Fashion Outfit Compatibility with Neural Networks

Ankitha Udupa, Ganesh Prasad Shivakumar, Parisa Ghanad Torshizi
Khoury College of Computer and Information Science
Northeastern University
Boston, MA
udupa.a@northeastern.edu, prasad.g@northeastern.edu,
ghanadtorshizi.p@northeastern.edu

12/13/2022

1 Objectives and Significance

Fashion plays a substantial role in society with widespread applications in online shopping and social media. The task of outfit recommendation involves predicting items of clothing that have high compatibility together whilst also fulfilling the fashion needs of the user. Since fashion preferences differ from user to user, these fashion preferences and compatibility between multiple objects are intrinsic features that are far more complex to learn as compared to visual similarity problems. The growing field of eCommerce and online shopping for fashion items has inspired us to develop a machine learning model that can generate fashionable outfits as a set of clothing items that are highly compatible with each other. Through this objective, we are targeting to solve our problem of Outfit Matching.

Although considerable research has been made in the fields of clothing retrieval [1, 2], clothing recognition [3], and clothing recommendation [4, 5], they fall short of considering the visual compatibility between multiple items to form an outfit. Previous work has also proposed learning Siamese [6] networks to learn the distance between different fashion items, however, these methods only focus on learning pairwise compatibility in contrast to learning compatibility between a set of items constituting an outfit as a whole. We propose to learn a visually semantic embedding space in conjunction with compatibility between fashion items in a comprehensive framework. Furthermore, we propose a multi layer perceptron comparison network that derives features of images at different levels, builds pairwise compatibility matrices for each feature, and finally computes the whole outfit's compatibility score . Figure 6 shows our proposed approach. Our end goal is to be able to achieve Compatibility prediction, which means given an outfit from the user, the model predicts how compatible or 'fashionable' the outfit is.

2 Background

2.1 Resnet-50

Vanishing gradient is a problem which occurs in deep neural networks with many layers. It happens when the gradient, which controls the learning of weights in the network, gets close to zero when going through the earlier layers of the network.

This causes the network to train at a very low pace or not learn at all. Resnet-50 or residual network is a model that addresses the problem of vanishing gradient in deep neural networks by introducing a “shortcut path” or “ skip connection” into the structure of the network. This skip connection solves this problem by allowing the gradient to be applied directly to the activations of the earlier layers before reaching to those deep layers. Resnet-50 has also shown promising results in solving computer vision problems, as it allows for very deep structures that can extract many features from the image data. Fig. 1 respectively shows the regular block and the residual block.

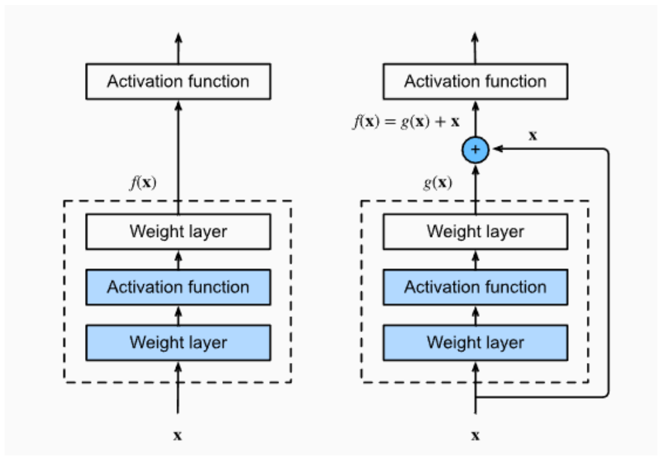


Figure 1: The regular block (left) and the residual block

In the regular block, the output of the block can be formulated as: $X = g(WX + b)$, where X is the activation, g is the activation function,W represents weights, and b is the introduced bias. This structure is also called the “main path”, where stacking these regular blocks builds a regular neural network.

In the residual block, the output of the block can be formulated as: $X = g(WX + b + X)$, where the output from an earlier layer is added to the formula. The residual blocks are used in the Resnet in two different architectures:

Identity block:
In the identity block the dimension of the input activation and output activation is the same. The Identity block is illustrated in Fig. 2.

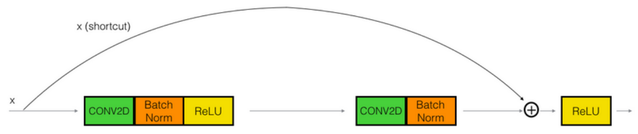


Figure 2: The Identity block

Convolutional block:
In the convolutional block the dimension of the input activation and output activation are not the same because of the Conv2D layer added to the shortcut path. The convolutional block is illustrated in Fig. 3.

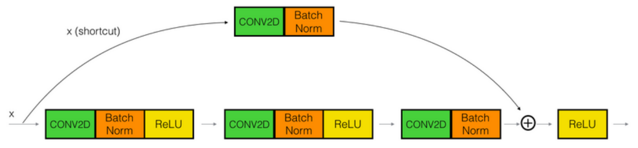


Figure 3: The convolutional block

The architecture of the Resnet-50 model:

The Resnet-50 model consists of 5 stages, each with a three layer convolutional block and a three layer identity block. Fig. 4

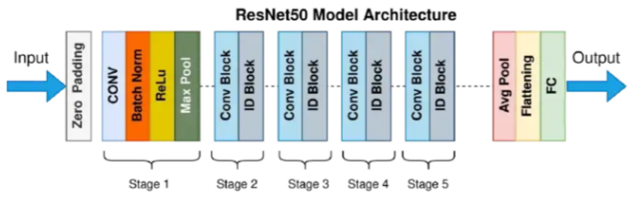


Figure 4: The Resnet-50 architecture

This Resnet model was further used in the code to extract features from the images for training the model. There are other alternatives to Resnet that can be used for feature extraction as well, e.g: Inception V3.

