Learning Representations for Fashion Style

Bryan Anenberg
Stanford University
anenberg@stanford.edu

Abstract

We present a system that directly learns a compact feature representation for fashion photographs taken in real-world environments. The learned representation is shown to encode an understanding of fashion style despite not directly being trained to classify between fashion styles. The representation is learned through training on weakly labeled data from fashion photographs posted to online social websites. The system learns a style similarity metric by comparing whether pairs of images share attributes – colors and garments. In particular, training is performed using triplets that consist of an anchor image, a dissimilar image, and a similar image. The performance of the learned embedding is thoroughly evaluated and is shown to be generic in that it can be used on tasks it was not directly trained for, such as fashion style classification.

1. Introduction

Clothing serves for much more than covering and protection. The clothing that we wear provides a visual signal that we can use to make predictions about the person. The style of ones clothing reflects their personality, their interests, their occupation, and even their socioeconomic status.

The Internet is full of images of people wearing different styles of clothing. In particular, there exist websites such as http://www.chictopia.com/ and Instagram that contain thousands of images of fashion photography. These sources contain images across a wide variety of fashion styles. Fashion posts often are annotated with a variety of tags describing the style, season, garment, or event. Posts also often contain comments and counters for the number of likes and shares.

In this paper we present a system that directly learns a mapping from fashion photographs taken in real-world environments to a compact Euclidian space where distances correspond to a measure of clothing style similarity. The method uses a deep convolutional neural network optimized to learn the feature embedding and similarity metric space directly. Training is performed using triplets that consist of an anchor image, a dissimilar image, and a similar image. Images are weakly labeled by the presence or absence of clothing attribute such as color and clothing type. The labels are noisy and not guaranteed to be exhaustive. The image triplet is selected by comparing the clothing attributes that have in common. The network is also jointly trained to perform clothing attribute prediction, which encourages the network to learn features that discriminate between imaged based on the clothing attributes rather than on the background or the person. Another goal of this paper is to demonstrate that the learned compact features are generic and powerful enough to be useful for other clothing related tasks such as attribute prediction, style classification, and even fashionabilty prediction.

An example photograph that we could consider to train the system is shown here 1. The photo contains a single model and is weakly labeled with a variety of metadata. For example, some tags describe the style such as "Chic", "Everyday," or "Fall," the garment color such as "red" or "black", and the garment type such as "sunglasses," "boots", or "watch".

Figure 1: An example post from the Fashion144k dataset. The post contains at least a single image with additional metadata including tags and comments.

2. Previous work

There is a long history of prior work related to clothing recognition [8] [1] [15], attribute prediction [3] [9] [4], clothing item retrieval, and clothing segmentation. This area of research is especially interesting because of the many real-world consumer applications.

Clothing recognition aims to detect and identify articles of clothing. Often clothing can be described with varying levels of precision. For example, clothing can be coarsely described by categories such as "dress", "pants", "suit", "shoes", and "sweater". Fine-grain description of clothing involves identifying attributes such as color, texture, pattern, fabric, shape, or style. The complete dictionary of clothing attributes is very large. Often a single article of clothing can be describe by many clothing attributes. For example, [1] performs clothing recognition by first classifying the image as one of 15 common clothing types and then identifying the set of attributes form a collection of 78 attributes that are present in the image. The complete description of the clothing can be given by the union of the type and attribute, such as "blue dress." [3] generates a list of attributes for clothes worn by humans in unconstrained images. His model first estimates the person's pose, then predicts each attribute using individual attribute classifiers relying on complementary features, and makes a final prediction by leveraging a CRF to enforce the Rules of Style constraints across the preliminary attribute predictions.

Another example of clothing attribute prediction is [9] in which they preform discriminative classification between 5 style categories: hipster, hipster, bohemian, pinup, preppy, and goth.

