SEMANTIC OUTFIT COMPATIBILITY

Eesha Kulkarni
eskulkar@andrew.cmu.edu

Shreya Pagaria
spagaria@andrew.cmu.edu

Suraj Botcha
sbotcha@andrew.cmu.edu

Abstract

Understanding higher-level semantic connections, such as style coherence, shape balance, and formality alignment, is equally vital for fashion compatibility as visual similarities. Traditional recommendation algorithms rely mainly on metadata or low-level color cues, limiting their capacity to recognize sophisticated aesthetic compatibility. In this research, we present a semantic outfit compatibility framework which is a lightweight MLP head developed on top of a frozen CLIP image encoder trained to predict paired fashion compatibility. Using the Polyvore Outfits dataset with a disjoint split, we create positive and negative item pairs and evaluate our model against baselines like color-histogram similarity and CLIP cosine similarity. Our approach targets typical bypass failures, such as strong color similarity, similar-category pairs, and visually misleading hard negatives, and yields significant improvements in AUC over both baselines while being robust across controlled assessment slices. We analyse typical failure instances in more detail, demonstrating that dataset noise, not model restrictions, is the cause of many errors. Lastly, we provide an interactive fashion-styling application - 'Stylo AI' that uses a local LLM stylist and our compatibility model to enable multi-item outfit grading, personalized explanation generation, and useful suggestions. The results reveal that semantic reasoning generated from CLIP embeddings, paired with a trained compatibility head, provides an achievable and understandable framework for computational outfit recommendation.

1. Introduction

Understanding whether two fashion items “go together” requires more than visual similarity; true compatibility depends on higher-level cues such as style coherence, formality, and color harmony. Traditional fashion recommenders struggle with this because they rely heavily on metadata or low-level features. Vision–language models like **CLIP**, however, encode richer semantic information, making them a strong foundation for modeling aesthetic relationships.

This project investigates how effectively CLIP’s pretrained representations can capture outfit compatibility and what additional learning is needed beyond raw similarity. Using the Polyvore Outfits dataset, we build a pairwise compatibility model by freezing the CLIP encoder and training a lightweight MLP head. We compare this approach to low-level color histograms and CLIP cosine similarity baselines, and perform controlled slice evaluations to test robustness to shortcuts such as color-only or category-only matches.

Finally, we integrate the model into an interactive styling application that uses a local LLM (Ollama) to generate explanations and suggestions, combining semantic prediction with user-friendly and interpretable feedback.

1.1. Motivation

Fashion recommendation systems aim to help users find visually compatible clothing items, e.g., suggesting shoes that match a dress. However, existing systems often rely on metadata (colors, categories, or co-purchase data) rather than true visual-semantic understanding. The goal of this project is to predict whether two clothing items go well together using vision-language embeddings from **CLIP (Contrastive Language–Image Pre-training)**. A system that understands *semantic compatibility* could support personal styling, online shopping, and creative exploration. The project connects computer vision, multimodal learning, and style reasoning with practical value for data-efficient and interpretable recommendation systems.

1.2. Limitations of Prior Systems

Prior work on Polyvore often learns compatibility via metric learning, graph reasoning, or category-aware embeddings. However:

- Many systems rely on manually defined categories (metadata).
- Over-reliance on low-level visual features
- Metric-learning approaches can inherit dataset shortcuts.
- Lack of semantic understanding

These lead to limitations between a fit actually being a good fit and just visually matching.

1.3. Our Contributions

We make the following contributions:

- We add a simple **MLP head** to our CLIP model, training it by keeping the CLIP embeddings frozen which outperforms low-level and CLIP-similarity baselines.
- We introduce **controlled evaluation methods** to test the robustness to color similarity, category overlap, and visually deceptive hard negatives.
- We performed a failure case analysis that shows the distinction between model limitations and noise from the data set.
- We create an **interactive styling tool** with pairwise compatibility matrices, multi-item outfit grading, and an LLM stylist that offers natural-language explanations.

2. Related Work

Fashion compatibility was popularized with the Polyvore Outfits dataset [2], inspiring models ranging from metric-learning approaches to graph neural networks and transformers [1, 4]. These methods often rely on category metadata or outfit co-occurrence rather than true semantic reasoning. Vision–language models such as CLIP [3] offer rich semantic embeddings and have been used for fashion retrieval and tagging, though their ability to model compatibility has been less explored. Our work builds on this space by evaluating how far a frozen CLIP encoder, combined with a lightweight MLP, can move beyond low-level similarity baselines and by incorporating an LLM explanation layer for interpretability.

