIP embeddings reduced incredibly training time and memory usage, achieving a CP AUC of 0.95. This pipeline enabled efficient FITB experiments.

4.2.5 CLIP with Standard Curriculum Learning

We tested CLIP embeddings with the standard curriculum learning strategy from the original paper [7], as described in Section 3.5. This improved FITB accuracy from 0.65 to 0.72.

4.2.6 Clustering for FITB

Random negative sampling in FITB often included near-identical or compatible items, causing label noise. We introduced curriculum learning with clustering:

- **K-Means**: Applied per subcategory, producing coherent clusters. Training with K-Means negatives moderately improved FITB accuracy, confirming structured sampling's value.
- **HDP**: Used text descriptions to capture outfit co-occurrence patterns. While less visually coherent than K-Means, HDP clusters reflected stylistic patterns, offering insights for future hybrid approaches.

5. Results

5.1. Compatibility Prediction (CP)

To evaluate the model's ability to predict outfit compatibility, we trained the model on the compatibility prediction task for 200 epochs. As shown in Figure 10, the training loss steadily decreases while the AUC reaches a value of 0.95 by epoch 50. On the test set, our implementation with CLIP features achieves an AUC of **0.9509**.

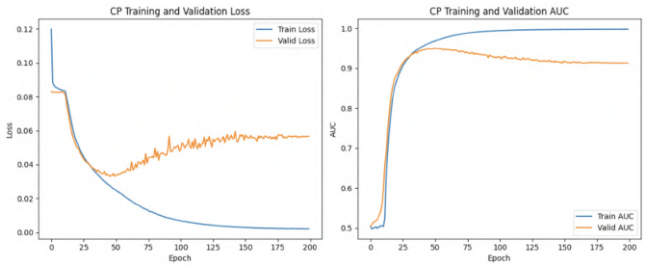

Figure 10: Training loss and AUC over epochs for the compatibility prediction task.

We compare our results against the original OutfitTransformer paper and the open-source implementation. Table 5 summarizes the AUC scores:

Model	CP (AUC)
OutfitTransformer (Paper)	0.93
OutfitTransformer (Open Source)	0.95
OutfitTransformer (Our Impl.)	**0.95**

Table 5: AUC comparison for Compatibility Prediction. Both Open Source Implemtation and our's using CLIP embeddings

5.1.1 Test Metrics

Metric	Value
AUC	0.9509
Accuracy	0.6186
F1 Score	0.7232
Precision	0.5676
Recall	0.9964

Table 6: Test performance of the CP model using the epoch 50 checkpoint and a decision threshold of 0.5.

The Compatibility Prediction (CP) model achieves a high Area Under the Curve (AUC) of 0.9509 on a

balanced test set (10,000 compatible and 10,000 incompatible outfits), indicating excellent ability to rank compatible outfits (label = 1) above incompatible ones (label = 0) across decision thresholds. However, the accuracy of 61.86% is significantly lower, primarily due to the default decision threshold of 0.5 applied to sigmoid-transformed logits.

At this threshold, the model exhibits high recall (99.64%) but low precision (56.76%), correctly identifying 9,964 compatible outfits (true positives) with only 36 misses (false negatives), yet misclassifying 7,592 incompatible outfits as compatible (false positives). The F1 score of 72.32% reflects a trade-off between these metrics. The confusion matrix reveals only 2,408 true negatives, highlighting the model's difficulty in identifying incompatible outfits. This bias toward compatibility predictions is attributed to the threshold being too permissive, as most probabilities exceed 0.5, and to the model's learned representations, which capture consistent stylistic patterns in compatible outfits' Fashion-CLIP embeddings more effectively than the diverse features of incompatible outfits.

Threshold optimization confirms this issue: at 0.6, accuracy improves to 87.94% and F1 to 87.68%, with precision rising to 89.62% and recall dropping to 85.83%, reducing false positives to 1,037. Thresholds below 0.4 classify all outfits as compatible (accuracy = 50.00%, recall = 100.00%), while those above 0.8 classify all as incompatible (accuracy = 50.00%, recall = 0.00%), indicating a skewed probability distribution.