However, as pointed out in [17], clothing recognition algorithms are especially difficult to operate in practice due to three primary challenges. First, clothes often have large variation in style, texture, and cut. Second, clothes are often subject to deformation and occlusion. Third, clothing images often exhibit sever variations when they are taken under different scenarios such as selfies vs. online shopping photos.

Many prior works address these challenges through transfer learning by first pre-training on a large well annotated related dataset and fine-tuning on the well annotated clothing dataset. [4] presents a domain adaptation technique to leverage a model trained on a large scale shopping dataset annotated with fine-grain clothing attributes to perform clothing attribute recognition in real-world surveillance videos. The authors observe that there exist lots of structured descriptions of clothes on e-Commerce websites such as Amazon.com, but very fewer cleanly annotated datasets of people wearing clothes in the natural environment. Images of clothes drawn from e-Commerce websites have ideal lighting, standard pose, high resolution, and good quality whereas images of clothes worn in the natural world are subject to deformation, occlusion, and all the other challenges recounted above.

Other authors approach the problem of the scarcity of well annotated real-world clothing images by adapting their model to train on images with mixed quality labels. The authors of [15] assert that training a model from scratch with limited clean labels and an abundance of noisy labels is better than only fine-tuning a pre-trained network on the clean labels. They present a system that layers a probabilistic graphical model on top of a CNN to perform clothing classification.

Many of these issues could be addressed through the introduction of large scaled annotated datasets. However, until recently there were no large scale datasets that addressed all of the above challenges with rich annotations. The large-scale fashion (DeepFashion) dataset introduced in [17] is perhaps the most diverse and well annotated fashion dataset.

The list of datasets related to clothing and fashion that I have encountered are as follows:

- Hipster Wars [9]: The hipster wars dataset contains 1893 photos modeling five clothing style categories: hipster, hipster, bohemian, pinup, preppy, and goth.
- Large-scale Fashion (DeepFashion) Database [17]: The DeepFashion dataset contains 800,000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos. Each image in this dataset is labeled with 50 categories, 1,000 descriptive attributes, bounding box and clothing landmarks.
- Apparel classification with Style dataset [1]: This dataset contains over 80,000 images across 15 classes.
- Clothing1M [15]: This dataset contains 1,000,000 images from online shopping websites. The majority of the images are assigned a noisy label according to keywords in the surrounding text (only 72,409 images are considered clean).
- Fashion 144k [12]: This dataset contains 144,169 user posts with images and their associated metadata.

In Neuroaesthetics of Fashion: Modeling the Perception of Fashionability [12], the authors aim to predict how fashionable a person looks in a photograph and suggest subtle improvements that the user can make to improve his or her style. They implement a CRF model that relies on several fashionability factors including the clothing attributes, user characteristics, the surrounding scenery, and the fashionability scores, which is derived from the likes and comments of the post.

In Fashion Style in 128 Floats [13] the author attempts to directly learn a mapping from fashion photography to a compact Euclidian space where distances directly correspond to a measure of style similarity. He proposes a multi-task CNN with a ranking loss on the 128 dimensional feature embedding to encourage similar images to have small Euclidian distance and different images to have large Euclidian distance. A classification loss for the task of clothing attribute prediction is also added. The model is trained on a filtered subset of the Fashion144k dataset. Fashion

photos are labeled by the presence or absence of attributes from a large collection of attributes. The images are weakly labeled - the labels are noisy and are not guaranteed to contain the full set of attributes that best describe the image. The work of our paper is directly inspired by [13].

2.1. Neural Networks to Learn Similarity

Optimizing a neural network to learn a similarity metric, rather than indirectly learn the metric as a byproduct of training a classification network, is an approach that has been applied to a various tasks such as wide angle matching and facial recognition [6] [16] [11]. Two popular network architectures are the Siamese network and the triplet network. A Siamese network [2] consists of two neural network towers and a loss function that reflects the similarity of the two input images. A triplet network [7] consists of three neural network towers to process three images, an anchor image, a similar image, and a dissimilar image. The triplet loss tries to learn the relative similarity between the images. The authors of FaceNet [11] successfully employ a triplet loss to learn a metric space that they use for facial recognition and clustering. Another approach to learning a similarity metric space is present in [14]. Here the authors lift the vector of pairwise distances within the batch to the matrix of pairwise distance.

