AfroNet: Shopping Afrowear from Anywhere

Oluwasanya Awe
oawe@stanford.edu

Robel Mengistu
robel@stanford.edu

Vikram Sreedhar
vsreed@stanford.edu

March 23, 2018

Abstract

Given a street image of Afrowear product, we seek to retrieve visually similar stock images of Afrowear products that match the apparel category and swatch style of the queried image. Street images pose problems in training given large variations in lighting, background and presence of occluding etc. In our model, we use triplet loss to fine-tune ResNet50 model which will be used as a feature extractor. After getting feature embeddings, we use K-Nearest Neighbors with Euclidean Metric to return top eight visually similar stock images. Using our best network architecture and hyperparameters, we were able to achieve a category matching MAP of 0.823 and histogram correlation MAP of 0.438.

1 Introduction

The African fashion industry is projected to be a $\$15 - \$30$ billion industry by 2019[2][4]. The demand for African clothing and footwear is currently split between local Africans and the diaspora where the Diaspora's buying power alone is valued at $162 billion. Given that most affordable African wear is created within Africa, there is a huge opportunity in the space that streamlines transactions between the Diaspora and local tailors.

In a typical afrowear retail experience, a customer will send a picture of an afrowear apparel they would like to order to their tailor/designer, and expect to get a piece that matches the one in the picture. However, delivering on this expectation is limited by the swatch/fabrics availbility and tailors' ability to replicate the product in customers' inspiration pictures.

We propose to build a model that inputs street images (customer inspiration images that could come from anywhere) and outputs top eight visually similary stock images (products designers can deliver on). Our solution can adjust customers' expecation and be used as an inspiration tool.

1.1 Problem Definition

Specifically, our model takes as input, a 299×299 color image containing at least one person posing with an African clothing product such as a floral jacket or dress against an arbitrary background under any lighting and outputs an ordered list of $K = 8$ images that are most similar to the input image based on one of 14 categories (women's jackets, men's pants etc.) and swatches/fabric pattern (color, pattern, shapes, etc.)

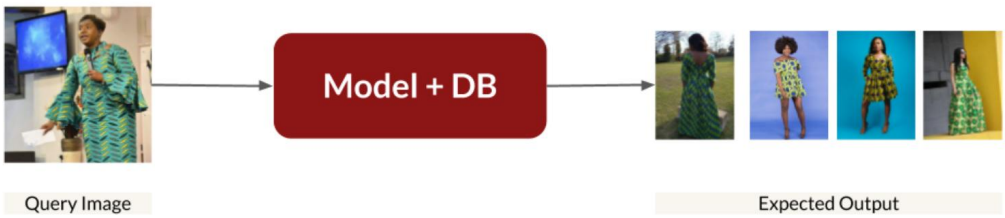

Figure 1: A high-level representation of our problem.

2 Related Work: Image Similarity and Retrieval

Kiapour et. al. explored a variety of models that matched photos of clothing taken from the wild to stock images of similar-looking clothes in online shops [6]. They first used AlexNet as a feature extractor to get a 4096-dimensional vector and measured similarity using the cosine similarity function. They alternatively modeled the problem as a binary classification task with a pair of inputs such that their model returned whether or not the pairs matched.

Shankar et. al. used a Deep CNN architecture to learn various features such as detail-based and concept-based similarities among E-Commerce clothing to outperform the state-of-the-art in image similarity with a 13%

improvement [10]. Similar to Wang et. al. [12], they used a pair of triplet losses by having in-class negatives and out-of-class negatives to train their network to learn the nuances among items that might appear to look similar but are not (e.g. based on thickness of stripes on shirts). Additionally, the out-of-class loss helped the network learn more coarse-grained differences (e.g. differences in plain shirt colors). This turned out to be a clever method which inspired our approach. Along the same line, we adopt Schroff et al.'s technique of normalizing the final layer of their Triplet Network to make the optimal margin invariant to the size of the final layer [9].