Resnet-50 was used by Wen et al. in Fault diagnosis in smart manufacturing [32], in facial expression recognition by Lima et al. [33], and as Malaria Cell-Image Classification by Rey et al. [34].

2.2 Multi Layer Perceptron

Neural networks, at first, were inspired by the brain's structure. However, later, they developed into deep learning models, their purpose was not to model human's brain, but to find multiple levels of patterns within the data. The structure of the neural networks started with studying a single neuron. These neurons were able to produce an output based on a set of weights and inputs. However, the problem with the neurons was that they were not able to learn the weights, like the brain does. Therefore, a new concept called perceptrons were introduced. Perceptron initializes a random set of weights, then based on the weights produces a weighted sum of input, and creates the output by applying the sigmoid activation function on the weight input. It could also learn the optimal set of weights using the gradient descent. The limitation of using a single perceptron was that it could only output a linear combination of data. In order to solve this problem, Multilayer Perceptron, aka MLP, was proposed. This Multilayer perceptron allows a non-linear mapping of inputs to outputs by stacking multiple perceptrons together in many hidden layers.

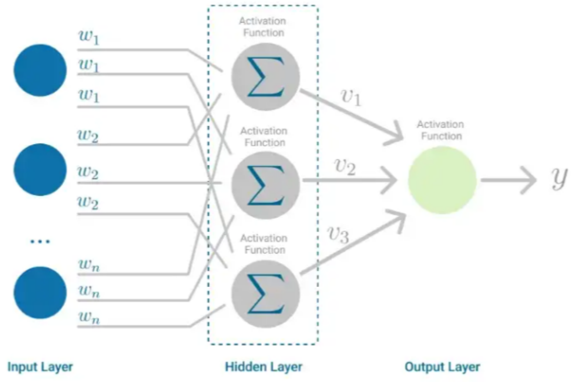


Figure 5: The structure of an MLP

2.3 Visual Semantic Embedding

This is a type of embedding that connects the multimodal visual and textual data. It basically maps the images and their related texts to the same or close feature space. This can be leveraged in

3 Related Work

3.1 Fashion image recognition and retrieval

Over the years, substantial work has been undertaken in image retrieval tasks and fashion image retrieval falls under this broad umbrella of work. Earlier work included simple approaches for image matching and retrieval using siamese networks architecture [7, 8]. These simple architectures learn the image features as clothing attributes in order to find similarity between corresponding fashion images where similar identities are likely to be closer to each other and vice versa. Ma, Zhe, et al. [12] uses two attention models to learn an attribute specific embedding network to jointly learn multiple attribute specific embeddings to compute fine-grained fashion image similarity. These networks work well for pairwise image matching, however they fail to consider similarity between multiple different objects. Moreover, Kalantidis et al. [13] use more fine-grained techniques for clothing recognition with human pose estimation and Yamaguchi, Kota, et al. [14] use deep learning methods to achieve pose-independent visual garment retrieval.

Clothing recognition has been used for fashion style recognition [9], occupation recognition [10] and social circle recognition. Clothing recognition has also been used for person re-identification by matching clothing images across cameras in the surveillance field [11]. Our study can significantly contribute to the mentioned scenarios as we are learning pairwise similarity between clothing items which could be used for similar image retrieval.

3.2 Fashion recommendation

Recommendation systems consist of three main branches - collaborative filtering, content-based filtering and hybrid method. Most existing clothing recommendations utilize collaborative filtering [17] techniques to make recommendations which are dependent on collecting information from the user. On the other hand, content-based filtering [18] enables the model to make recommendations based on the contents of the item itself. Our proposed work assumes the content-based filtering method.

Clothing recommendation is a novel area of research with only a few documented machine learning research. Liu, Si et al. [16] developed the concept of “Magic Mirror” which modeled low-level features from the human body and mid-level features from facial attributes to develop a high-level recommendation system. Yu-Chu, Lin, et al. [19] propose a Bayesian network approach that makes situation and user-specific recommendations for combinations of clothing from the user's wardrobe. Moreover, Lin, Yun-Rou, et al. [20] use convolutional neural networks to develop clothing recommendation systems that takes into account the user’s gender, body height, clothing attributes and clothing color to make more robust and personalized recommendations. Our experiment can be extended to personalized recommendations of outfits based on outfit compatibility scores that we compute.