2.2. Semantic Embedding

Other interesting works that combine vision and language models to learn semantic embeddings include [10] and [5].

3. Technical Part

The proposed model is heavily inspired by [13] and [11].

The model consists of jointly training a feature embedding network and a classification network. The goal of the feature embedding network is to directly learn the compact Euclidian space where distance is measure of fashion style similarity. This network is trained using image triplets.

The classification network adds additional layers to the feature embedding network to predict the presence or absence of clothing attributes in an image.

3.1. Triplet Selection

We use the same methodology as [13] to select triplets. Each fashion photograph is weakly labeled by a set of attributes or "tags". A label vector $l = (l^t)_{t \in T}$ for an image assigns a $l^t \in 0, 1$ to each tag $t \in T$. If the tag applies to the image, the label for the image assigns 1 to the tag.$|l|$ is the number of tags that label l assigns 1. The similarity function between labels a and b is defined as the intersection over union of the label vectors a and b:

$$r(a, b) = \frac{|a \wedge b|}{|a \vee b|}$$

where $\wedge$ and $\vee$ operate on the labels as tag-wise AND and OR respectively.

Triplets are selected by first fixing an anchor image, and then selecting a similar and dissimilar image to add along side it to the triplet. The label vectors of the anchor image, similar image, and dissimilar image are given by y, y_+, and y_- respectively. The label vectors are a 3,302 dimensional binary vector. The similar image I_+ is chosen such that $r(y, y_+) > \tau_s$ where τ_s is a similarity threshold. (In the experiments this is set to 0.75). The dissimilar image I_- is chosen such that $r(y, y_-) < \tau_d$ where τ_d is the dissimilar image threshold. (In the experiments this is set to 0.1). The image triplet is written as (I_-, I, I_+). An example image triplet is shown here 2.

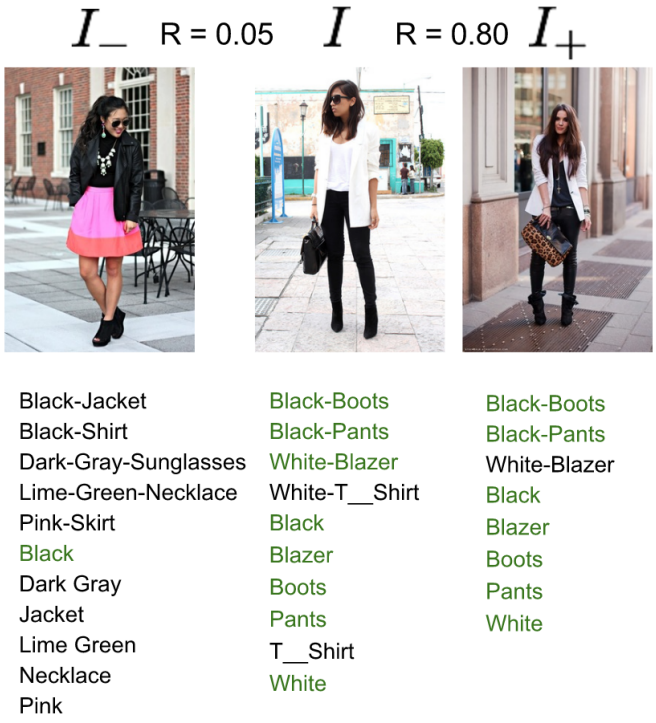

Figure 2: An example image triplet used during training. The images are ordered as dissimilar, anchor, similar image. The attributes shared between the similar and anchor image are highlighted in green. Only a single attribute is shared between the dissimilar and anchor image. R reflects the intersection over union tag-wise similarity score calculated between the images. Notice that not all of the attributes present in each image are displayed in the figure.

