Fashion Outfit Generation for E-commerce

Elaine M. Bettaney
ASOS.com
London, UK
elaine.bettaney@asos.com

Stephen R. Hardwick
ASOS.com
London, UK
stephen.hardwick@asos.com

Odysseas Zisimopoulos
ASOS.com
London, UK
odysseas.zisimopoulos@asos.com

Benjamin Paul Chamberlain
ASOS.com
London, UK
ben.chamberlain@asos.com

ABSTRACT

Combining items of clothing into an outfit is a major task in fashion retail. Recommending sets of items that are compatible with a particular seed item is useful for providing users with guidance and inspiration, but is currently a manual process that requires expert stylists and is therefore not scalable or easy to personalise. We use a multilayer neural network fed by visual and textual features to learn embeddings of items in a latent style space such that compatible items of different types are embedded close to one another. We train our model using the ASOS outfits dataset, which consists of a large number of outfits created by professional stylists and which we release to the research community. Our model shows strong performance in an offline outfit compatibility prediction task. We use our model to generate outfits and for the first time in this field perform an AB test, comparing our generated outfits to those produced by a baseline model which matches appropriate product types but uses no information on style. Users approved of outfits generated by our model 21% and 34% more frequently than those generated by the baseline model for womenswear and menswear respectively.

KEYWORDS

Representation learning, fashion, multi-modal deep learning

1 INTRODUCTION

User needs based around outfits include answering questions such as "What trousers will go with this shirt?", "What can I wear to a party?" or "Which items should I add to my wardrobe for summer?". The key to answering these questions requires an understanding of *style*. Style encompasses a broad range of properties including but not limited to, colour, shape, pattern and fabric. It may also incorporate current fashion trends, user's style preferences and an awareness of the context in which the outfits will be worn. In the growing world of fashion e-commerce it is becoming increasingly important to be able to fulfill these needs in a way that is scalable, automated and ultimately personalised.

In: J. Degenhardt, S. Kallumadi, U. Porwal, A. Trotman (eds.):
Proceedings of the SIGIR 2019 eCom workshop, July 2019, Paris, France, published at http://ceur-ws.org

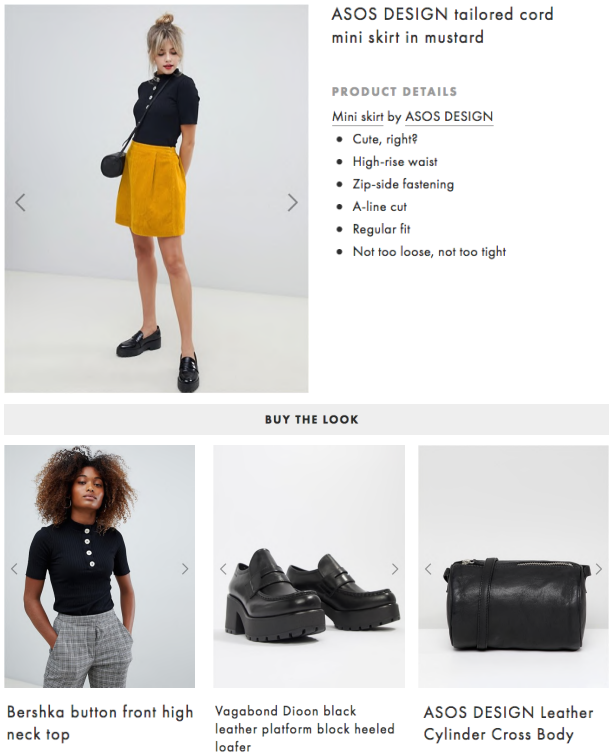


Figure 1: An ASOS fashion product together with associated product data and styling products in a *Buy the Look* (BTL) carousel as shown on a Product Description Page (PDP).

This paper describes a system for Generating Outfit Recommendations from Deep Networks (GORDN) under development at ASOS.com. ASOS is a global e-commerce company focusing on fashion and beauty. With approximately 87,000 products on site at any one time, it is difficult for customers to perform an exhaustive search to find products that can be worn together. Each fashion product added to our catalogue is photographed on a model as part of an individually curated outfit of compatible products chosen by our stylists to create images for its Product Description Page (PDP). The products comprising the outfit are then displayed to the customer in a *Buy the Look* (BTL) carousel (Figure 1). This offering however is not scalable as it requires manual input for every outfit. We aim to learn from the information encoded in these outfits to automatically generate an unlimited number of outfits.

A common way for people to compose outfits is to first pick a seed item, such as a patterned shirt, and then find other compatible items. We focus on this task: completing an outfit based on a seed item. This is useful in an e-commerce setting as outfit suggestions can be seeded with a particular product page or a user's past purchases. Our ASOS outfits dataset comprises a set of outfits originating from BTL carousels on PDPs. These contain a seed, or 'hero product', which can be bought from the PDP. All other items in the outfit we refer to as 'styling products'.

There is an asymmetry between hero and styling products. Whilst all items are used as hero products (in an e-commerce setting), styling products are selected as the best matches for the hero product and this matching is directional. For example when the hero product is a pair of Wellington boots it may create an engaging outfit to style them with a dress. However if the hero product is a dress then it is unlikely a pair of Wellington boots would be the best choice of styling product to recommend. Hence in general styling products tend to be more conservative than hero products. Our approach takes this difference into account by explicitly including this information as a feature.