In terms of evaluation of our results, we found that using semantic descriptions of images similarly to how Frejlichowski et. al. [3] would be a useful method of checking whether our outputs are accurate. We decided to use a histogram comparison metric alongside our semantic categorical metric in order to see if our metrics were reporting correctly.

Given the previous work done on the topic of image similarity and retrieval, we plan to focus specifically on African wear since the existing models have not been trained specifically on the peculiar patterns, styles, and cuts associated with this type of clothing. Additionally, we will contribute to the discussion on the appropriate margin for

3 Image Dataset

Our image dataset consists of 15846 images of 5503 unique Afrowear products (i.e. on average, each product has three different images). There are 14 different labels that describe the gender and apparel category of a product (e.g. women's jumpsuits). We collected the images by scraping seven shopping sites [1] and one site [2] that contains a mix of stock and street images.

3.1 Data Preprocessing and Triplet Sampling

Given that the Triplet Network requires a triplet, $T = (A, P, N)$ where A, P, N are the anchor, positive and negative inputs, the number of possible triplets scales cubically with the number of training samples. To avoid directly searching the whole space and achieve faster convergence, we selected triplets that violate the triplet margin constraint:

$$||f_a - f_n||_2^2 > ||f_a - f_p||_2^2 + \alpha. \tag{1}$$

To this end, we created two different sets of triplets –semi-hard-negatives swatches triplets and hard-negatives category triplets. For semi-hard swatches triplets, we sampled two images of the same product and created negatives by picking a random image from the product folder as the model to generate the negatives. We selected 30 different swatches by scraping the web and used these swatches as overlaying hand-drawn masks for the patterned fabric on the model image with a Photoshop script. Thus, we were able to create 30 negative images for each anchor-positive pair. We then sampled $38,000$ triplets and found that they satisfied the semi-hard triplet constraint in [9] where their average margin was $\alpha = 0.05$, which was calculated using a pretrained ResNet50 network to extract the embeddings.

Figure 2: Sample anchor (left), swatch negative (center) and positive (right).

For category triplets, we generated triplets such that the anchor-positive pair belonged to the same class and the negative image belonged to a different class. We then sampled over $60,000$ triplets that strictly violated the triplet margin constraint in (1). We believed this selection would allow the model to learn both high-level differences such as type of Afrowear (from the category triplets) and fine-grained details such as pattern and swatch type (from the swatch triplets). This pattern of selection was motivated by the in-class and out-of-class triplet sampling used in [10].

[1] Ofuure, Ohema Ohene, Omi Woods, Wren Fashion, Zabba Designs, Grass Fields, Diyanu
[2] Afrikrea

Figure 3: Dataset samples showing the stock image (left) and street image (right)

4 Methods

For our Triplet Network, we used ResNet50's final FC layers to extract 2048-dimensional feature embeddings from high quality stock images and applied K-Nearest Neighbors to perform similarity ranking on these top-K images while evaluating with histogram correlation and categorical similarity metrics. Our architecture for deriving the embeddings is described below.

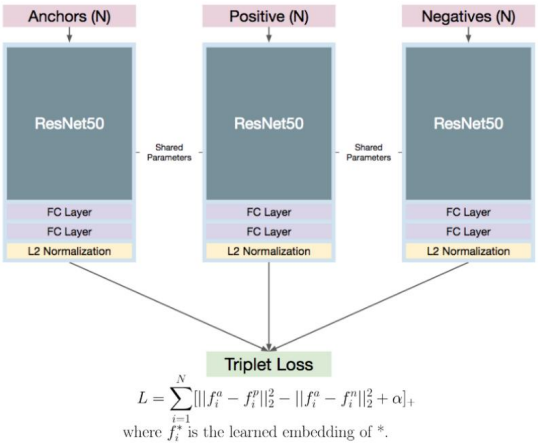

Figure 4: Triplet loss architecture trained on the same network with shared parameters..