5.1.2 Qualitative results

We also show qualitative results of the CP task in Figure 11, Figure 12, Figure 13 and Figure 14. These examples illustrate successful predictions as well as failure cases.

True Positives (TP). As seen in Figure 11, the model assigns high compatibility scores (all around 0.74) to truly compatible outfits. These examples show consistent alignment in both color palette and style, suggesting that the model effectively captures visual coherence when it is present.

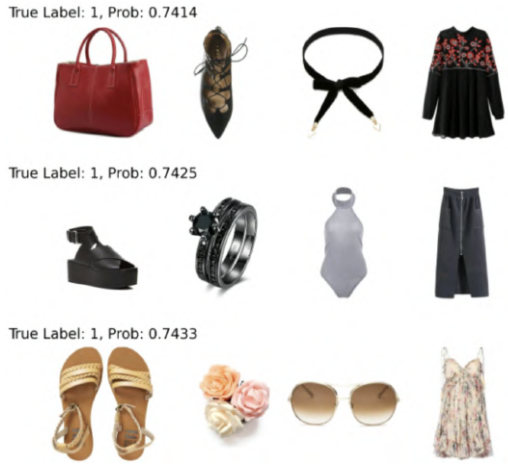

Figure 11: Examples of true positives in CP.

True Negatives (TN). In contrast, the true negatives in Figure 12 are classified correctly but with scores very close to the 0.5 decision threshold. For example, pairing a winter coat with a bikini and heels may be obviously incompatible to a human observer, yet the model only assigns a score of 0.49. Similarly, outfits with clashing styles or prints—such as combinations of strawberries, metallic textures, plaid, and leopard—are also classified as incompatible, but again with low certainty. This reflects a broader pattern observed during evaluation: the model struggles to confidently reject incompatible outfits, rarely assigning very low probabilities. This indecisiveness, clustered around the threshold, reinforces the earlier observation of a poorly calibrated decision boundary.

Figure 12: Examples of true negatives in CP.

False Positives (FP). False positives, shown in Figure 13, further illustrate this issue. The first example scores 0.50, barely surpassing the threshold, even though the outfit is clearly mismatched. The second outfit receives a high compatibility score (0.73), despite being labeled incompatible, highlighting the subjectivity of the task. In a survey we conducted with 36 respondents [18], 69% classified this outfit as compatible, suggesting human disagreement with the dataset's label. Similarly, the third example, scored at 0.54, was deemed compatible by 83% of respondents, reinforcing that some users perceive these looks as coherent. These cases reveal that the model often overestimates compatibility, aligning with human perceptions in subjective cases but conflicting with the binary dataset labels.

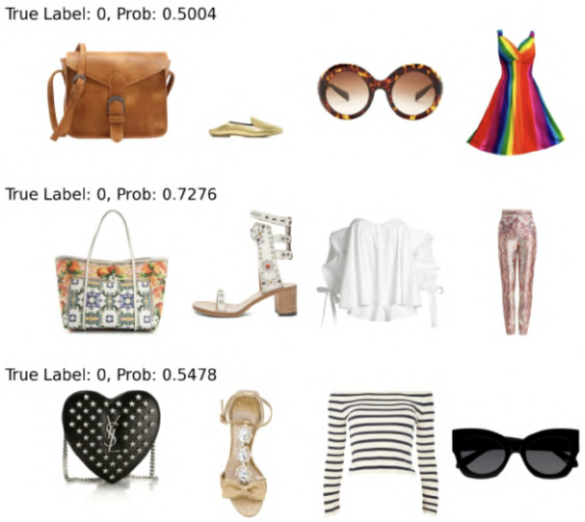

Figure 13: Examples of false positives in CP.

False Negatives (FN). The false negatives in Figure 14 are particularly challenging. All examples are visually bold and involve complex combinations of colors and textures—traits that might confuse the model's compatibility judgment. Scores for these examples cluster tightly around 0.49, demonstrating the model's lack of confidence. Our survey[18] revealed that 55.5% classified the second example as incompatible, despite its compatible label, and 86% deemed the third example incompatible, indicating significant disagreement with the dataset. These findings underscore the difficulty of classifying unconventional outfits, where both the model and human observers struggle to align with the binary ground truth.

Figure 14: Examples of false negatives in CP.