In order to ensure fast convergence when training, it is necessary to partially precompute the image triplets. It is quite rare for a pair of images to exceed the similarity threshold. It often requires sampling many pairs of images before finding a pair whose tag-wise similarity score r exceeds the similarity threshold. For this reason, the similarity

score between all $\binom{N}{2}$ images is precomputed and those pairs with similarity score greater than τ_s are stored. The triplet is completed at training time by sampling from the dataset a third image whose similarity score with at least one of the images in the precomputed similar pair is less than τ_d.

3.2. Embedding Loss

We use either the ranking loss of [13] or the triplet loss of [11]. Both losses operate on the triplet of feature embedding vectors that we refer to as $T_f = (f_-, f, f_+)$.

3.2.1 Ranking Loss

As described in [13], the ranking loss $l_{Ranking}$ encourages the normalized distance d_+ between the anchor and the similar image to be smaller than the distance d_- between the anchor and the dissimilar image.

$d_- = \frac{\exp(||f_- - f||_2)}{\exp(||f_- - f||_2)+\exp(||f_+ - f||_2)}$

$d_+ = \frac{\exp(||f_+ - f||_2)}{\exp(||f_- - f||_2)+\exp(||f_+ - f||_2)}$

$l_{Ranking}(f, f_-, f_+) = 0.5\left((d_+)^2 + (1 - d_-)^2\right) = (d_+)^2$

3.2.2 Triplet Loss

As described in [11], the triplet loss is a variant of the hinge loss which encourages the d_+ between the anchor and the similar image to be smaller than the distance d_- between the anchor and the dissimilar image by at least some parameter α. α is a tunable parameter, but for our experiments we left it fixed at $\alpha = 0.2$.

$l_{Triplet}(f, f_-, f_+) = \max(0, ||f_+ - f||_2^2 + \alpha < ||f_- - f||_2^2)$

3.3. Feature Embedding Network

The convolutional neural network to obtain the feature embedding vector is inspired by [13]. The goal of the feature embedding network is to learn a compact (128 dimensional) representation of input image. This motivates the networks relatively shallow depth. Most fashion photography has $3 : 4$ aspect ratio. The network reflects detail by expecting a rectangular shaped input image 1.

3.4. Classification Network

The classification network is shallow fully connected network operating on the output of the feature embedding network. The objective of the classification network is to predict the presence or absence of the clothing attributes associated with the image. Recall that to construct the training image triplet the tag-wise similarity metric $r(a, b)$ was computed between image pairs. The label vector used to compute the similarity metric was $3,302$ dimensional. The binary vector l is a 123 dimensional subset of the full label vector consisting of tags that appear with more frequency. The classification network is trained with sigmoid cross entropy loss to predict the binary vector l. 2

During training, the classification network is only applied to the dissimilar image in the image triplet. This is because the dissimilar image was sampled at random from the full training dataset, whereas the anchor and similar image are drawn from a smaller precomputed set of similar images.

layer	kernel size	output size
convolution	3×3	$384 \times 256 \times 64$
convolution	3×3	$384 \times 256 \times 64$
dropout(25%)		$384 \times 256 \times 64$
max pooling	4×4	$96 \times 64 \times 64$
batch normalization		$96 \times 64 \times 64$
convolution	3×3	$96 \times 64 \times 128$
convolution	3×3	$96 \times 64 \times 128$
dropout(25%)		$96 \times 64 \times 128$
max pooling	4×4	$24 \times 16 \times 128$
batch normalization		$24 \times 16 \times 128$
convolution	3×3	$24 \times 16 \times 256$
convolution	3×3	$24 \times 16 \times 256$
dropout(25%)		$24 \times 16 \times 256$
max pooling	4×4	$6 \times 4 \times 256$
batch normalization		$6 \times 4 \times 256$
convolution	3×3	$6 \times 4 \times 128$
fully-connected		128

Table 1: The feature embedding network. The output feature vector is 128 dimensional.

layer	output size
batch normalization	128
fully-connected	128
fully-connected	123

Table 2: The classification network expects a 128 dimensional extracted by the feature embedding network. The output is a 123 vector of scores across the 123 attribute classes the network aims to predict.