4.1 ResNet50

We decided to use ResNet50 [5] for our triplet model for image matching. We evaluated this model with a K-Nearest Neighbors model and determined the accuracy of the output based on similarity labels and histogram correlations. We decided to use ResNet as our model of choice due to it outperforming many other models, such as VGG19, InceptionV3, Xception, and MobileNet, in terms of extracting non-null convolutional features [1].

For our K-nearest neighbors model, we used neighbors = 8 with the ball tree algorithm. We chose the ball-tree data structure for storing the embeddings as it is very efficient for high-dimensional vectors [11] although it takes up more time and memory to build initially.

We used the Minkowski distance as a measure of similarity such that given two images, I_1, I_2 the similarity score $D(I_1, I_2)$ is defined as $D(I_1, I_2) = \left(\sum_{i=1}^{n} |f_i(I_1) - f_i(I_2)|^p\right)^{1/p}$ where $f_i(I) \in \mathbf{R}$ is the ith element of the image I's embedding. The choice of the Minkowski distance serves the practical purpose of allowing us to easily tune our choice of p (whose current value is 2).

4.2 L2 Normalization

Given that the margin is a hyperparameter and we experimented with various sizes of the last fully-connected layer, we chose to normalize the final output $n-$dimensional vector to unit length (similar to [9]). This ensured that the maximum distance between any two embeddings, is at most 2 as shown in the lemma below. This helped reduce the search space of the problem.

Lemma 4.1. *Given two vectors,* $v_1, v_2 \in \mathbf{R}^n$ *such that* $||v_1||_2 = ||v_2||_2 = 1, ||v_1 - v_2||_2 \leq 2$.

This direct application of the Triangle Inequality can be proved using the Cauchy-Schwarz Inequality ($||v_1||_2 \cdot ||v_2||_2 \leq ||v_1||_2||v_2||_2$). An intuitive proof is that the length between any two points on a Euclidean hypersphere is less than or equal to its 'diameter', $d = 2$.

5 Experiments

5.1 Metrics

The primary metrics we used for evaluating our model relied on MAP:

$$MAP(y, \hat{y}) = \frac{1}{K} \sum_{i=1}^{K} \frac{\sum_{j=1}^{i} score(y_i, \hat{y}_i)}{i}$$

1. Histogram Correlation MAP(HCM): We used this metric to assess the stylistic similarity of our images. The HCM formulation [8] is shown below:

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}$$

Practically, we cropped the centers of each image being compared and calculated the HC based on these crops to reduce the influence of the background. Using MAP allows giving more weight to highly ranked products in the list of recommendations. The score function for MAP is given below:

$$HCS = 0[H_K + 0.5 < 0] + 1[H_K + 0.5 > 1] + 0.5[H_K + 0.5 < 1]$$

$$\text{Where } H_K = \frac{1}{N} \sum_J H_K(J) \text{ and } J = \text{number of bins}$$

2. Category Matching MAP (CMM): For our CMM, the following sets of categories are considered to be perfect matches and receive a score of 1.

- women-dresses,women-outerwear, women-matching-sets
- women-tops, women-jackets

The following sets of categories are considered to be half matches and receive a score of 0.5.

- "women-jumpsuits", "women-pants-and-shorts","women-matching-sets","women-dresses", "women-skirts"
- "women-tops", "women-jackets", "women-dresses", "women-matching-sets", "women-outerwear".

Hence, we used the following socring function to obtain Category Match MAP:

$$CMS = 1[qc = qr] + 0.5[qc \approx qr] + 0[qc \neq qr]$$

5.2 Hyperparameter Tuning

While tuning hyperparameters, we used the average margin, $m = \sum_{i=1}^{N} (||f_i^a - f_i^n||_2 - ||f_i^a - f_i^p||_2)$ to measure how well our network was able to separate positives from negatives. We found that higher batch sizes led to better convergence during training and, in general, most models converged within 60 epochs of training. We used RMSProp as the learning algorithm and found this to be . The hyperparameters we found to be most important in finding an optimal model were, in increasing order of importance.

1. Number of Fully-Connected Layers (FC): We noticed that having $3-4$ fully-connected layers after the ResNet50 network was sufficient to have good average margins. Higher layers did not produce considerably better results.