We formulate our training task as binary classification, where GORDN learns to tell the difference between BTL and randomly generated negative outfits. We consider an outfit to be a set of fashion items and train a model that projects items into a single *style space*. Compatible items will appear close in style space enabling good outfits to be constructed from nearby items. GORDN is a neural network which combines embeddings of multi-modal features for all items in an outfit and outputs a single score. When generating outfits, GORDN is used as a scorer to assess the validity of different combinations of items.

In summary, our contributions are:

(1) A novel model that uses multi-modal data to generate outfits that can be trained on images in the wild i.e. dressed people rather than individual item flat shots. Outfits generated by our model outperform a challenging baseline by 21% for womenswear and 34% for menswear.
(2) A new research dataset consisting of 586,320 fashion outfits (images and textual descriptions) composed by ASOS stylists. This is the world's largest annotated outfit dataset and is the first to contain Menswear items.

2 RELATED WORK

Our work follows an emerging body of related work on learning clothing style [11, 24], clothing compatibility [18, 20, 24] and outfit composition [4, 7, 10, 23]. Successful outfit composition encompasses an understanding of both style and compatibility.

A popular approach is to embed items in a latent *style* or *compatibility* space often using multi-modal features [10, 20, 21, 24]. A challenge with this approach is how to use item embeddings to measure the overall outfit compatibility. This challenge is increased when considering outfits of multiple sizes. Song et al. [20] only consider outfits of size 2 made of top-bottom pairs. Veit et al. [24] use a Siamese CNN, a technique which allows only consideration of pairwise compatibilities. Li et al. [10] combine text and image embeddings to create multi-modal item embeddings which are then combined using pooling to create an overall outfit representation.

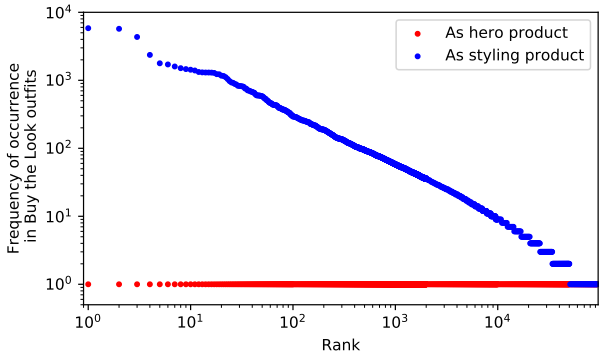


Figure 2: Frequency of occurrence of each womenswear item in our ASOS outfits dataset. Items are ranked by how frequently they occur as styling products. Each item appears once at most as a hero product (red), while there is a heavily skewed distribution in the frequency with which items appear as styling products (blue).

Pooling allows them to consider outfits of variable size. Tangseng et al. [21] create item embeddings solely from images. They are able to use outfits of variable size by padding their set of item images to a fixed length with a 'mean image'. Our method is similar to these as we combine multi-modal item embeddings, however we aim not to lose information by pooling or padding.

Vasileva et al. [23] extend this concept by noting that compatibility is dependent on context - in this case the pair of clothing types being matched. They create learned type-aware projections from their style space to calculate compatibility between different types of clothing.

3 OUTFIT DATASETS

The ASOS outfits dataset consists of 586,520 outfits, each containing between 2 and 5 items (see Table 1). In total these outfits contain 591,725 unique items representing 18 different womenswear (WW) product types and 22 different menswear (MW) product types. As all of our outfits have been created by ASOS stylists, they are representative of a particular fashion style.

Most previous outfit generators have used either co-purchase data from Amazon [12, 24] or user created outfits taken from Polyvore [4, 5, 10, 14, 20, 21, 23], both of which represent a diverse range of styles and tastes. Co-purchase is not a strong signal of compatibility as co-purchased items are typically not bought with the intention of being worn together. Instead it is more likely to reflect a user's style preference. Data collected from Polyvore gives a stronger signal of compatibility and furthermore provide complete outfits.

The largest previously available outfits dataset was collected from Polyvore and contained 68,306 outfits and 365,054 items entirely from WW [23]. Our dataset is the first to contain MW as well. Our WW dataset contains an order of magnitude more outfits than the Polyvore set, but has slightly fewer fashion items. This is a consequence of ASOS stylists choosing styling products from a subset of items held in our studios meaning that styling products can appear in many outfits.

Table 1: Statistics of the ASOS outfits dataset

Department	Number of Outfits	Number of Items	Outfits of size 2	Outfits of size 3	Outfits of size 4	Outfits of size 5
Womenswear	314,200	321,672	155,083	109,308	42,028	7,781
Menswear	272,120	270,053	100,395	102,666	58,544	10,515

For each item we have four images, a text title and description, a high-level product type and a product category. We process both the images and the text title and description to obtain lower-dimensional embeddings, which are included in this dataset alongside the raw images and text to allow full reproducibility of our work. The methods used to extract these embeddings are described in Sections 4.3 and 4.4, respectively. Although we have four images for each item, in these experiments we only use the first image as it consistently shows the entire item, from the front, within the context of an outfit, whilst the other images can focus on close ups or different angles, and do not follow consistent rules between product types.

4 METHODOLOGY

Our approach uses a deep neural network. We acknowledge some recent approaches that use LSTM neural networks [4, 14]. We have not adopted this approach because fundamentally an outfit is a set of fashion items and treating it as a sequence is an artificial construct. LSTMs are also designed to progressively forget past items when moving through a sequence which in this context would mean that compatibility is not enforced between all outfit items.