3. Dataset

3.1. Polyvore Outfits Dataset

The Polyvore outfits dataset (first introduced by Han et al.) includes thousands of fashion combinations created by users.Each outfit includes many different clothing items and accessories, and they were put together in a way that creates visual harmony with one another.The dataset has high-quality images, item metadata (IDs, item characteristics, item categories), and train/validation/test splits where the splits do not share items. This means that no item appears in more than one split. Because the way in which items were put together inherently indicates that these items are similar in some sense, this scheme serves as a natural source of weak supervision for creating a learning system to determine which items go together well (i.e., consider to be compatible).

3.2. Pair Construction

To train our pairwise compatibility model, we turn each outfit into a set of positive examples. For any outfit with items $\{i_1, i_2, \ldots, i_n\}$, we list every possible pair of items:

$$(i_a, i_b) \quad \text{for all} \quad a < b.$$

Because these outfits were put together by human stylists, we treat all of these pairs as compatible.Whereas, for negative examples, we randomly sample items from different outfits. The assumption is that randomly pairing pieces usually doesn't result in a cohesive look, so these pairs represent incompatible combinations. We also make sure that none of our negatives overlap with the positive pairs.
Finally, to keep the dataset balanced and the model from being biased, we use a 1:1 ratio of positive and negative pairs.

3.3. Disjoint Split

In order to prevent any images, item identification numbers or metadata from being included in multiple training sets, validation sets and test sets within Polyvore; whenever possible, we try to preserve the disjoint splits of images and items within the dataset.By maintaining these disjoint splits for Polyvore we eliminate any potential data leakage that was likely to occur, which is when a model's performance improves based on memorization of particular item embeddings as opposed to interpreting them according to a general set of compatibility rules. Therefore when completed, the processed dataset consists of 9434 training pairs, 950 validation pairs and 4880 test pairs.

3.4. Data Statistics and Preprocessing

The CLIP preprocessing pipeline is used to resize, center crop, and normalize all item photos that are loaded as RGB.
We primarily use Polyvore's metadata for:

- Slice assessments based on categories
- Recognizing hard negatives
- Qualitative evaluation

Since the model doesn't use metadata, compatibility is determined essentially via visual characteristics.

3.5. Hard Negative Construction

Along with conventional negative examples, we also detect ”hard negative” examples, where two items are considered incompatible even though they look alike (so have similar color-histograms) and are in the same category. These situations allow us to evaluate how well our model can distinguish between ”low-level similarity” (e.g., two red items) and ”semantic compatibility” (e.g., while both items may be red, one item is a shirt while the other is a book).Hard negatives can only be used for evaluating slices and cannot be used in training; therefore, we can adequately test our Robustness without changing the distribution for our training data.

4. Method

Our goal is to model pairwise outfit compatibility using a CLIP-based feature extractor and a lightweight MLP classifier. Given two item images x_a and x_b, we predict a scalar compatibility score $s(x_a, x_b) \in [0, 1]$. We additionally

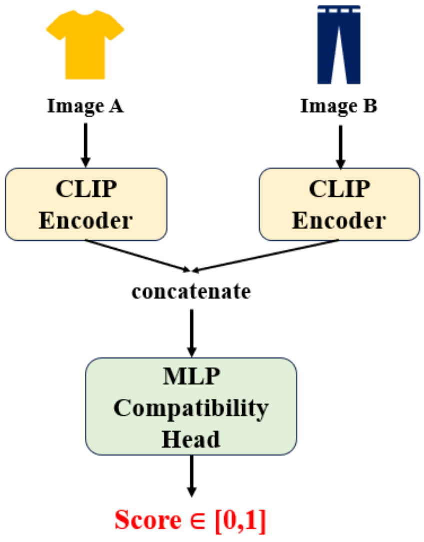


Figure 1. CLIP encoder used to extract semantic embeddings for each fashion item and then topped with a MLP compatibility head for getting compatibility score.

compare our method to several baselines and incorporate an LLM layer for interpretability in the final application.