4. Experiments

4.1. Datasets

The model was trained on a filtered subset of the Fashion144k dataset `http://hi.cs.waseda.ac.jp/~esimo/en/research/fashionability/` which consists of fashion posts gathered from `http://www.chictopia.com/`. Images taken with strong filters, close-ups of objects, or that were severely cropped were removed [13]. The training split consisted of 80,554 images. The test split contained 8,948

images.

The HipsterWars dataset `http://www.cs.unc.edu/~hadi/hipsterwars/` contains fashion photos belonging to five style classes: "Bohemian", "Goth", "Hipster", "Pinup", and "Preppy." This dataset was used to evaluate the learned embedding's performance on the task of style classification.

4.2. Triplet Selection

Selecting image triplets proved very time intensive. For this reason we precomputed all possible image pairs whose tag-wise similarity $r(a, b)$ was greater than the threshold, 0.75. This resulted in 61,951 similar image pairs in the training split and 832 similar pairs in the test split. The dissimilar image to complete the image triplet was sampled at random from the dataset at training time.

4.3. Joint Training

The model was trained jointly with the either the Triplet or Ranking embedding loss and the classification sigmoid cross entropy loss. The classification network provided a monotonically decreasing loss whereas the embedding loss often was more noisy. This observation could have been due to the fact that solving the binary attribute prediction problem is easier (in that there is clear annotation) than learning a relative ordering between image pairs in the triplet.

The embedding loss and the classification loss are weighted according to the parameter β, which is tuned in the experiments.

$l = \beta l_{classification} + (1 - \beta) l_{embedding}$

We decided to pretrain the weights of the embedding network on the classification task by adding a single fully connected layer on top of the embedding network. Pretraining the weights of the embedding network should help the embedding loss avoid diverging.

4.4. Metrics

To evaluate the quality of the learned representations we compute a variety of metrics.

4.4.1 Metrics to Evaluate Embedding Quality

The first set of metrics we compute are used to evaluate the embedding vectors themselves. These metrics were inspired by [11]. Given a pair of two images x_i and x_j, the squared L_2 distance is computed $D(x_i, x_j)$. The pair is classified as either "similar" or "dissimilar" depending on the value of $D(x_i, x_j)$. Image pairs of that are actually similar (using the tag-wise similiarity metric $r(\cdot, \cdot)$) are denoted by $P_{similar}$ and image pairs that are actually dissimilar are denoted by $P_{dissimilar}$. We define the set of all true accepts as

$TA(d) = \{(i, j) \in P_{similar}, \text{with } D(x_i, x_j) \leq d\}$

which are all similar image pairs (i, j) that were classified correctly according to some threshold d. The false accepts are defined asthe set of all dissimilar image pairs (i, j) that were classified as similar according to the threshold d.

$FA(d) = \{(i, j) \in P_{dissimilar}, \text{with } D(x_i, x_j) \leq d\}$

The validation rate $VAL(d)$ and the false accept rate for a given threshold d are defined as

$VAL(d) = \frac{|TA(d)|}{|P_{similar}|}$

$FAR(d) = \frac{|FA(d)|}{|P_{dissimilar}|}$

The threshold d reflects the maximum L_2 distance $D(x_i, x_j)$ required to classify an image pair x_i and x_j as similar.

The threshold d is not chosen arbitrarily, it is learned. We have both a training and test set of similar and dissimilar images. The false accept rate is computed on the training dataset across a range of thresholds (in our experiments, from 0 to 20 in steps of 0.01). The threshold at the targeted false accept rate (in our experiments we held this fixed at 1×10^{-3}) is selected. This threshold is then used to compute the validation rate and false accept rate across the test set of similar and dissimilar image pairs.