3.3 Fashion compatibility learning

In the fashion domain, fashion compatibility measures whether different items of clothing match for a visually compatible set of items. Over the years, image data has been used along with neural networks to learn network embeddings. McAuley, Julian, et al. [26] uses convolutional neural network features to measure the difference between two items using a distance metric to ascertain compatibility between the items.Wang, et al. [24] uses deep autoencoders to capture an accurate embedding space by learning highly non-linear network structures. However, research has indicated that using multi-model could be highly beneficial for complex data and representation learning tasks. For example, Ngiam, Jiquan, et al. [25] attempts to solve the problem of speech recognition by training a multimodal encoder to learn shared representation between visual and speech inputs. In addition, Song, Xuemeng, et al. [21] proposes a content-based neural scheme model along with a bayesian personalized ranking metric to learn compatibility between clothing items with multi-modal inputs. In this paper, we intend to learn compatibility between clothing items that construct a 'fashionable' outfit using multimodal inputs - text description of the clothing item along with the image. Our work closely aligns with the work done by Han, Xintong, et al.[15] where they learn compatibility between clothing items using Bi-directional LSTMs. We use MLP instead of Bi-Directional LSTM in our experiment to compute fashion compatibility in clothing items.

3 Methods

We aim to learn compatibility between clothing items using an end-to-end comparison network which consists of four steps.

1. Extracting features for different concepts like color, style *etc.* using a multi-layer feature extractor with ResNet-50 CNN architecture mentioned in section 3.1.
2. Creating enumerated pairwise similarity between features using comparison modules in section 3.3.
3. Computing the compatibility score of the comparison modules using a Multi-layered perceptron network (MCN) to compute in section 3.2. The MCN is first used to predict outfit compatibility and then by backpropagating the gradients from the output score to input to determine the relation of each similarity to the compatibility score.
4. Enabling multi-modal inputs using a visual semantic embedding in section 3.4

The overall framework is shown in Figure 6.

3.1 Multi-Layer Representation

We require high-level features of fashion images that the model can use to recognize the outfit in a detailed manner to learn concepts like fashion compatibility. One approach used in deep learning is predefining concepts like color, print etc. However, these definitions may be insufficient to learn high level concepts of fashion. Present-day deep learning makes use of deep CNN architectures to learn high level features. As the CNN architecture gets deeper, the receptive field of the model increases. Therefore, the initial layers learn low-level features like color, texture etc. whereas the later layers effectively learn high-level features such as style and fashion compatibility.

In our experiment, we use the ResNet-50 architecture to learn representation of fashion clothing items that range from low to high level features. Leveraging the information at each layer, extract representations from different layers of ResNet-50 and construct comparison modules. Furthermore, we use Global Average Pooling at intermediate layers to reduce the feature maps extracted to vectors. There are two advantages to using the Global Average Pooling - Firstly, it enables easier comparison by converting the feature maps at intermediate layers into vectors. Secondly, it effectively eliminates irrelevant spatial information.

In our experiment, we have extracted features from 4 intermediate layers {conv_block_188,conv_block_189,conv_block_190,conv_block_191} of ResNet-50 which are