We consider an outfit to be a set of fashion items of arbitrary length which match stylistically and can be worn together. In order for the outfit to work, each item must be compatible with all other items. Our aim is to model this by embedding each item into a latent space such that for two items (I_i, I_j) the dot product of their embeddings $(\mathbf{z}_i, \mathbf{z}_j)$ reflects their compatibility. We aim for the embeddings of compatible items to have large dot products and the embeddings of items which are incompatible to have small dot products. We map input data for each item I_i to its embedding $\mathbf{z}_i$ via a multi-layer neural network. As we are treating hero products and styling products differently, we learn two embeddings in the same space for each item; one for when the item is the hero product, $\mathbf{z}_i^{(h)}$ and one for when the item is a styling product, $\mathbf{z}_i^{(s)}$; which is reminiscent of the context specific representations in language modelling [13, 15].

4.1 Network Architecture

For each item, the inputs to our network are a textual title and description embedding (1024 dimensions), a visual embedding (512 dimensions), a pre-trained GloVe embedding [15] for each product category (50 dimensions) and a binary flag indicating the hero product. First, each of the three input feature vectors is passed through their own fully connected ReLU layer. The outputs from these layers, as well as the hero product flag, are then concatenated and passed through two further fully connected ReLU layers to produce an item embedding with 256 dimensions (Figure 3). We use batch normalization after each fully connected layer and a dropout rate of 0.5 during training.

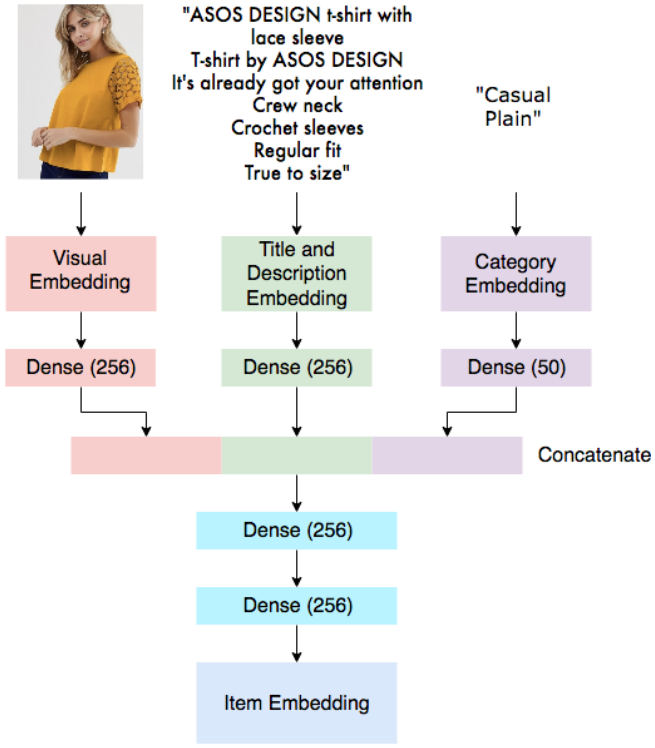


Figure 3: Network architecture of GORDN's item embedder. For each item the embedder takes visual features, a textual embedding of the item's title and description, a pre-trained GloVe embedding of the item's product category and a binary flag indicating if the item is the outfit's hero product. Each set of features is passed through a dense layer and the outputs of these layers are concatenated along with the hero product flag before being passed through two further dense layers. The output is an embedding for the item in our *style* space. We train separate item embedders for womenswear and menswear items.

4.2 Outfit Scoring

We use the dot product of item embeddings to quantify pairwise compatibility. Outfit compatibility is then calculated as the sum over pairwise dot products for all pairs of items in the outfit (Figure 4).

For an outfit $\mathcal{S} = \{I_1, I_2, ..., I_N\}$ consisting of N items, the overall outfit score is defined by

$$y(\mathcal{S}) = \sigma\left(\frac{1}{N(N-1)} \sum_{\substack{i,j=1 \\ i<j}}^{N} \mathbf{z}_i \cdot \mathbf{z}_j\right), \tag{1}$$

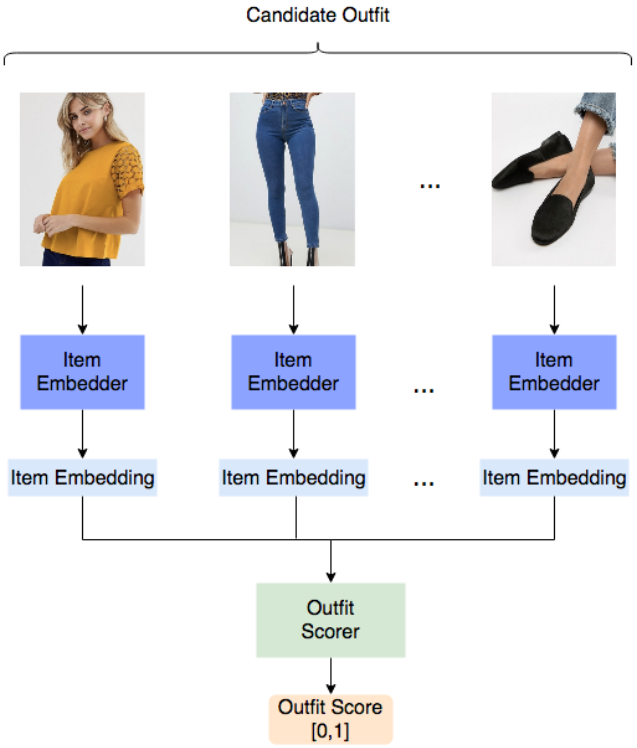


Figure 4: GORDN's outfit scorer takes the learnt embeddings of each item in an outfit and produces a score in the range [0,1], with a high value representing a compatible outfit. The scorer takes the sum of the compatibility scores (dot product) for each pair of items within the outfit, normalised by the number of pairs of items. This is then passed through a sigmoid function.

where σ is the sigmoid function. The normalisation factor of $N(N-1)$, proportional to the number of pairs of items in the outfit is required to deal with outfits containing varying numbers of items. The sigmoid function is used to ensure the output is in the range [0,1].