2. Learning Rate (LR): We did a logarithmic search over learning rates in the range $[10^{-4}, 10^{-1}]$ and found that the best values to be around 7×10^{-3}. We also used an LR decay of 0.2 on loss plateau but this did not have much of an impact.

3. Margin (M): We searched over margins in the range $[0.2, 6.0]$. We decided on using a margin of 2.0 as it yielded some of the best results.

5.3 Results and Evaluation

	HCM	CMM
Train	0.438	0.88
Test	0.33	0.823

Table 1: HCM and CMM results from our best model.

Figure 5: Train and validation losses and average margins across training

5.4 Discussion

Our model performed significantly better than the baseline on category matching (0.615 vs. 0.823) as it was able to learn that the background should be ignored. Even though we cropped as much of the background as possible, however, the distracting background colors still negatively affected our similarity metrics. Cropping to the center of images helped improve our test accuracy for color using histogram correlation between the query and output images. We also noticed that setting a higher margin generally led to better results in training. But from margins greater than 2.0, there was no significant impact on the margin after it converged.

5.4.1 Influence of Margin on Performance

Figure 6: Graph showing the average difference between anchor-positive and anchor-negative pairs based on the margin, α used during training.

As stated earlier, we observed that the margin was the most important factor in influencing performance. In general, we found that increasing the margin during training led to an improvement between anchor-positive and anchor-negative pairs. Surprisingly, for $\alpha \geq 1.8$, the average margin converged between 1.0 and 1.4. We

5.4.2 Learning Similarity

While the network was not able to discriminate between swatches with high precision, it performed well on able to discriminate based on categories, it had not quite learned. The fact that the loss metric converged indicates that we might either need to adjust the triplet loss or make the sampling of triplets harder.

6 Future Work

There are a couple of issues we would like to explore further in terms of this project:

1. Localization: We hope to utilize a Faster R-CNN to draw a bounding box around models in our images and feed the cropped localized part to our subsequent network to remove background pixels from affecting our accuracy metrics.

2. Metric Searching: We plan to categorize the data using attribute semantics in order to determine similarity based on other categories. This would involve manually labeling each image into several categories such as sleeve type, color, clothing style, etc. Motivated by [7], we are considering using Variational Autoencoders to learn image similarity.

3. Data Generation: We would like to increase our dataset by generating triplet examples from the rest of our dataset. However, triplet generation takes a long time due to negative images having to be manually Photoshopped. This manual aspect also introduces human error where the negative swatch doesn't contain the fold and wrinkles in the original model's outfit due to the swatch being an overlay. Finding a better way to generate negative examples without manually doing so would be ideal.

7 Contributions

1. **Robel Mengistu:** Data Generation, Triplet Sampling, Database and MAP Implementation
2. **Vikram Sreedhar:** Data Generation, Triplet Sampling, Similarity Metrics
3. **Oluwasanya Awe:** Triplet Network Implementation

References

[1] Convolutional features for keras' pretrained neural nets. 2017.

[2] Niyi Aderibigbe. Why the world should invest in african fashion. 2014.

[3] Dariusz Frejlichowski1, Piotr Czapiewski1, and Radoslaw Hofman. Finding similar clothes based on semantic description for the purpose of fashion recommender system.

[4] Nosmot Gbadamosi. The rising stars of a $31 billion industry. *CNN*, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.

[6] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. Where to buy it: Matching street clothing photos in online shops.

[7] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[8] Adrian Rosebrock. How-to: 3 ways to compare histograms using opencv and python. 2014.

[9] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[10] Devashish Shankar, Sujay Narumanchi, Ananya H A, Pramod Kompalli, and Krishnendu Chaudhury. Deep learning based large scale visual recommendation and search for e-commerce. *arxiv*, 2017.

[11] Jake VanderPlas. Benchmarking nearest neighbor searches in python. *Pythonic Perambulations*, 2013.