Other metrics we computed included the accuracy, true positive rate (Recall), and false positive rates.

$Recall(d) = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}|}$

$FalsePositiveRate(d) = \frac{|\text{false positives}|}{|\text{false positives}| + |\text{true negatives}|}$

$Accuracy(d) = \frac{|\text{true positives}| + |\text{false negatives}|}{|P_{similar}| + P_{dissimilar}|}$

These metrics were also computed with respect to a learned threshold d. This threshold is found in a similar way to before by computing the accuracy at various threshold values (in our experiments, from 0 to 20 in steps of 0.001) on the training dataset. The threshold that yields the greatest accuracy is selected to compute the metrics on the test set of similar and dissimilar image pairs.

4.4.2 Metrics to Evaluate Attribute Prediction

To evaluate the performance of the classification network on the binary label attribute prediction task, we compute the precision, recall, hamming score, F1 score, and top-k recall.

A forward pass of the classification network is ran to generate a 123 dimensional vectors that represents the unnormalized score of presence of each binary attribute in the image. To generate the prediction, the logistic function is applied to each score vector and thresholded at 0.6. (The selection of 0.6 is entirely arbitrary.) The precision, recall, F1 score, and hamming score of the correct predictions is computed with respect to the number of correct binary attribute predictions made.

$HammingScore = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}| + |\text{false positives}|}$

The top-k recall is inspired by [17]. The top-k recall for an image is calculated by ranking the 123 unnormal-

Loss Type	Acc	TPR	FPR	VAL
FAR				
Triplet	0.68	0.67	0.31	0.01
Ranking	0.5	0	0	0

Table 3: Evaluating the quality of the learned embeddings for the network trained with triplet loss and ranking loss. Acc = Accuracy, TPR = True Positive Rate, FPR = False Positive Rate, VAL = Validation Rate, FAR = False Accept Rate.

Loss Type	Precision	Recall	F1	Hamming Score
Triplet	0.079	0.142	0.098	0.054
Ranking	0.38	0.053	0.092	0.053

Table 4: Comparison of triplet loss vs ranking loss for the task of attribute prediction.

Loss Type	1	2	3	4	5
Triplet	0.13	0.105	0.107	0.078	0.084
Ranking	0.51	0.39	0.327	0.305	0.276

Table 5: Comparison of triplet loss vs ranking loss evaluated on attribute prediction using top-k recall.

ized attribute prediction scores and determining how many attributes have been matched in the top-k list.

4.5. Comparing Triplet Loss vs. Ranking Loss

We compare the performance of Triplet loss vs. Ranking loss when jointly training the network with the classification loss.

The triplet loss demonstrates significantly better scores than the ranking loss with respect to the metrics that evaluate the quality of the learned embeddings. This could be due to the fact that the triplet loss directly optimizes the separation of similar and dissimilar images with respect to squared L_2 distance whereas the ranking loss operates with respect to normalized distances and only encourages the relative ranking the image pairs – namely that the distance between the anchor and similar image is less than the distance between the anchor and dissimilar image.

We report the performance of either network evaluated on attribute prediction 45. We observe drastic differences in the attribute prediction performance when using each loss even though the classification loss remains constant. The likely reason for why we observe this result is that the network trained with Ranking loss was allowed to train for more iterations and as a result the classification loss was able to slowly converge. The convergence for the classification loss was rather long since the classification loss weight parameter was set to 0.1.

We also performed a few experiments tuning the classification loss weight.

Figure 3: An exemplary image from each of the five classes in the Hipster Wars dataset.

Network	Accuracy	Precision	Recall	F1
Fashion-Triplet	0.34	0.30	0.34	0.29
Fashion-Ranking	0.41	0.36	0.41	0.37
VGG16	0.62	0.63	0.62	0.62

Table 6: Comparison of the quality of features extracted from Fashion network vs. VGG network for task of fashion classification.