4.3 Visual Feature Extraction

As described in Section 1 and illustrated in Figure 1, items are photographed as part of an outfit and therefore our item images frequently contain the other items from the BTL outfit. Feeding the whole image to the network would result in features capturing information for the entire input leaking information to GORDN. It was therefore necessary to localise the target item within the image. To extract visual features from the images in our dataset we use VGG [19]. Feeding the whole image to the network would result in features capturing information for the entire input, that is both the hero and the styling products. To extract features focused on the most relevant areas of the image, we adopt an approach based on Class Activation Mapping (CAM) [25]. Weakly-supervised object localisation is performed by calculating a heatmap (CAM) from the feature maps of the last convolutional layer of a CNN, which highlights the discriminative regions in the input used for image classification. The CAM is calculated as a linear combination of the feature maps weighted by the corresponding class weights.

Before using the CAM model to extract image features, we fine-tune it on our dataset. Similar to [25] our model architecture combines VGG with a Global Average Pooling (GAP) layer and an output classification layer. We initialize VGG with weights pre-trained on ImageNet and fine-tune it towards product type classification (e.g. Jeans, Dresses, etc.). After training we pass each image to the VGG and obtain the feature maps.

To produce localised image embeddings, we use the CAM to spatially re-weight the feature maps. Similar to Jimenez et al. [8], we perform the re-weighting by a simple spatial element-wise multiplication of the feature maps with the CAM. Our pipeline is shown in Figure 5. This re-weighting can be seen as a form of attention mechanism on the area of interest in the image. The final image embedding is a 512-dimensional vector. The same figure illustrates the effect of the re-weighting mechanism on the feature maps.

4.4 Title and Description Embeddings

Product titles typically contain important information, such as the brand and colour. Similarly, our text descriptions contain details such as the item's fit, design and material. We use pre-trained text embeddings of our item's title and description. These embeddings are learned as part of an existing ASOS production system that predicts product attributes [3]. Vector representations for each word are passed through a simple 1D convolutional layer, followed by a max-over-time pooling layer and finally a dense layer, resulting in 1024 dimensional embeddings.

4.5 Training

We train GORDN in a supervised manner using a binary cross-entropy loss. Our training data consists of positive outfit samples taken from the ASOS outfits dataset and randomly generated negative outfit samples. We generate negative samples for our training and test sets by randomly replacing the styling products in each outfit with another item of the same type. For example, for an outfit with a top as the hero product and jeans and shoes as styling products, we would create a negative sample by replacing the jeans and shoes with randomly sampled jeans and shoes. We ensure that styling products appear with the same frequency in the positive and negative samples by sampling styling products from their distribution in the positive samples. This is important as the frequency distribution of styling products is heavily skewed (Figure 2) and without preserving this GORDN could memorise frequently occurring items and predict outfit compatibility based on their presence. By matching the distribution GORDN must instead learn the characteristics of items which lead to compatibility. Although some of the negative outfits generated in this way may be good quality outfits, we assume that the majority of these randomly generated outfits will contain incompatible item combinations. Randomly selecting negative samples in this way is common practice in metric learning and ranking problems (e.g. [6, 16]). In both training and testing, we generate one negative outfit sample for each positive outfit sample.

To assess the relative importance of each set of input features, we conduct an ablation study. We separately train five different versions of GORDN using only the textual title and description

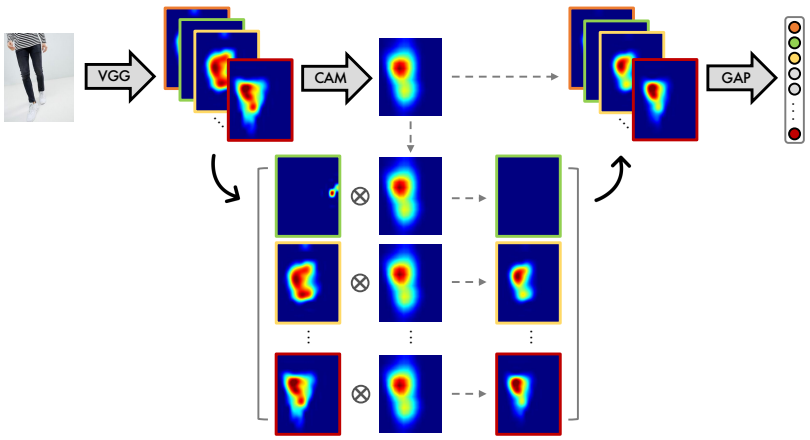


Figure 5: Pipeline for extracting image embeddings. An image is passed to VGG and the final convolutional feature maps are used to calculate the class activation map (CAM). The CAM is then used to spatially re-weight the feature maps by element-wise multiplication. Finally, a global average pooling (GAP) layer averages each re-weighted feature map to calculate a single value (shown in same colours) and outputs a 512-dimensional image embedding. During training, re-weighting is ignored and the output of the model is passed into a fully-connected layer for product type classification. We can see the effect of feature map re-weighting in the brackets for the case of an item with trousers as the hero product and top and shoes as styling products. Activations in the feature maps that correspond to the relevant region of interest in the input image (trousers) are refined by re-weighting (i.e. second and third row) whereas irrelevant activations are ignored (first row).

embeddings as input (text), only the visual embeddings (vis), both text and visual embeddings (text + vis), text, visual and category embeddings (text + vis + cat), and finally the full set of inputs (text + vis + cat + hero). For each of these configurations we trained 20 models from scratch using Adam [9] for 30 epochs.