[12] Jiang Wang, Thomas Leung, Chuck Rosenberg, Jinbin Wang, James Philbin, Bo Chen, Ying Wu, et al. Learning fine-grained image similarity with deep ranking. *arXiv preprint arXiv:1404.4661*, 2014.

9 Appendix

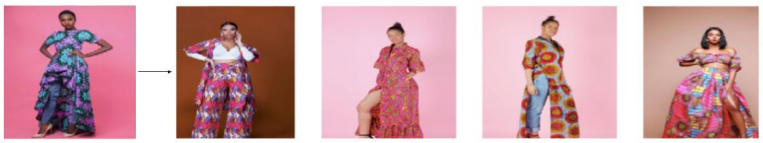

Figure 7: Baseline output.

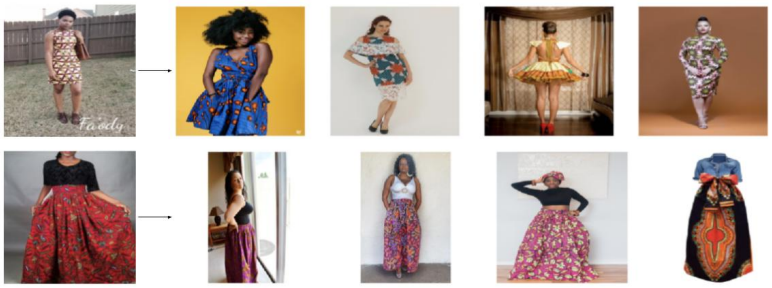

Figure 8: Sample outputs from our best models. It appears, in some cases that the model is able to learn certain pattern features but there is still work to be done in style similarity.

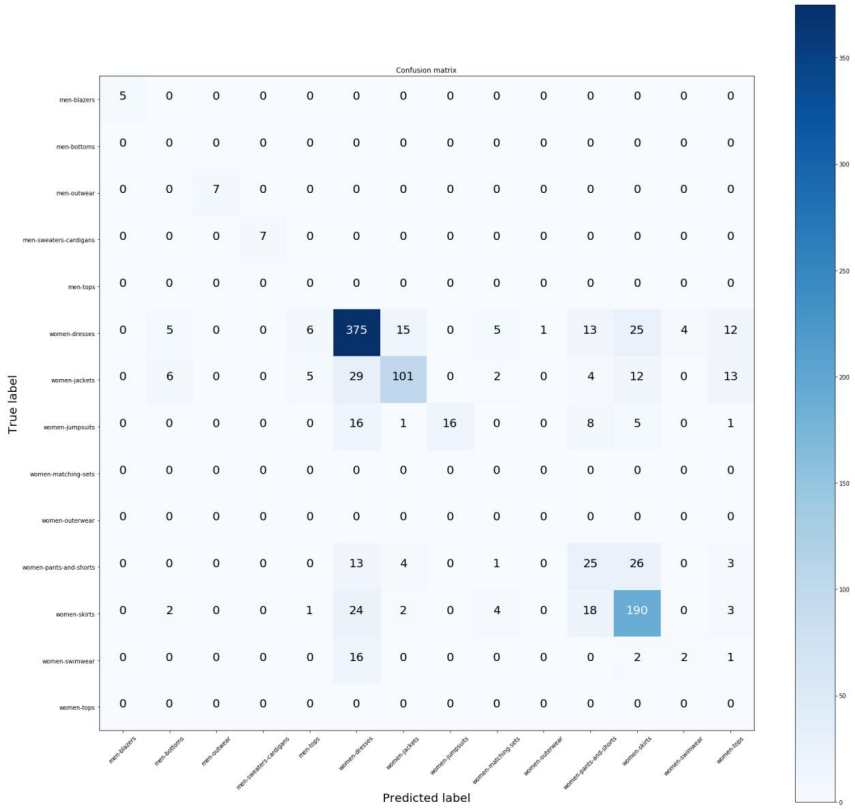

Figure 9: Confusion Matrix on Test Street Urls. We noticed that the model often confused women's jackets for