4.1. CLIP Image Encoder

We use the CLIP ViT-B/32 image encoder to obtain semantic embeddings for each fashion item. For an input image x, CLIP outputs a normalized embedding:

$$z = f_{\text{CLIP}}(x) \in \mathbb{R}^{512}, \quad \|z\|_2 = 1.$$

These embeddings capture high-level concepts such as silhouette, style, formality, and texture, making CLIP well-suited for compatibility reasoning. During training, the CLIP encoder remains frozen to ensure efficiency and avoid overfitting to Polyvore.

4.2. MLP Compatibility Head

Given two CLIP embeddings z_a and z_b, we compute their element-wise interaction using concatenation:

$$h = \big[\, z_a,\ z_b,\ |z_a - z_b|,\ z_a \odot z_b \,\big]$$

A lightweight multilayer perceptron then produces a compatibility probability:

$$\hat{y} = \sigma\big(W_2\, \text{ReLU}(W_1 h + b_1) + b_2\big)$$

We train the MLP using binary cross-entropy loss over positive (same outfit) and negative (cross-outfit) pairs. This head enables the model to learn compatibility patterns beyond raw cosine similarity capturing asymmetry, nonlinearity, and style-specific interactions.

4.3. Baselines

We compare our approach to two non-learned baselines:
1. CLIP Cosine Similarity: The baseline score is simply:

$$s_{\text{cos}} = z_a^\top z_b.$$

This measures semantic similarity but does not model negative evidence or nuanced style interactions.

2. Color Histogram Similarity: For each item, we compute a 3D RGB histogram and evaluate cosine similarity between histogram vectors:

$$s_{\text{color}} = \frac{h_a^\top h_b}{\|h_a\|\,\|h_b\|}.$$

This tests whether compatibility can be explained purely by low-level cues such as color alignment.
These baselines help establish whether our model truly captures semantic compatibility.

5. Results and Robustness Analysis

Evaluating our model on the disjoint Polyvore split, ensuring that no item appears in more than one of the train/val/test partitions. All threshold used for binary decisions is reported on the held-out test set and chosen on the validation split.

5.1. Metrics

Fashion compatibility is inherently continuous rather than binary. Thus, our primary metric is the **Area Under the ROC Curve (AUC)**, which measures how well the model ranks compatible pairs above incompatible ones independent of any threshold. We additionally report Accuracy, F1 score and ROC curves.
Accuracy and F1 scores depend heavily on threshold choice, whereas AUC reflects ranking quality and is therefore more appropriate for compatibility prediction.

5.2. Overall Test Performance

Table 1 summarizes performance on the full test set. Our CLIP+MLP compatibility model significantly outperforms both baselines:

- **Color histogram** which only captures low-level similarity and performs near random.
- **CLIP cosine similarity** is a bit better than color histogram but lacks task-specific reasoning.

5.3. Robustness via Controlled Slices

To test whether our model relies on shortcuts such as color or category correlation, we evaluate performance on controlled subsets (“slices”) of the test set. We construct slices that simulate different failure modes:

Model	AUC	Accuracy	F1
Color Histogram	0.553	0.500	0.665
CLIP Cosine	0.549	0.500	0.667
CLIP + MLP*	**0.762**	**0.714**	**0.761**

Table 1. Overall test performance. Our model substantially outperforms low-level and CLIP similarity baselines.

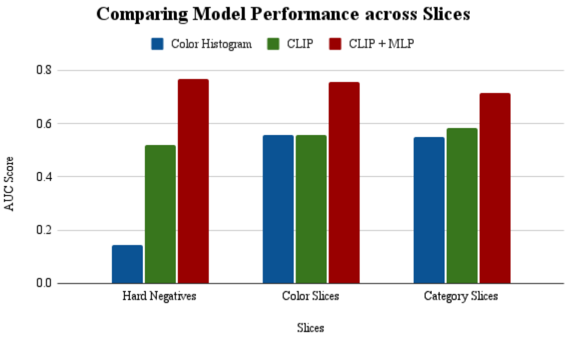


Figure 2. Robustness across controlled slices. CLIP+MLP consistently outperforms both baselines, especially in settings where color or category shortcuts fail.

- **Same-category pairs** (e.g., tops–tops): tests whether the model confuses similarity with compatibility.
- **Different-category pairs**: tests category invariance.
- **High color similarity**: evaluates whether the model relies on color alone.
- **Low color similarity**: requires more semantic reasoning.
- **Hard negatives**: visually similar but incompatible items.

Across all slices, the CLIP+MLP model maintains strong performance, while baselines often collapse especially in high-color-similarity and same-category conditions confirming that our model learns semantic compatibility rather than relying on surface cues.