These results, supported by the survey findings, highlight the inherent subjectivity of fashion compatibility. Designers often break traditional rules—intentionally mixing patterns, textures, and styles in ways that defy expectations yet result in acclaimed looks. The divergence between dataset labels and human judgments, as seen in the false positives and negatives, underscores that compatibility is rarely binary, explaining the model's tendency to assign borderline scores and suggesting the need for more nuanced evaluation metrics in future work.

5.2. Fill-in-the-Blank (FITB)

5.2.1 Test Metrics

We evaluate the Fill-in-the-Blank (FITB) task under several training regimes to understand the effectiveness of different learning strategies. Our baseline implementation using CLIP embeddings but without curriculum learning (CL) achieves an accuracy of 0.65. The GitHub implementation [10] uses CLIP embeddings but omits the CL strategy described in the original paper [7], achieving an accuracy of 0.69. By implementing the original paper's CL approach [7]—progressively sampling harder negatives based on semantic categories, as detailed in Section 3.5—we improved accuracy to **0.72**, a significant enhancement over both the original paper's 0.67 and the GitHub baseline.

With our variant of curriculum learning, which introduces negative samples via clustering, sampling negatives from K-Means clusters results in an accuracy of **0.72**, matching the best results achieved with semantic curriculum learning. Using HDP-based clusters also improves performance over the non-CL baseline (0.68 vs. 0.65), although it remains slightly below other strategies. These results suggest that even unsupervised cluster-driven curricula can be beneficial, especially when constructed with meaningful representations.

Model Variant	FITB
OutfitTransformer (Paper, CL) [7]	0.67
OutfitCLIPTransformer (Oh et al.) [10]	0.69
OutfitCLIPTransformer (No CL)	0.65
OutfitCLIPTransformer (Standard CL)	**0.72**
OutfitCLIPTransformer (K-Means CL)	**0.72**
OutfitCLIPTransformer (HDP CL)	0.68

Table 7: Comparison of FITB accuracy for different model variants using FashionCLIP embeddings, except for the original OutfitTransformer, which uses ResNet-18 and Sentence-BERT. CL denotes curriculum learning.

While the clustering strategy (using HDP) improves accuracy over the baseline, it does not outperform curriculum learning with CLIP embeddings. This indicates that while the use of clusters can help, the richness of CLIP embeddings plays a crucial role in the model's performance. Further optimization of the clustering techniques, including more refined hyperparameter tuning and exploring ways to integrate continuous CLIP embeddings into methods like HDP, could potentially enhance the model's performance and clustering effectiveness.

It is also important to note that fashion is inherently subjective. Visual inspection of FITB predic-

tions reveals that many candidate items are visually similar or equally compatible. This observation emphasizes the limitations of using FITB accuracy as the sole metric for performance. In the next section, we explore the model's predictions qualitatively to gain deeper insights into its behavior.

5.2.2 Qualitative Results

Figures 15 and 16 showcase examples of correct and incorrect predictions. These visualizations provide insights into what the model learns and the inherent ambiguities in the dataset.

In the correctly classified examples of Figure 15 , we observe that the model is capable of capturing dominant colors and subtle stylistic coherence. In Example 1, although multiple candidates could plausibly complete the outfit, the model selects the one that aligns better with the outfit's pink tones. In Example 2, the outfit features two pink and three turquoise garments, and the model again selects the pastel color that complements the rest. Example 3 is more ambiguous, involving varied tones, yet the model correctly predicts a pair of earrings that visually match the necklace, revealing the model's capacity to understand finer correlations. Finally, in Example 4, two nearly identical white t-shirts are among the candidates—one with a small logo and the other plain. Despite the minimal difference, the model correctly chooses the labeled option, underscoring both its sensitivity and the problematic nature of this evaluation setup, where multiple answers could be equally valid.

The failure cases in Figure 16 highlight key limitations of the dataset and the task itself. In Example 1, all options are plausible, and the model selects one that—despite being marked incorrect—may not be less compatible than the labeled choice. Example 2 demonstrates a similar issue: although the model chooses a rose-gold watch instead of the golden one labeled correct, the decision is arguably more appropriate given the outfit's palette. Example 3 further exposes the challenge of indistinguishable options—multiple black sunglasses with only slight variations in shape. As expected, the model picks a plausible candidate, but not the one marked correct. Finally, in Example 4, the ambiguity is maximal: two identical pairs of Converse sneakers are presented, yet only one is labeled as correct. The model selects the other, highlighting the unrealistic rigidity of assuming a single ground-truth answer.