4.6 Outfit Generation Method

Once trained, GORDN can generate novel outfits of any length by sequentially adding items and re-scoring the new outfit. Each outfit starts with a hero product from our catalogue. We then define an outfit template $\mathcal{P} = \{T^{(h)}, T_1, ..., T_{N-1}\}$ as a set of product types including the hero product type $T^{(h)}$ and $N-1$ other compatible styling product types. Our aim is to find the set of items of the appropriate product types that maximises the outfit score y.

An exhaustive search over every possible combination of styling products cannot be computed within a reasonable e-commerce latency budget. Instead, we map the maximum inner product search in Equation 1 to a Euclidean nearest neighbour problem that is solved approximately and combine this with a beam search (illustrated in Figure 6). The approximate nearest neighbours algorithm uses a PCA-tree that has been adapted for recommendations problems [1]. We use a beam width of three because it returned the optimal outfit 77.5% of the time. The beam search algorithm is repeated for all $(N-1)!$ permutations of the styling product types in the template as different outfits may be generated depending on the order in which product types are added. The outfit returned is the one that has the maximal score across all template permutations. For each step of the beam search, we calculate the resultant vector of the partial outfit and find the w approximate nearest neighbours from the product catalogue (where w is the beam width). With each step searching through 2000-5000 products we achieved a ten times speed up in outfit generation whilst still maintaining a precision@5 of over 80%.

Table 2: The number of outfits and items in our training and test partitions after applying the Louvain community detection method to the full ASOS outfits dataset. There are no items which appear in both the training and test set.

Department	Dataset	Number of Outfits	Number of Items
Womenswear	Train	237,478	239,818
	Test	76,722	81,854
Menswear	Train	201,844	198,947
	Test	70,276	71,106

The choice of template $\mathcal{P}$ depends on the use case. Templates for each hero product type can be found from our ASOS outfits dataset. The distribution of templates can be used to introduce variety into the generated outfits. For the purposes of our AB test, we picked the most frequently occurring template for each hero product type.

5 EVALUATION

We evaluate the performance of GORDN on two tasks. The first task is binary classification of genuine and randomly generated outfits, using a held out test set. The second task is user evaluation of outfits generated by GORDN in comparison to randomly generated outfits from a simple baseline model.

5.1 Train/test split

We split the ASOS outfits dataset first into WW and MW and each of these into a training and test set ensuring that no items appeared

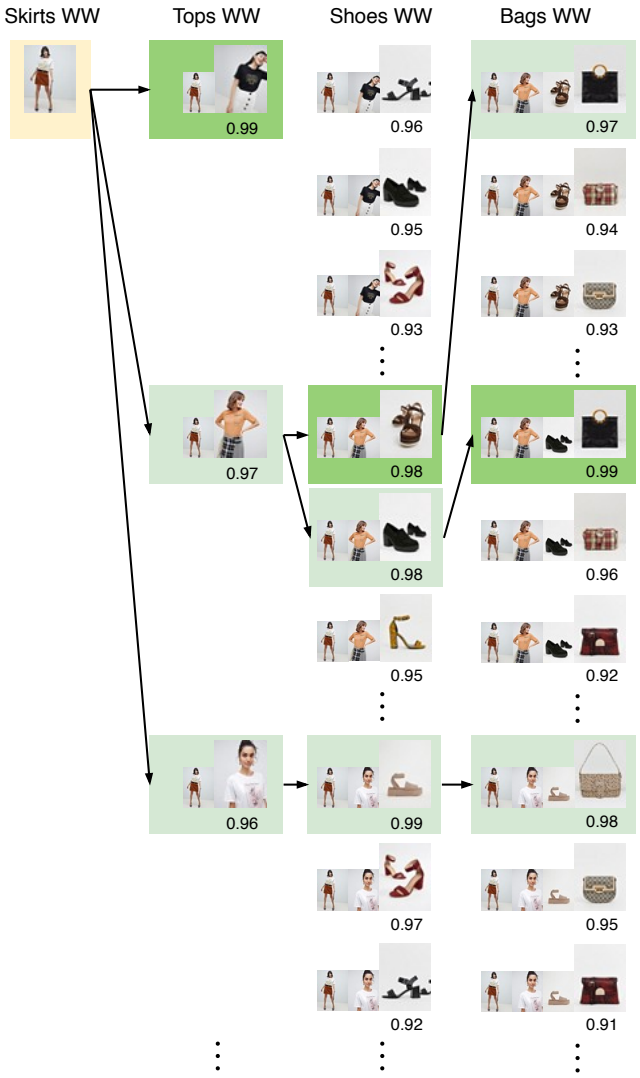


Figure 6: Beam search in the context of outfit generation. Starting with an outfit template of product types and a hero product (highlighted in yellow) each product type in the template is filled sequentially by finding the products from the catalogue which when added to the outfit give the highest outfit score. After each step the number of outfits retained is reduced to the beam width (set to 3). The retained outfits after each step are highlighted in green with the highest scoring outfit in dark green.

in both sets. To achieve this we first represented the ASOS outfits dataset as a graph where the nodes are items and edge weights are defined by the number of outfits pairs of items are found together in. We then used the Louvain community detection method [2] to split the graph into communities which maximise the modularity. This resulted in many small communities which could then be combined together to create the train and test sets. When re-combining communities care was taken firstly to respect the desired train-test split ratio as far as possible and secondly to ensure items from each season are proportionally split between the train and test sets. This resulted in 76:24 and 74:26 train-test splits in terms of outfits for WW and MW respectively. The use of disjoint train and test sets provides a sterner test for GORDN as it is unable to simply memorise which items frequently co-occur in outfits in the training set. Instead, the embeddings GORDN learns must represent product attributes that contribute to fashion compatibility.