5.4. Failure Case Analysis

To better understand model behavior, we analyze false positives and false negatives on the test set.

False Positives: These typically arise when two items share strong color palettes or accessory-like features, causing high visual similarity despite poor style coherence. This highlights limitations of relying solely on image embeddings when style cues are subtle.

False Negatives: Many occur due to noisy Polyvore labels e.g., outfits containing clip-art, symbolic items, or visually incoherent but labeled "compatible." In such cases, the model's "error" often reflects dataset noise rather than a modeling flaw.

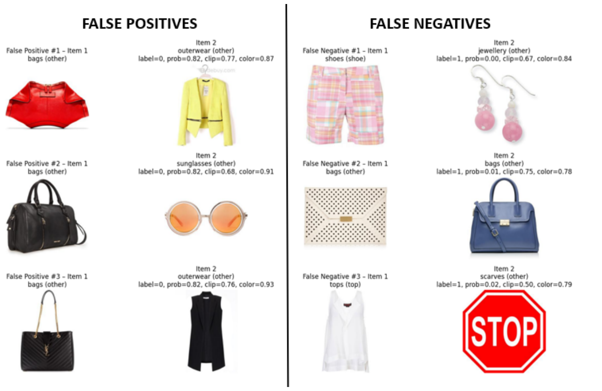

FALSE POSITIVES
False Positive #1 – Item 1
bags (other)
Item 2
outerwear (other)
label=0, prob=0.82, clip=0.77, color=0.87
False Positive #2 – Item 1
bags (other)
Item 2
sunglasses (other)
label=0, prob=0.82, clip=0.68, color=0.91
False Positive #3 – Item 1
bags (other)
Item 2
outerwear (other)
label=0, prob=0.82, clip=0.76, color=0.93
FALSE NEGATIVES
False Negative #1 – Item 1
shoes (shoe)
Item 2
jewellery (other)
label=1, prob=0.00, clip=0.67, color=0.84
False Negative #2 – Item 1
bags (other)
Item 2
bags (other)
label=1, prob=0.01, clip=0.75, color=0.78
False Negative #3 – Item 1
tops (top)
Item 2
scarves (other)
label=1, prob=0.02, clip=0.50, color=0.79
STOP

Figure 3. Examples of false positives (top) and false negatives (bottom). False positives are often caused by strong color similarity; false negatives frequently stem from noisy Polyvore labels.

Understanding these patterns helped clarify the boundary between model limitations and annotation inconsistencies.

6. Interactive Styling Application

We built an interactive fashion styling system that demonstrates the practical value of our compatibility model. To provide real-time outfit feedback and recommendations, the application combines CLIP-based embeddings, our trained compatibility head, and a lightweight LLM stylist.

6.1. App Architecture

Our system follows a modular architecture with three primary components:

- **Feature Extraction:** CLIP encodes each uploaded item into a semantic embedding. Color histograms are extracted in parallel for low-level analysis.
- **Compatibility Engine:** A trained MLP predicts pairwise and overall outfit compatibility using CLIP embeddings.
- **Stylist Module:** A local LLM (via Ollama) generates natural-language explanations and improvement suggestions.

Figure 4 shows an overview of the application pipeline.

6.2. Features

The application provides several user-facing features designed to support both casual users and power users interested in understanding compatibility behavior.

6.2.1. Full Outfit Scoring

Users can upload between 2 and 10 fashion items. The system computes:

- an overall outfit compatibility score,
- a pairwise compatibility matrix,
- optional item-level descriptions (entered by the user).

This feature offers a wholistic assessment of visual harmony and outfit coherence.

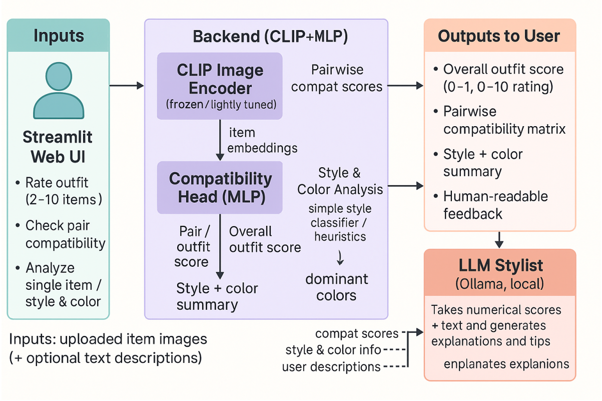


Figure 4. System architecture of the interactive styling application.