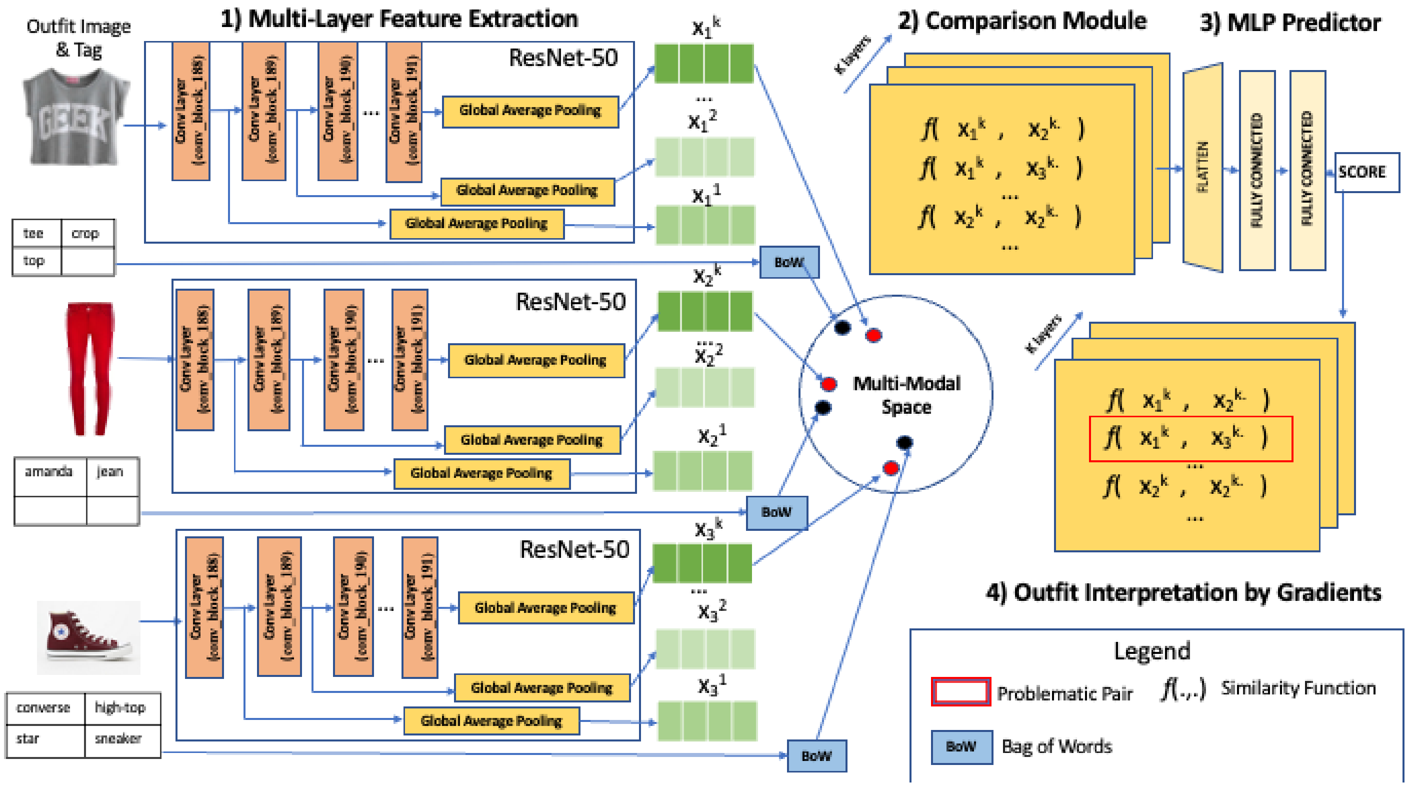


Figure 6) Overview of the proposed framework for outfit compatibility.

passed to 4 different comparison modules to calculate pairwise similarity $\{M^1, M^2, \dots M^K\}$ which is the input to compute the score for outfit compatibility in equation (2).

3.2 Outfit Interpretation Using Gradients

Outfit compatibility is the result of comparing individual items in an outfit with others based on different features like color, texture, style etc. A simple linear model can be used to learn comprehensively compatibility and pairwise similarity. Using a simple linear model has advantages as the importance of each input can be indicated by the weight of each input and linear models tend to have high interpretability. However, linear models suffer from limited capacity. To that end, we use a Multilayer Perceptron model (MLP) which has low explainability due to hidden units but has better capacity. We leverage back propagation gradients to estimate the significance of each similarity for incompatibility.

Given an outfit with N items, their features can be denoted as $X = \{x_1, x_2, x_3, \dots x_N\}$ where x_i is the vector representing the i-th item.The matrix that denotes the enumerated pairwise similarity among features can be represented as :

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1N} \\ m_{21} & m_{22} & \dots & m_{2N} \\ \dots & \dots & \dots & \dots \\ m_{N1} & m_{N2} & \dots & m_{NN} \end{bmatrix} \quad (1)$$

M is the comparison matrix, where m_{ij} represents the similarity between vectors x_i and x_j and $m_{ij} = m_{ji}$. There are K-different comparison matrices $\{M^1, M^2, \dots M^K\}$ to compare K-different aspects. These matrices are fed into a 2-layer multi-perceptron network to compute compatibility scores.

$$cs = W_2 ReLU(W_1[M^1, M^2, ..M^K] + b) \tag{2}$$

Where the different *M* matrices are concatenated and flattened into a vector representation. The MLP gives us a nonlinear relationship between the compatibility score *cs* and comparison matrix *M.* However, the nonlinear relationship makes it difficult to approximate the significant contribution of each input to the compatibility score. To that end, to estimate the importance of each input, we use the derivatives of *cs*

$$cs \approx W'[M^1, M^2, ..M^K] + b \tag{3}$$

The equation above represents a linear model whose weights can be interpreted as the importance of each input similarity matrix. The W' is the derivative of *cs* w.r.t each input similarity $\{M^1_0, M^2_0, \dots M^K_0\}$ at each point. If incompatibility is labeled as 1, w^k_{ij} represents the importance of each similarity with regards to compatibility.

$$w^k_{ij} = \frac{\delta cs}{\delta m^k_{ij}}\Big|_{[M^1_0, M^2_0, ..M^k_0]} \tag{4}$$

For each item, the importance can be derived by summing up all the gradients of all similarity vectors containing it. W_s corresponds to the importance of the s-th item.

$$w_s = \sum_{1 \leq K} \sum_{i=s; j \neq s} w^k_{ij} \tag{5}$$

While training, we use a sigmoid activation function as denoted by (6) which gives us an output score in terms of compatibility probability. We use Binary-cross entropy loss as denoted in (7).

$$\sigma(x) = \frac{1}{1 + exp^{-x}} \tag{6}$$

$$\mathcal{L}_{\mathcal{MLP}} = ylog\sigma(x) + (1-y)log(1-\sigma(x)) \tag{7}$$

3.3 Projected Embedding Comparison

The pivotal aspect of this experiment is computing the pairwise similarity m_{ij} in the comparison matrix. Usually, metrics like Cosine similarity, Euclidean distance, Manhattan distance *etc.* are used to compute similarity/distance between features in a space. However, using these metrics leads to additional problems like 1) variations in compatibility will be constricted *e.g.* all shirts that are compatible with a bottom will be pushed closer together. This forces a relationship between the shirts to make them seem compatible with each other even if they are not. 2) Triangular inequality may restrict the positioning of items in the embedding. The theorem states that the sum of any two sides of a triangle is greater than or equal to the third side. This may lead to problems in our experiment *e.g.* If a shirt matches with a bottom and the bottom matches with a footwear, then the triangular inequality forces a relationship between the footwear and the shirt. To that end, to address these problems we use projected embeddings for different fashion type compositions.