4.6. Evaluating on the Hipster Wars dataset

In order to evaluate the quality of the learned representation we experiment with training the network on the Fashion144k dataset, fixing the weights, and using the network as a feature extractor for the task of fashion style classification on the Hipster Wars dataset. This task involves classifying an image as one of five classes : Hipster, Bohemian, Goth, Preppy, or Pinup. An example image is shown here 3.

We compare the performance of our network, which we refer to as FashionNet, to that of pretrained VGG16 network.

After extracting the feature vector, we use 5-fold cross validation with a 9:1 train-test split to train a SVM with a variety of hyperparamters. The SVM uses L_2 regulariztion and L_2 loss. We consider linear kernel with $C \in [1, 10, 100, 1000]$ and the RBF kernel with $C \in [1, 10, 100, 1000]$ and $\gamma \in [1e-2, 1e-3, 1e-4, 1e-5]$.

Notice that the results in the table 6 indicates that the VGG network has learned better representations than the Fashion Networks. Although this is discouraging, this is likely not the most accurate result because at the time that this experiment was performed the Fashion networked were experiencing a bug with their training. I could not include the results from the most recent networks that were trained because of a issue loading the pretrained weights in Tensorflow. As a next step I would like to follow up on this result.

4.7. t-SNE evaluation

We use t-SNE to visualize the embedding learned by the Fashion Network. We ran ran t-SNE on images sampled from the Fashion144k dataset. The visualization is displayed below 4.

Figure 4: A t-sne embedding of images sampled from the Fashion 144k dataset.

It is difficult to evaluate the t-SNE embedding to determine what structure the representation has learned. This is likely due to the fact that at the time of constructing this t-SNE plot, the network was experiencing an bug with training. I unfortunately could not visualize t-SNE with one of the more recent networks that were trained because of a issue loading the pretrained weights in Tensorflow.

5. Conclusion

In this paper we presented a system that directly learns a mapping from fashion photographs taken in real-world environments to a compact Euclidian space where distances correspond to a measure of clothing style similarity. We demonstrated that the embeddings encode information of similarity with respect to the clothing attributes. We also attempted to demonstrate that the representations are generic and can be used as a pre-trained feature for other related tasks such as style classification. Unfortunately due to technical challenges with Tensorflow, not all of the most recent results were included. Future work will involve improving these results and visualizations.

References

[1] Lukas Bossard, Matthias Dantone, Christian Leistner, Christian Wengert, Till Quack, and Luc Van Gool. Apparel classification with style. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part IV*, ACCV'12, pages 321–335, Berlin, Heidelberg, 2013. Springer-Verlag. `http://people.ee.ethz.ch/~lbossard/projects/accv12/index.html`.

[2] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Sckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *In NIPS Proc*, 1994.

[3] Huizhong Chen, Andrew Gallagher, and Bernd Girod. *Describing Clothing by Semantic Attributes*, pages 609–623. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[4] Q. Chen, J. Huang, R. Feris, L. M. Brown, J. Dong, and S. Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5315–5324, June 2015.

[5] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013.

[6] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, June 2015.

[7] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. *CoRR*, abs/1412.6622, 2014.

[8] Yannis Kalantidis, Lyndon Kennedy, and Li-Jia Li. Getting the look: Clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, pages 105–112, New York, NY, USA, 2013. ACM.

[9] Alexander C. Berg Tamara L. Berg M. Hadi Kiapour, Kota Yamaguchi. Hipster wars: Discovering elements of fashion styles. In *European Conference on Computer Vision*, 2014.

[10] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[12] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. Neuroaesthetics in Fashion: Modeling the Perception of Fashionability. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] Edgar Simo-Serra and Hiroshi Ishikawa. Fashion Style in 128 Floats: Joint Ranking and Classification using Weak Data for Feature Extraction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. *CoRR*, abs/1511.06452, 2015.

[15] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[16] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *CoRR*, abs/1504.03641, 2015.

[17] Shi Qiu Xiaogang Wang Ziwei Liu, Ping Luo and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. `http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html`.