6.2.2. LLM Stylist Explanations

The local stylist LLM produces proper explanations of:

- why the outfit works or does not work,
- which items clash or align stylistically,
- actionable improvements (e.g., swap shoes, add layers).

This adds interpretability beyond pure numerical outputs.

6.2.3. Style and Color Analysis

For each item, we compute:

- dominant color palette via histogram clustering,
- CLIP-based style cues,
- a summary explanation generated by the LLM.

This allows users to analyze individual pieces in isolation.

6.2.4. Matching Search

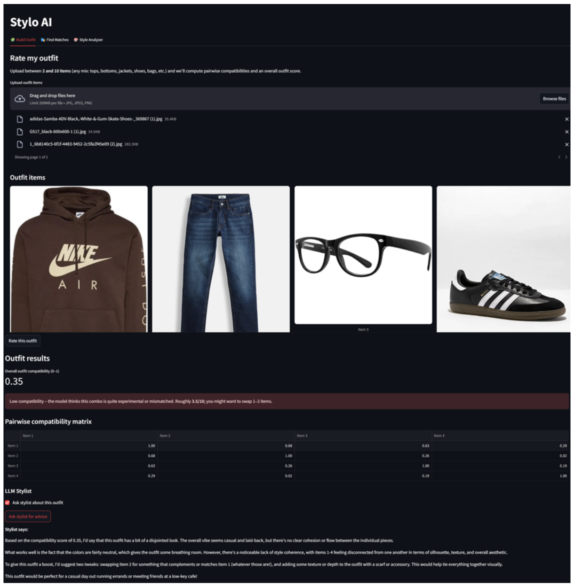
Given a single uploaded item, the application retrieves visually or stylistically similar pieces from the dataset using CLIP embedding similarity. This mirrors a "shopping assistant" experience and enables users to build outfits iteratively.

6.2.5. Demo

A video demonstrating how the app runs locally can be found here. The cloud-deployed version of this application is available online for you to try out!.[1] A sample screenshot of the interface is shown in Figure 5.

7. Discussion

Our experiments highlight both the strengths and limitations of CLIP-based compatibility modeling. While the model consistently outperforms low-level and similarity-based baselines, fashion compatibility remains a subjective and context-dependent task, and several behaviors emerge only through controlled evaluations. We also noticed how adding textual descriptions enhances not only the LLM stylist's explanation, but also helps the model understand cues better.



Rate my outfit

Upload between **2 and 10 items** (any mix: tops, bottoms, jackets, shoes, bags, etc.) and we'll compute pairwise compatibilities and an overall outfit score.

Upload outfit items

adidas-Samba-ADV-Black,-White-&-Gum-Skate-Shoes-_369867 (1).jpg 35.4KB

G517_black-600x600-1 (1).jpg 24.5KB

1_6b8140c5-6f1f-4483-9452-2c5fa2f45e09 (2).jpg 283.3KB

Outfit items

Outfit results

Overall outfit compatibility (0–1)

0.35

Low compatibility – the model thinks this combo is quite experimental or mismatched. Roughly **3.5/10**; you might want to swap 1–2 items.

Pairwise compatibility matrix

	Item 1	Item 2	Item 3	Item 4
Item 1	1.00	0.68	0.63	0.29
Item 2	0.68	1.00	0.26	0.02
Item 3	0.63	0.26	1.00	0.19
Item 4	0.29	0.02	0.19	1.00

LLM Stylist

☑ Ask stylist about this outfit

Stylist says:

Based on the compatibility score of 0.35, I'd say that this outfit has a bit of a disjointed look. The overall vibe seems casual and laid-back, but there's no clear cohesion or flow between the individual pieces.

What works well is the fact that the colors are fairly neutral, which gives the outfit some breathing room. However, there's a noticeable lack of style coherence, with items 1-4 feeling disconnected from one another in terms of silhouette, texture, and overall aesthetic.

To give this outfit a boost, I'd suggest two tweaks: swapping item 2 for something that complements or matches item 1 (whatever those are!), and adding some texture or depth to the outfit with a scarf or accessory. This would help tie everything together visually.

This outfit would be perfect for a casual day out running errands or meeting friends at a low-key cafe!

Figure 5. Example interface of the outfit compatibility application showing the overall score, pairwise matrix, and stylist feedback.