For a given fashion outfit, we model pairwise combinations between different types of items (shirt, footwear, bottoms *etc*.) for projecting the embeddings to a different subspace $m_{ij} = f(x_i,x_j)$, where m_{ij} is the similarity between x_i and x_j.

$$Proj^{i \rightarrow (i,j)} x_i = ReLU(x_i \otimes mask_{ij})$$

(8)

$$f(x_i, x_j) = CosineSimilariy(Proj^{i \rightarrow (i,j)} x_i, Proj^{j \rightarrow (i,j)} x_j)$$

(9)

Where the $Proj^{i \rightarrow (i,j)}$ is the projection of the i-th item conditioned on the combination (i,j). The projection in equation (9) is the dot product between feature x_i and $mask_{ij}$ - the learnable mask with the same dimensions as x_i. The mask term selects suitable feature vectors for different conditions of compatibility and can be interpreted as an element wise gating function.

$$L_{masks}(mask) = ||mask||_1$$

(10)

$$L_{embedding}(x) = ||x||_2 \quad (11)$$

Finally, we have loss functions corresponding to mask and embeddings where L_{masks} pushes masks to be sparse and $L_{embedding}$ encourages the CNN to devise normalized feature representations in the subspace.

3.4 Visual Semantic Embedding

Visual semantic embeddings leverage information from visual items using labeled images as well as semantic information from text to promote multimodal learning. The polyvore dataset provides descriptions for each clothing item in an outfit like "studded denim backpack". These descriptions can be denoted as $W = \{w_1,w_2,...w_n\}$. Where w_i is the word at the i-th position and can be transformed to a one-hot-encoded vector h_i.

The word embeddings can be computed as $u_i = W_T h_i$. The semantic embedding for a given outfit is $U = 1/N \sum_{i=1}^{N} u_i$. W_T is the weight of the word embedding model. Similarly the visual features x are mapped to the embedding space as $V = W_I x$. The objective of the visual semantic embedding is to place WE and IE for a given item close to each other in the latent space. The loss for the VSE is a contrastive loss as described by (12).

$$L_{vse}(u, v, W_T, W_I) = \sum_{v}\sum_{k} max(0, m - dist(u,v) + dist(v, u_k) + \sum_{u}\sum_{k} max(0, m - dist(u,v) + dist(u, v_k) \quad (12)$$

Where, d(u,v) is the distance function that calculates the distance between two embeddings u and v. For a matching composition u,v. u_k and v_k are the semantic embedding and visual embeddings for all non-matching items. The loss for the model minimizes the distance between matching pairs u,v and increases the distance between non-matching items with a margin m.

Finally, the joint loss L_{Total} for our final framework is

$$L_{Total} = L_{MLP} + \lambda_1 L_{masks} + \lambda_2 L_{embedding} + \lambda_3 L_{VSE} \quad (13)$$

$\lambda_{\{1,2,3\}}$ are the weights for the additional losses. The trainable parameters for our model are $\{\Theta, W_T, W_I, W_1, W_2, b, Mask\}$. Θ is the parameter for the ResNet-50 model. W_1, W_2, b are the learnable parameters for the Multi-perceptron model as stated in equation (2) and Mask is the learnable masks from equation (8).

With the current model we have built we are learning nearly 16M parameters, please refer below for more specifics about our model:

```
Total params: 16,041,448
Trainable params: 16,010,600
Non-trainable params: 30,848
```

4 Experiment

In this section, we introduce our dataset and the method used for pre-processing the data before training. We also summarize the details related to the training process of our framework.

4.1 Dataset

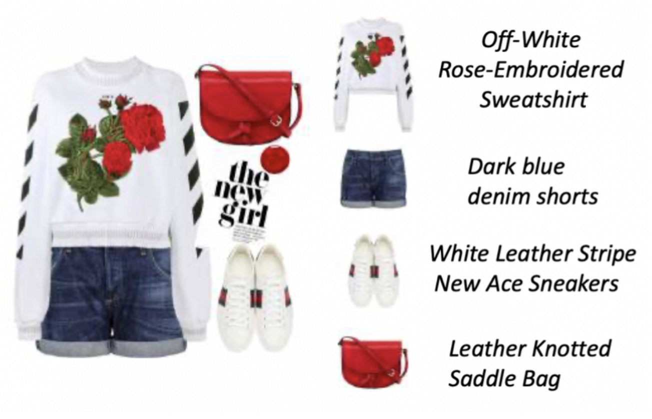


Figure 7: Shows a sample outfit from the polyvore website. This includes an image with fashion items along with their text descriptions.

Fashion-focused online communities such as Polyvore, Chictopia, Lookbook have developed rapidly in recent years. People are fond of making and sharing outfits on these platforms. The Polyvore fashion website enables users to create outfits as compositions of clothing items, each containing rich multi-modal information such as product images, text descriptions, associated tags, popularity score, and type information as seen in Figure 7. We are planning to refer to the dataset used by Xintong Han [31], which contains 21,889 outfits from polyvore.com, in which 17,316 are for training, 1,497 for validation and 3,076 for testing. This dataset is sufficient to validate our use case of Outfit prediction and it supports multimodal learning with image and text description. This dataset has some inconsistencies in the test set that make quantitative evaluation unreliable. The following sections describe the steps taken to deal with the intricacies in the dataset.

4.2 Data Cleaning and Aggregation:

There were two problems with this data set.