Table 3: Comparison of GORDN when using different features on the binary classification task. Scores are the mean over 20 runs of the test set AUC after 30 epochs of training.

Features	AUC	
	WW	MW
vis	0.66	0.55
text	0.80	0.66
text + vis	0.82	0.66
text + vis + cat	0.82	0.67
text + vis + cat + hero	**0.83**	**0.67**

5.2 Outfit Classification Results

The test set contains BTL outfits and an equal number of negative samples. We use GORDN to predict compatibility scores for the test set outfits and then calculate the AUC of the ROC curve. We found that training separate versions of GORDN for WW and MW produced better results and so we report the performance of these here.

Table 3 shows the AUC scores achieved for different combinations of features. As we add features to GORDN we increase its performance, with the best performing model including text, visual, category and hero item features. The majority of the performance benefit came from the text embeddings with visual embeddings adding a small improvement. We expected our visual embeddings to be of poorer quality than those for Polyvore datasets as our images show whole outfits on people as opposed to a photograph of the fashion item in isolation. In contrast the success of our text embeddings could be due to the attribution task on which they were trained [3]. A total of 34 attributes were predicted, including many attributes that are directly applicable for outfit composition e.g. 'pattern', 'neckline', 'dress type' and 'shirt style'.

For all feature combinations the WW model greatly outperforms the MW one. This could be due to fashion items being more interchangeable in MW than in WW hence having more similar embeddings making the training task harder. For example the mean correlations between the text embeddings for the most prevalent product type in the WW and MW training sets are 0.041 (dresses) and 0.077 (T-shirts) respectively. More simply, there are many combinations of MW T-shirts and jeans that make equally acceptable outfits whereas there are far fewer for WW dresses and shoes.

Using GORDN to predict compatibility scores for our test set is equivalent to the *outfit compatibility* task used by [4] and [23]. As noted by Vasileva et al., Han's negative samples contain outfits that are incompatible due to multiple occurrences of product types e.g. multiple pairs of shoes in the same outfit. Since our negative

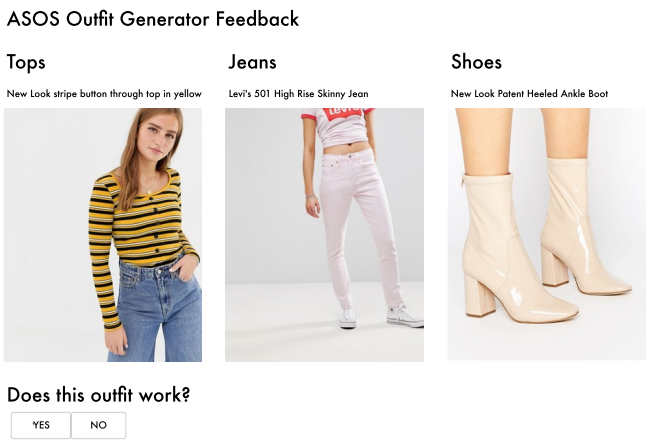


Figure 7: Screenshot of Outfit Evaluation App

samples were generated using templates respecting product type our data does not have this characteristic and hence we compare only to results in [23]. Our WW model achieves an AUC score just slightly less than Vasileva et al.'s compatibility AUC on their disjoint Polyvore outfits dataset.

5.3 Generated Outfit Evaluation

We perform an AB test to evaluate the quality of outfits generated by GORDN. We select six popular outfit templates to test, three each for WW and MW (shown in Table 4), and generate 100 WW and 100 MW outfits split evenly across the templates. We use a large pool of in stock products from which we randomly select hero products of the required product types. The remaining items in the outfits were generated using the beam search method described in Section 4.6 and illustrated in Figure 6. These outfits constitute our test group. For a control group we take the same hero products and templates and generate outfits by randomly selecting items of the correct type from the same pool of products. By using outfit templates we ensure that none of the outfits contain incompatible product type combinations, such as by pairing a dress and a skirt, or by placing two pairs of shoes in one outfit. Instead, the quality of the outfits depends solely on style compatibility between items.

To run the AB test we developed an internal app which we exposed to ASOS employees. A screenshot of the app is shown in Figure 7. The app displayed an outfit to the user asking them to decide if the items in the outfit work stylistically. The outfits were shown one at a time to each user with the order of outfits randomised for each user. WW and MW outfits were only shown to female and male users respectively and each user rated all 200 outfits from their corresponding gender.

The data collected from the app comprised a binary score for each user-outfit pair. The data exhibit two way correlation — all scores from the same user are correlated due to the inherent user preferences and all scores on the same outfit are also correlated. We therefore used a two-way random effects model as described in [17] to calculate the variance of the sample mean. We could then use a t-test for the difference between means to calculate if the difference between the test and control groups was significant.