These examples demonstrate a major shortcoming of the FITB evaluation setup: the assumption of one correct answer among four often clashes with the subjectivity and flexibility of fashion. The model may be penalized for selecting perfectly valid alternatives. This observation strongly motivated the introduction of cluster-based sampling, not only to reduce confusion during training—by avoiding equally valid negatives—but also to promote more reliable and interpretable evaluation. Clustering helps avoid presenting items that are visually indistinguishable from the

(a) Example 1

(b) Example 2

(c) Example 3

(d) Example 4

Figure 15: Examples of 3 correct predictions of FITB task.

(a) Example 1

(b) Example 2

(c) Example 3

(d) Example 4

Figure 16: Examples of 3 incorrect predictions of FITB task.

ground-truth option, thereby yielding a fairer testing scenario.

5.3. Clustering Analysis

We compare the quality of clusters produced by K-Means and HDP.

5.3.1 K-Means Clusters

In the case of K-Means, we generated clusters independently for each subcategory. For Subcategory 11 (tops), the algorithm formed 30 distinct clusters. Figure 17 showcases examples of three representative clusters.

(a) Cluster 19 - Example 1

(b) Cluster 2 - Example 2

(c) Cluster 24 - Example 3

Figure 17: Examples of 3 different K-Means clusters in Subcategory 11.

Overall, the visual coherence of the clusters suggests that K-Means is effective at capturing meaningful patterns. For instance, the algorithm successfully distinguishes plaid shirts, denim tops, and striped t-shirts—even though all fall under the same subcat-

egory. This implies that CLIP embeddings are rich enough to encode subtle differences in fabric, cut, and texture, which the clustering algorithm can then leverage.

However, K-Means also has limitations. Since it relies purely on visual similarity in the embedding space, it does not account for how garments interact within actual outfits. As a result, it may group items that are visually similar but do not necessarily pair well together in a fashion context. Additionally, the number of clusters must be manually defined per subcategory. Although we used a heuristic to set k, a more principled or adaptive approach could further improve cluster quality, albeit at the cost of higher computational expense.

5.3.2 Example of K-Means Negatives

Figure 18 shows an example from the FITB task using K-Means clusters. In this case, we remove a pair of floral pants from an outfit and sample ten negative items from other clusters within the same subcategory.

(a) Outfit with floral pants removed

(b) Negative candidates from other clusters

Figure 18: Example of cluster-based negative sampling. While all negative samples differ clearly from the removed floral pants, some (like plain ones) could still be compatible—revealing a limitation of this approach.

It can be seen that the sampled negatives differ clearly from the removed item in terms of color, texture, material... This forces the model to learn to identify the most similar item according to visual and semantic cues.

However, this strategy also reveals a key limitation. In this example, the floral pants complete the outfit because they match the jacket's pattern, creating a coherent look. Still, a plain pant—though visually dissimilar to the correct item—could also be compatible. Since K-Means selects negatives purely based on visual distance from the removed item, it often includes valid items as negatives simply because they are not similar in embedding space, even if they could be valid matches in context.

In other words, since K-Means clustering is based solely on the visual embeddings of individual items—without considering how they interact with other garments in real outfits—it may inadvertently teach the model that only items visually close to the removed one are acceptable, ignoring the fact that compatibility can depend on broader contextual relationships within the outfit.

5.3.3 HDP Clusters

Unlike K-Means, which clusters individual items purely based on their CLIP visual embeddings, HDP is applied over full outfits by aggregating the textual descriptions of the items in them. This means the model is indirectly learning which garments tend to appear together—capturing notions of style rather than just visual similarity.

Figure 19 shows three examples of HDP-generated clusters. In the first, we observe a mix of vibrant green, white, and pink tops, including cropped cuts and leaf prints, all suggesting a cheerful or youthful aesthetic. In the second example, pastel-colored blouses dominate, implying a softer, perhaps more elegant style—although a few items seem mismatched, indicating imperfect boundaries. The third cluster features tighter, cropped tops that align with a more modern or bold fashion style, though again, a few outliers appear.