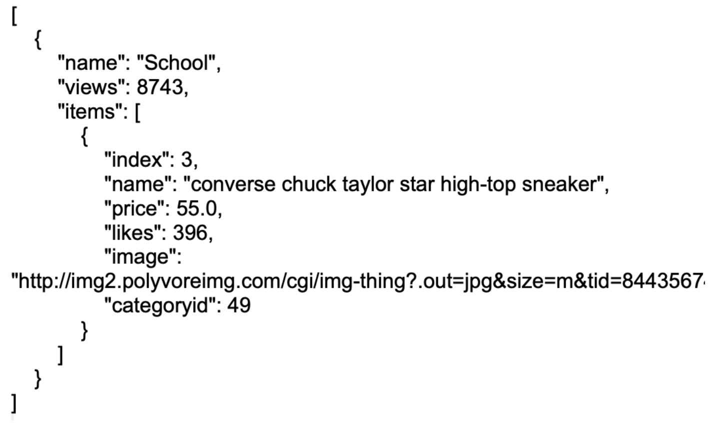
Heavy Data: The data volume was 45 GB, which required high disk space. To tackle that, we reduced unnecessary information by writing a cleaning script to reduce the data by removing some of the inconsequential attributes like price, color and size of the outfit. Other features like description were maintained for the prediction.

Inconsistency in the test set: There were inconsistencies in the test set that made the quantitative evaluation unreliable. This problem was tackled by enriching the test dataset annotated with outfit and item ID, fine-grained item type, title, text descriptions, and outfit images.

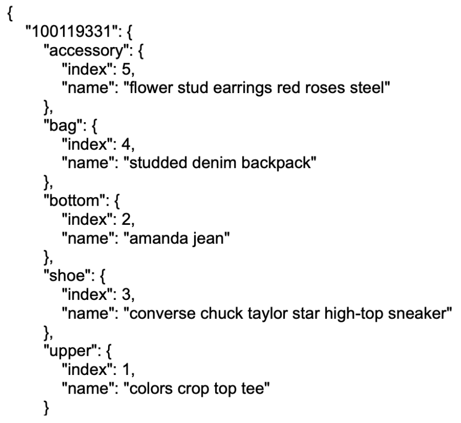
The results of data cleaning are displayed in Figure 8. By tackling these issues in the data, we constructed a dataset where each item in an outfit is associated by one of these five categories: Top, Bottom, Footwear, BackPacks, Addons.

4.3 Data Generator

We will be creating visual-semantic embedding out of images and titles of the outfits. However, if we try to load around 100 outfits, it requires an in-memory space of 2GB. Therefore, to train the model with more outfits to use the full potential of the dataset that is available we have adapted a *Data Generator*. Through a data generator, we can read images on the go when they will be used for training. Since we are reading the images on the go, we are saving memory and even a system with 8GB RAM can be used to train the model from the 30GB dataset.



```
[
   {
      "name": "School",
      "views": 8743,
      "items": [
         {
            "index": 3,
            "name": "converse chuck taylor star high-top sneaker",
            "price": 55.0,
            "likes": 396,
            "image":
"http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=8443567
            "categoryid": 49
         }
      ]
   }
]
```



```
{
   "100119331": {
      "accessory": {
         "index": 5,
         "name": "flower stud earrings red roses steel"
      },
      "bag": {
         "index": 4,
         "name": "studded denim backpack"
      },
      "bottom": {
         "index": 2,
         "name": "amanda jean"
      },
      "shoe": {
         "index": 3,
         "name": "converse chuck taylor star high-top sneaker"
      },
      "upper": {
         "index": 1,
         "name": "colors crop top tee"
      }
```

Figure 8: The figure on the left displays the raw data from the polyvore dataset. The figure on the right displays the cleaned data with item ID, fine-grained item type, title, text descriptions *etc.*

The Data Generator is a technique that breaks down the memory-intensive problem into smaller batches, and works on those batches on the go. Keras already has a built-in data generator function; however, in this experiment, we implemented a data generator that changes the data into sequences, which allows for multiprocessing and ensures that each data is trained once on each sample per epoch. In the code, the image data is first loaded using the image_loader, it is then being resized into target dimensions, so that all images have the same dimension. It is then transformed into tensors. The run_for_each_data returns image id, parts stating which part of the style (including upper, bottom, shoe, bag, accessory) that image belongs to, index, and the name of the image which is a description of it. The __getitem__is responsible for creating batches of the input data created by the run_for_each_data function. The whole process allowed us to create sequential batches of data points using the tensorflow library.

For each batch of data generator will have:

- 16 outfits

- Images encoded with dimensions 224 x 224 x 3 using Pillow module.
- Names are encoded.
- Sequence lengths.
- Categorical index of the outfit.
- Introducing compatibility flag

4.4 Data Encoding:

In this section, we discuss the processing techniques of images, texts and labels before passing the data onto ResNet-50 for the image feature extraction.

Preprocessing Images: We have pre-downloaded the images before model training. And we get the image object from the PIL.Image package and transform it to 224 x 224 x 3 tensors.

Preprocessing Texts: We will try to encode the outfit names using Bag of Words technique, where the whole product title is changed to a vector of size equal to the number of words that are present in the product title. This creates the problems with variable sequence lengths for training sequence models, but it's handled using padding and masking methods.