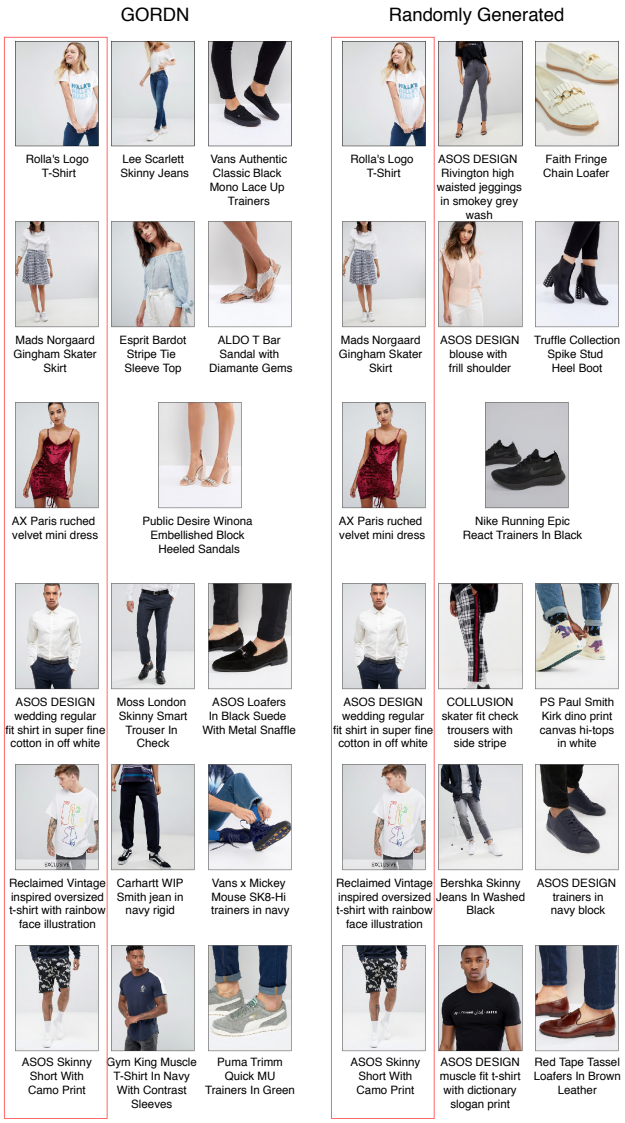


Figure 8: Example outfits generated by GORDN (left column) and by random selection of items (right column) that were used in our AB test. In each row the same hero product is used (red box) and each model is given the same template of product types.

The results are shown in Figure 4. We analyse the results for WW and MW separately as the WW and MW models were trained separately. We collected 1,200 observations per group for WW and 900 for MW. We found the relative difference between the test and control groups to be 21.28% and 34.16% for WW and MW respectively. Testing at the 1% level these differences were significant. We were able to further break down our results to find that GORDN outperformed the control significantly for all templates.

Examples of outfits generated for our AB test are shown in Figure 8. For each hero product we show the outfit produced by GORDN alongside the randomly generated outfit. Many of the

Table 4: Relative differences between the test and control group user scores. All results are significant at the 1% level.

		Ctrl score	Test score	Rel. diff. (%)	p-value
WW	all	0.49	0.60	21.28	< 0.01
	Dress | Shoes	0.54	0.78	46.12	< 0.01
	Tops | Jeans | Shoes	0.61	0.64	4.53	< 0.01
	Skirts | Tops | Shoes	0.33	0.36	10.77	< 0.01
MW	all	0.49	0.66	34.16	< 0.01
	T-Shirts | Jeans | Shoes, Boots & Trainers	0.63	0.76	19.07	< 0.01
	Shirts | Trousers & Chinos | Shoes, Boots & Trainers	0.42	0.60	44.35	< 0.01
	Shorts | T-Shirts | Shoes, Boots & Trainers	0.43	0.63	47.24	< 0.01

random examples appear to be reasonable outfits. Although the random model is simple, the use of outfit templates, combined with selecting only products that were in stock in the ASOS catalogue on the same day makes this a challenging baseline.

5.4 Style space

We visualise our style space using a t-Distributed Stochastic Neighbour Embedding (t-SNE) [22] plot in two dimensions (Figure 9a). While similar items have similar embeddings, we can also see that compatible items of different product types have similar embeddings. Rather than dresses and shoes being completely separate in style space, these product types overlap, with casual dresses having similar embeddings to casual shoes and occasion dresses having similar embeddings to occasion shoes. We built an app for internal use that uses t-SNE to visualise our style space and allows us to easily explore compatible item combinations, as predicted by GORDN (Figure 9b).

6 CONCLUSION

We have described GORDN, a multi-modal neural network for generating outfits of fashion items, currently under development at ASOS. GORDN learns to represent items in a latent style space, such that compatible items of different types have similar embeddings. GORDN is trained on the ASOS outfits dataset, a new resource for the research community which contains over 500,000 outfits curated by professional stylists. The results of an AB test show that users approve of outfits generated by GORDN 21% and 34% more frequently than those generated by a simple baseline model for womenswear and menswear, respectively.

REFERENCES

[1] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 257–264.

[2] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *J. Stat. Mech* (2008), 1–12.

[3] Ângelo Cardoso, Fabio Daolio, and Saúl Vargas. 2018. Product Characterisation towards Personalisation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*. 80–89.

[4] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. 2017. Learning Fashion Compatibility with Bidirectional LSTMs. *Proceedings of the 2017 ACM on Multimedia Conference - MM '17* 1 (2017), 1078–1086.