7.1. What the model learns

The results across controlled slices suggest that the model captures meaningful *semantic* signals rather than relying on superficial similarity. For instance:

- The CLIP+MLP head maintains strong AUC on high color-similarity slices, indicating that it does not simply equate matching colors with compatibility.
- Performance on same-category pairs shows that the model distinguishes between "visually similar" and "stylistically coherent" items.
- Hard-negative experiments reveal that the MLP corrects for some of CLIP's weaknesses, learning nonlinear interactions (e.g., when silhouettes or formality conflict despite similar appearance).

Combined with failure case analysis, these findings suggest the model learns a form of *style alignment* beyond raw cosine similarity.

7.2. Fine-Tuning CLIP on Polyvore

To understand if we can further improve the semantic understanding beyond the frozen CLIP encoder, we performed a small-scale fine-tuning experiment using the Polyvore train split. We froze the transformer backbone except for the final transformer block and trained it along with our MLP compatibility head. After five epoch, the validation AUC improved steadily from the frozen CLIP baseline (0.736) to

[1] App link: `https://semantic-outfit-compatibility-vlr.streamlit.app/`

0.750. However, for the test set evaluation, the fine-tuned model achieved an AUC of 0.737 (slightly lower than the frozen model's 0.762). These results suggested how limited fine-tuning on Polyvore provides marginal gains when validating, but does not yield significant performance boost on the test data.

7.3. Limitations

Despite its effectiveness, several limitations remain:

- **Dataset Noise:** Certain Polyvore "outfits" include symbolic images, clip art, or visually illogical objects that have been labelled as compatible. These result in false negatives that are annotation artefacts rather than modelling flaws.
- **Pairwise Assumption:** Our method models compatibility only between pairs of items. True outfit reasoning often depends on global interactions among multiple pieces (e.g., balancing colors across three items), which pairwise scoring cannot fully capture.
- **Frozen CLIP Encoder:** While CLIP provides strong semantic priors, freezing the encoder limits adaptation to fashion-specific nuances. Fine-tuning or lightweight adaptation (e.g., LoRA) may yield richer style representations.
- **Lack of Context** The model does not consider:
  - user preferences,
  - seasonality or occasion,
  - cultural style variations.

  Thus, compatibility is interpreted in a generic, dataset-driven manner.
- **Interpretability Constraints** The LLM stylist offers explanations but cannot inspect image features directly (in its current non-vision form). Future versions may integrate a vision-capable LLM for more grounded reasoning. These limitations motivate future work on multi-item transformers, improved datasets, context-aware modeling, and human-in-the-loop personalization.
- **Fine-Tuning Constraints** The fact that Polyvore dataset is relatively small and somewhat biased in style, makes it challenging to meaningfully improve CLIP without overfitting. A larger-scale fine-tuning dataset like DeepFashion or ShopLook would perhaps lead to significant improvements.

8. Conclusion

Using frozen CLIP embeddings and a lightweight MLP head, we introduced a semantic compatibility model that significantly outperformed both low-level and cosine-similarity baselines. We showed that the model captures significant style-level relationships instead of surface-level cues like colour or category overlap through controlled slice evaluations and failure-case analysis. Lastly, we demonstrated how compatibility modelling can support useful, user-facing fashion tools by integrating the model into an interactive styling application with an LLM-based explanation layer.

9. Future Work

Several promising directions remain:

- **Multi-item reasoning:** Extend beyond pairwise scoring using a transformer or graph-based model to evaluate outfit compatibility holistically.
- **Improved datasets:** Curate cleaner, more consistent outfit labels to reduce noise observed in Polyvore false negatives.
- **Personalization:** Model user preferences, occasions, and style profiles for individualized recommendations.
- **Vision-capable LLM integration:** Replace text-only stylist with a vision-enabled LLM to provide grounded, image-aware explanations.

These directions offer opportunities to develop richer, more expressive systems for computational fashion understanding.

References

[1] Z. et al. Cui. Dressing as a whole: Outfit compatibility learning via node-wise graph neural networks. In *ICCV*, 2019. 2

[2] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry Davis. Learning fashion compatibility with bidirectional lstms. In *CVPR*, 2017. 2

[3] A. et al. Radford. Learning transferable visual models from natural language supervision. *ICML*, 2021. 2

[4] et al. Song. Polyvore outfit generation with transformers. In *ECCV*, 2019. 2