Labels: Outfit ids are added during data preprocessing as part of each batch of the data generator which helps in training the model and calculating the predictions and losses.

Introduction of Compatibility Flag: Since we have scraped the data from the polyvore website, all the samples which are added on the website are compatible. While processing the data, we have introduced random noise to creative negative or incompatible samples. And based on the classification loss we are evaluating the model performance.

Incompatible samples introduction, actually helps in getting new outfits for the future generation. This is one of the real life use case where fashion designers try to invent new outfits, through VSE embedding we are predicting compatibility of the new outfit.

Also, For our experiment, due to lack of GPU resources we have used 100 outfits with 5 images in each outfit for training and 30 outfits with 5 images each for testing and validation.

4.5 Experiment Setting

Training: In our experiment, we use a ResNet-50 model as the backend CNN with pretrained weights. The spatial size of each input is 224x224x3 which is resized during the data generator step as mentioned in section 4.3. An outfit has a range of individual items between 3 to 5. The empty sequences are padded with zeros. Post the comparison module, we add batch normalization on the similarities computed. The experiments were conducted on Google Colab with GPU. We ran 50 epochs with a mini batch size of 32. The optimizer we used is SGD with a momentum of 0.9. We set the additional losses weights lambda {5e-3, 5e-4, 1} respectively. The training and validation loss can be seen in table 1.

5 Results

Metric	Evaluation Score
Training loss	0.0328
Validation loss	1.2322
Training AUC	0.5227
Accuracy	0.6667
Test Loss	0.6800
Testing AUC	0.6458

Table 1: The evaluation metric values

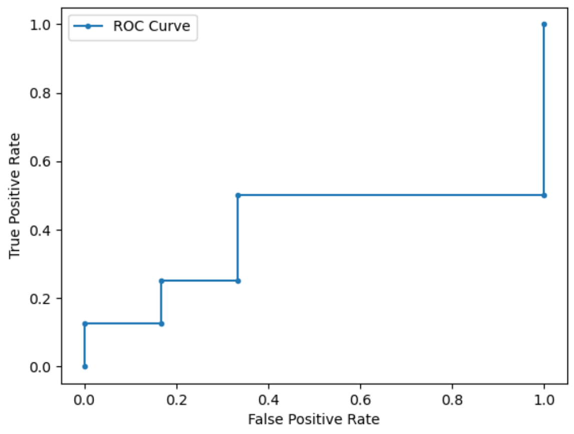


Figure 9: The ROC Curve for test set

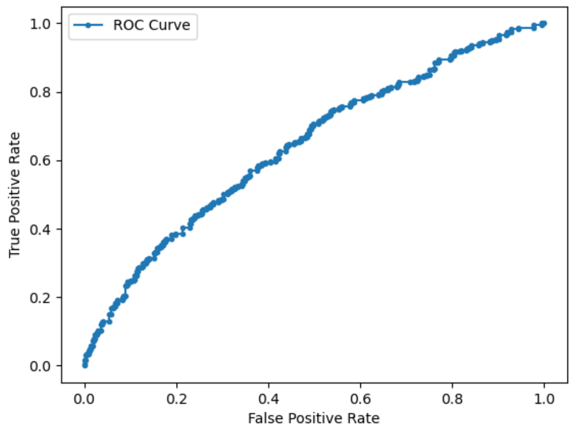


Figure 10: The ROC Curve for the training set

Metrics we have calculated are the AUC, Accuracy and losses such as: Total Loss according to the formula 13 , consists of the clf_loss (the MLP model loss), vse_loss (the visual semantic embeddings loss), features_loss (loss of the driven features of the images) ,and tmasks_loss (loss of the learnable mask vectors).

As we introduced the incompatible or new outfits during data preprocessing we use this classification problem over VSE embedding to validate the model performance. Metrics AUC and Accuracy comes in handy to evaluate models for the classification problems.

We like to highlight one point here, due to limited resources we had to calculate this metrics with fewer epochs of around 50 and 100 outfit samples. Most of different category outfits are not even trained, due to which AUC and accuracy are pretty low. We have future plans to improvise this project. So currently with the limited resource we ran training which resulted in the AUC of ~0.6, we suspect with more epochs this will definitely get better.

5 Conclusion

In this paper, we introduce an end-to-end framework for fashion outfit compatibility which includes a multi-layered network, comparison modules and visual semantic embedding. We first use the MLP model to compute the compatibility score and then use backpropagation to gauge the importance of computed similarity to incompatibility. The use of multi-layered representation enhances performance as the model can learn both low and high level features to predict compatibility scores. For future work, we can explore different models like Bi-Directional LSTM, Transformers and also different evaluation techniques.

6 Individual Tasks

Ankitha Udupa: I was responsible for constructing the backend ResNet-50 model for multi-layer feature extraction. I used Tensor Flows subclassing to create the ResNet-50 architecture as mentioned in section 2.1. I was also responsible for creating the MLP model for outfit interpretation along with creating the embeddings. In addition, I also contributed to writing the report.

Ganesh Prasad: I was responsible for the data extraction and preprocessing through DataGenerators to optimize the RAM utilization. I also worked on the outfit compatibility loss calculation where we selected a random combination of tops, bottoms and accessories and calculated the compatibility score for the provided outfit combination. In addition, I also contributed to writing the report.