[5] Tong He and Yang Hu. 2018. FashionNet: Personalized Outfit Recommendation with Deep Neural Network. (2018), 1–9.

[6] Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

[7] Yang Hu, Xi Yi, and Larry S. Davis. 2015. Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach. In *Proceedings of the 23rd ACM international conference on Multimedia*. Brisbane, Australia, 129–138.

[8] Albert Jimenez, Jose M. Alvarez, and Xavier Giro-i Nieto. 2017. Class-Weighted Convolutional Features for Visual Instance Search. In *28th British Machine Vision Conference (BMVC)*.

[9] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Internationl Aconference for Learning Representations*. San Diego.

[10] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. 2017. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia* 19, 8 (2017), 1946–1955.

[11] Yihui Ma, Jia Jia, Suping Zhou, Jingtian Fu, Yejun Liu, and Zijian Tong. 2017. Towards better understanding the clothing fashion styles: A multimodal deep learning approach. *AAAI Conference on Artificial Intelligence* (2017), 38–44.

[12] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. In *SIGIR Converence on Research and Development in Information Retrieval*. 43–52.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems*. 3111–3119.

[14] Takuma Nakamura and Ryosuke Goto. 2018. Outfit Generation and Style Extraction via Bidirectional LSTM and Autoencoder. In *The third international workshop on fashion and KDD*.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Conference on Uncertainty in Artificial Intelligence*, Vol. 1120. 452–461.

[17] Flâvio Ribeiro, Dinei Florêncio, Cha Zhang, and Michael Seltzer. 2011. CROWDMOS: An approach for crowdsourcing mean opinion score studies. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2416–2419.

[18] Yong-Siang Shih, Kai-Yueh Chang, Hsuan-Tien Lin, and Min Sun. 2018. Compatibility Family Learning for Item Recommendation and Generation. In *AAAI Conference on Artificial Intelligence*. 2403–2410.

[19] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).

[20] Xuemeng Song, Fuli Feng, Xianjing Han, Xin Yang, Wei Liu, and Liqiang Nie. 2018. Neural Compatibility Modeling with Attentive Knowledge Distillation. In *SIGIR Conference on Research & Development in Information Retrieval*. 5–14.

[21] Pongsate Tangseng, Kota Yamaguchi, and Takayuki Okatani. 2018. Recommending Outfits from Personal Closet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 269–277.

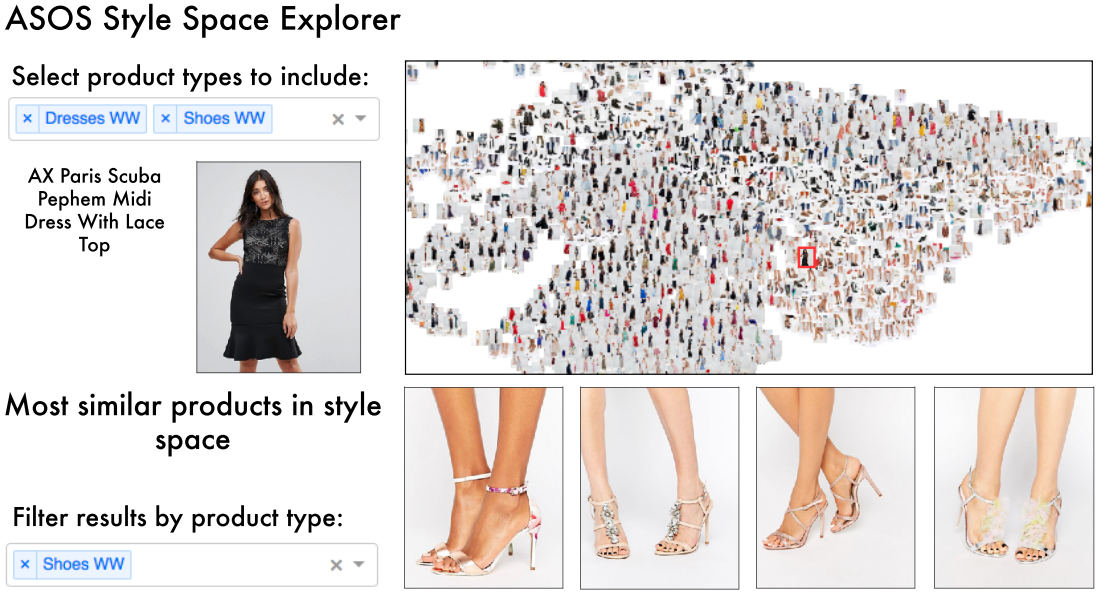
[22] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[23] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. 2018. Learning Type-Aware Embeddings for Fashion Compatibility. In *European Conference on Cumputer Vision*. 405–421.

[24] Andreas Veit, Balazs Kovacs, Sean Bell, Julian Mcauley, Kavita Bala, and Serge Belongie. 2015. Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences. In *IEEE International Conference on Computer Vision*. 4642–4650.

[25] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning Deep Features for Discriminative Localization. In *Computer Vision and Pattern Recognition*.

(a)



(b)

Figure 9: a) A section of a t-Distributed Stochastic Neighbour Embedding (t-SNE) visualisation of the embeddings learnt by GORDN for womenswear dresses and shoes. Similar items have similar embeddings, but so do compatible items of different types. The two highlighted areas illustrate that casual dresses are embedded close to casual shoes (red), while occasion dresses are embedded close to occasion shoes (blue). b) Screenshot of an internal app, developed to allow exploration of the learnt style space. Users can create t-SNE plots using different product types and then select individual items to view the most similar items in style space of different types.