These results suggest that HDP excels at identifying stylistic patterns across outfits, but it occasionally groups garments that do not visually resemble the rest. We attribute this limitation to the lower fidelity of textual metadata compared to CLIP embeddings: item descriptions are often sparse, inconsistent, or incomplete, making it difficult for HDP to fully capture nuanced visual properties.

In future work, we envision combining the strengths of both approaches—retaining the contextual awareness of HDP while leveraging the representational richness of visual embeddings. For instance, adapting HDP to operate over discretized embedding features could offer a hybrid solution. However, discretization may introduce information loss, and developing a principled way to incorporate both signals remains an open challenge.

5.3.4 Example of HDP Negatives

Figure 20 shows an example from the FITB task using HDP-based clustering. In this case, the outfit exhibits a distinct aesthetic—featuring ripped jeans, studded black boots, and an olive green jacket with metallic embellishments, reminiscent of the edgy style of brands like Zadig & Voltaire. The top is removed from the outfit, and the ten negative samples are drawn from other HDP clusters within the same subcategory.

(a) Cluster 6 - Example 1

(b) Cluster 5 - Example 2

(c) Cluster 1 - Example 3

Figure 19: Examples of 3 different HDP clusters in Subcategory 11.

Interestingly, most of the sampled negatives reflect entirely different aesthetics, which would likely clash with the bold and cohesive look of the original outfit. This suggests that HDP can be especially effective at generating meaningful negatives when the outfit style is well-defined.

However, not all outfits are as stylistically distinctive. In more ambiguous or mixed-style cases, HDP's performance tends to degrade, and the sampled negatives may be less useful. Nonetheless, this example illustrates HDP's potential to capture higher-level stylistic coherence, even when relying solely on textual information.

(a) Outfit with t-shirt removed

(b) Negative candidates from other HDP clusters

Figure 20: Example of cluster-based negative sampling using HDP. The sampled negatives differ clearly in style from the removed item, demonstrating how HDP can generate stylistically meaningful contrasts in outfits with a coherent theme.

6. Conclusions and Future Work

6.1. Conclusions

This project has explored the application of transformer-based architectures, specifically the OutfitCLIPTransformer, to the challenging domain of fashion outfit recommendation, focusing on Compatibility Prediction (CP) and Fill-in-the-Blank (FITB) tasks. Our findings underscore several critical insights into the interplay between advanced AI techniques and the inherently subjective and complex nature of fashion.

A key enabler of this work was the use of precomputed CLIP embeddings, as proposed by Oh [10], which significantly improved computational efficiency and model performance. These multimodal embeddings, combining visual and textual features, reduced training time from hours to minutes or seconds per epoch and lowered memory requirements compared to earlier approaches relying on ResNet-18 or Sentence-

BERT. This efficiency facilitated extensive experimentation, achieving an AUC of 0.9509 for CP and an accuracy of 0.72 for FITB, surpassing the original OutfitTransformer's reported results (0.93 AUC for CP and 0.67 accuracy for FITB) and aligning with the performance of Oh's optimized implementation. The superiority of CLIP embeddings over ResNet and Sentence-BERT highlights their ability to capture nuanced visual and semantic features critical for fashion recommendation tasks.

The CP task exposed the subjective and complex nature of outfit compatibility. Despite the high AUC and recall (99.64%), the model's low precision (56.76%) at the default decision threshold of 0.5 indicates a tendency to overpredict compatibility, often misclassifying incompatible outfits. Qualitative analysis revealed that scores for true negatives and false positives clustered near the threshold, reflecting the model's difficulty in confidently rejecting incompatible outfits. This challenge stems from the ranking-based nature of compatibility prediction, where decision boundaries are not well-defined, and the dataset's subjective labels, where even "incompatible" outfits may appear coherent to some users. Threshold optimization improved accuracy to 87.94% at 0.6, but the task remains inherently challenging due to the lack of clear separation between compatible and incompatible outfits.