Parisa Ghanad: I was responsible for implementing the training, evaluation, conducting the experiments and getting the results from the data, and writing the report. I also read the related papers to find and understand the models.

7 References

[1] Chen, Qijin, et al. "Measuring clothing image similarity with bundled features." *International Journal of Clothing Science and Technology* (2013).

[2] Frejlichowski, Dariusz, Piotr Czapiewski, and Radosław Hofman. "Finding similar clothes based on semantic description for the purpose of fashion recommender system." *Asian Conference on Intelligent Information and Database Systems*. Springer, Berlin, Heidelberg, 2016.

[3] Shubathra, S., P. C. D. Kalaivaani, and S. Santhoshkumar. "Clothing image recognition based on multiple features using deep neural networks." *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2020.

[4] Kang, Wang-Cheng, et al. "Visually-aware fashion recommendation and design with generative image models." *2017 IEEE international conference on data mining (ICDM)*. IEEE, 2017.

[5] Hu, Yang, Xi Yi, and Larry S. Davis. "Collaborative fashion recommendation: A functional tensor factorization approach." *Proceedings of the 23rd ACM international conference on Multimedia*. 2015.

[6] Veit, Andreas, et al. "Learning visual clothing style with heterogeneous dyadic co-occurrences." *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[7] Melekhov, Iaroslav, Juho Kannala, and Esa Rahtu. "Siamese network features for image matching." *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016.

[8] Gajic, Bojana, and Ramon Baldrich. "Cross-domain fashion image retrieval." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.

[9] Kiapour, M. Hadi, et al. "Hipster wars: Discovering elements of fashion styles." *European*

conference on computer vision. Springer, Cham, 2014.

[10] Song, Zheng, et al. "Predicting occupation via human clothing and contexts." *2011 International Conference on Computer Vision*. IEEE, 2011.

[11] Layne, Ryan, et al. "Person re-identification by attributes." *Bmvc*. Vol. 2. No. 3. 2012.

[12] Ma, Zhe, et al. "Fine-grained fashion similarity learning by attribute-specific embedding network." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 07. 2020.

[13] Kalantidis, Yannis, Lyndon Kennedy, and Li-Jia Li. "Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos." *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*. 2013.

[14] Yamaguchi, Kota, et al. "Parsing clothing in fashion photographs." *2012 IEEE Conference on Computer vision and pattern recognition*. IEEE, 2012.

[15] Han, Xintong, et al. "Learning fashion compatibility with bidirectional lstms." *Proceedings of the 25th ACM international conference on Multimedia*. 2017.

[16] Liu, Si, Luoqi Liu, and Shuicheng Yan. "Magic mirror: An intelligent fashion recommendation system." *2013 2nd IAPR Asian Conference on Pattern Recognition*. IEEE, 2013.

[17] Yeung, Kam Fung, and Yanyan Yang. "A proactive personalized mobile news recommendation system." *2010 Developments in E-systems Engineering*. IEEE, 2010.

[18] Yeung, Kam Fung, and Yanyan Yang. "A proactive personalized mobile news recommendation system." *2010 Developments in E-systems Engineering*. IEEE, 2010.

[19] Yu-Chu, Lin, et al. "Personalized clothing-recommendation system based on a modified Bayesian network." *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*. IEEE, 2012.

[20] Lin, Yun-Rou, et al. "Clothing recommendation system based on visual information analytics." *2019 International Automatic Control Conference (CACS)*. IEEE, 2019.

[21] Song, Xuemeng, et al. "Neurostylist: Neural compatibility modeling for clothing matching." *Proceedings of the 25th ACM international conference on Multimedia*. 2017.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567 (2015).

[23] Cheng Wang, Haojin Yang, Christian Bartz, Christoph Meinel. "Image Captioning with Deep Bidirectional LSTMs." arXiv preprint arXiv:1604.00790 (2016).

[24] Wang, Daixin, Peng Cui, and Wenwu Zhu. "Structural deep network embedding." *Proceedings of the 22nd ACM SIGKDD*

international conference on Knowledge discovery and data mining. 2016.

[25] Ngiam, Jiquan, et al. "Multimodal deep learning." *ICML*. 2011.

[26] McAuley, Julian, et al. "Image-based recommendations on styles and substitutes." *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 2015.

[27] Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013.

[28] Donahue, Jeffrey, et al. "Long-term recurrent convolutional networks for visual recognition and description." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[29] Yue-Hei Ng, Joe, et al. "Beyond short snippets: Deep networks for video classification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[30] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernock`y, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In Interspeech.

[31] Han, Xintong, et al. "Learning fashion compatibility with bidirectional lstms." *Proceedings of the 25th ACM international conference on Multimedia*. 2017

[32] Wen, Long, Xinyu Li, and Liang Gao. "A transfer convolutional neural network for fault diagnosis based on ResNet-50." *Neural Computing and Applications* 32, no. 10 (2020): 6111-6124.

[33] Li, Bin, and Dimas Lima. "Facial expression recognition via ResNet-50." International Journal of Cognitive Computing in Engineering 2 (2021): 57-64.

[34] Reddy, A. Sai Bharadwaj, and D. Sujitha Juliet. "Transfer learning with ResNet-50 for malaria cell-image classification." In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0945-0949. IEEE, 2019.