For the FITB task, adopting the curriculum learning (CL) strategy from the original OutfitTransformer paper [7] improved accuracy from 0.65 (no CL) to 0.72, demonstrating the value of progressively harder negative sampling. Our contribution lies in extending this approach with clustering-based negative sampling, using K-Means and Hierarchical Dirichlet Processes (HDP), to address the issue of near-identical or equally valid candidates in the dataset. K-Means clustering, applied to CLIP embeddings within subcategories, matched the CL performance (0.72 accuracy) by sampling negatives from different clusters, reducing label noise during training. HDP, applied to textual metadata, achieved a slightly lower accuracy (0.68) but offered a complementary perspective by capturing stylistic patterns based on item co-occurrence in outfits. These clustering strategies mitigate the confusion caused by sampling negatives that are visually or stylistically similar to the ground truth, a significant issue given the dataset's vast pool of approximately 251,000 items.

The Complementary Item Retrieval (CIR) task proved exceptionally difficult, akin to finding a needle in a haystack. The requirement to select a single correct item from such a large candidate pool, where many items are nearly identical or equally compatible, led to poor performance. This challenge, combined with observed disagreements among human annotators on outfit compatibility in the dataset, underscores the limitations of current evaluation frameworks that assume a single correct answer. Our qualitative analysis of FITB predictions further highlighted this issue, as the model was often penalized for selecting plausible alternatives that differed only marginally from the labeled ground truth.

In summary, this work demonstrates the effectiveness of leveraging CLIP embeddings, as introduced by Wonjun Oh [10], in conjunction with transformer-based models for fashion recommendation. Our novel clustering-based negative sampling strategies enhance model robustness by addressing dataset ambiguities, while our analysis reveals the critical impact of subjectivity and task complexity on performance. These findings contribute to a deeper understanding of AI-driven fashion recommendation and lay a foundation for future improvements.

6.2. Future Work

Building on the insights gained, several directions for future research emerge to address the limitations identified and further advance fashion recommendation systems:

Hybrid Clustering Techniques While K-Means effectively captured visual similarity and HDP revealed stylistic co-occurrence patterns, neither fully addressed the dual need for visual coherence and contextual compatibility. A hybrid clustering approach could combine the strengths of both methods. For instance, discretizing CLIP embeddings into a vocabulary suitable for HDP could allow the model to learn style-based clusters while retaining the rich visual information of embeddings. Alternatively, graph-based clustering methods that model item co-occurrence as edges and embeddings as node features could better capture the interplay between visual similarity and outfit context. Such techniques would enable negative sampling that avoids confusing the model with compatible alternatives, improving both CP and FITB performance.

Refined Negative Sampling Strategies The CIR task's complexity suggests a need for smarter negative sampling during training. Instead of sampling negatives solely based on visual or categorical differences, future work could incorporate outfit-level compatibility scores (e.g., from the CP model) to ensure negatives are truly incompatible with the query outfit. Additionally, adaptive curriculum learning schedules that dynamically adjust the difficulty of negatives based on model performance could further enhance learning efficiency and robustness.

Multi-Answer Evaluation Metrics The subjectivity in fashion compatibility, evident in both CP and FITB, necessitates evaluation metrics that account for multiple valid solutions. For FITB, rather than penalizing the model for selecting a plausible but "incorrect" candidate, future work could focus on detecting equally valid candidates in the dataset using clustering or similarity analysis of candidate embeddings. By identifying items that are visually or stylistically indistinguishable from the ground truth, evaluation metrics such as accuracy could be adjusted to credit the model for selecting any valid candidate, yielding more representative

performance measures. For CP, exploring soft labels or probabilistic compatibility scores could mitigate the binary nature of current annotations, aligning better with human disagreement in the dataset.

Outfit-Level Style Modeling HDP's ability to identify stylistic patterns suggests potential for outfit-level style modeling. Future work could develop models that explicitly learn style representations (e.g., casual, formal, bohemian) by combining visual embeddings with co-occurrence patterns. These representations could guide negative sampling, improve compatibility predictions, and enable style-transfer applications, such as recommending outfits that adapt a user's preferred style to new contexts.

Enhanced CIR Frameworks To make CIR more tractable, future research could explore hierarchical retrieval approaches, where the model first narrows down the candidate pool to a smaller subset (e.g., using clustering or category filtering) before performing fine-grained ranking. Pre-filtering near-identical items using clustering or similarity thresholds could also reduce the haystack effect, making the task less daunting and more aligned with real-world recommendation scenarios.

References

[1] Han, X., Wu, Z., Jiang, Y. G., Davis, L. S. *Learning Fashion Compatibility with Bidirectional LSTMs.* arXiv:1707.05691, 2017. https://arxiv.org/abs/1707.05691.

[2] McAuley, J., Targett, C., Shi, Q., van den Hengel, A. *Image-Based Recommendations on Styles and Substitutes.* arXiv:1506.04757, 2015. https://arxiv.org/pdf/1506.04757.

[3] Veit, A., Kovacs, B., Bell, S., McAuley, J. *Learning Visual Clothing Style with Heterogeneous Dyadic Co-Occurrences.* arXiv:1509.07473, 2015. https://arxiv.org/pdf/1509.07473.

[4] Cucurull, G., Taslakian, P., Vazquez, D. *Context-Aware Visual Compatibility Prediction.* arXiv:1902.03646, 2019. https://arxiv.org/abs/1902.03646.

[5] Cui, Z., Li, Z., Wu, S., Zhang, X.-Y., Wang, L. *Dressing as a Whole: Outfit Compatibility Learning Based on Node-Wise Graph Neural Networks.* arXiv:1902.08009, 2019. https://arxiv.org/pdf/1902.08009.

[6] Lin, Y., Ren, P., Chen, Z., Ren, Z., Ma, J., de Rijke, M. *Improving Outfit Recommendation with Co-Supervision of Fashion Generation.* arXiv:1908.09104, 2019. https://arxiv.org/pdf/1908.09104.

[7] Sarkar, R., Bodla, N., Vasileva, M. I., et al. *Outfit-Transformer: Learning Outfit Representations for Fashion Recommendation.* arXiv:2204.04812, 2022. https://arxiv.org/abs/2204.04812.

[8] Radford, A., Kim, J. W., Hallacy, C., et al. *Learning Transferable Visual Models From Natural Language Supervision.* ICML, 2021. https://arxiv.org/abs/2103.00020.

[9] Mariya I. Vasileva, Bryan A. Plummer, Krishna Du- sad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. *Learning type-aware embeddings for fashion compatibility.* arXiv:1803.09196, 2018. https://arxiv.org/pdf/1803.09196.

[10] Wonjun Oh. *OutfitTransformer: Outfit Representations for Fashion Recommendation.* GitHub repository, 2023. https://github.com/owj0421/outfit-transformer.

[11] Teh, Y. W., Jordan, M. I., Beal, M. J., Blei, D. M. *Hierarchical Dirichlet Processes.* Journal of the American Statistical Association, 101(476), 2006. https://www.researchgate.net/publication/221997021_Hierarchical_Dirichlet_Processes.

[12] Teh, Y. W. *A Tutorial on Hierarchical Dirichlet Processes.* Machine Learning Group, University of Cambridge, 2006. https://mlg.eng.cam.ac.uk/zoubin/tut06/ywt.pdf.

[13] Řehůřek, R. *Gensim: HdpModel Documentation.* Gensim, 2023. https://radimrehurek.com/gensim/models/hdpmodel.html.

[14] PyMC Team. *Dirichlet Process Mixture Model.* PyMC Examples, 2023. https://www.pymc.io/projects/examples/en/latest/mixture_models/dp_mix.html.

[15] Biased Algorithms. *Hierarchical Dirichlet Process.* Medium, 2023. https://medium.com/biased-algorithms/hierarchical-dirichlet-process-d163e0f43418.

[16] We Talk Data. *Hierarchical Dirichlet Process.* Medium, 2023. https://medium.com/we-talk-data/hierarchical-dirichlet-process-ba2ba7a04615.

[17] Ocón Madrid, L. *Transforming Fashion.* GitHub repository, 2025. https://github.com/lara-ocon/TransformingFashion.

[18] Ocón Madrid, L. *Survey on Fashion Compatibility Prediction.* GitHub repository, 2025. https://github.com/lara-ocon/TransformingFashion/blob/main/CPsurvey.pdf.