Clothing Recommendation

Raja Rajeshwari Premkumar (rpremkum)

Abstract
When we enter a store looking for an item, most of the times we have an idea of what it should look like. This look is inspired by the clothing worn by a celebrity or a model that we might have seen. We can use deep learning to clear the dilemma of whether the store has clothes similar to the ones we have in mind. The goal of our project is to facilitate this kind of functionality to consumers. Given an image of a person wearing a clothing, our model recommends the clothing that matches it or that which most closely resembles it.

Contents

Introduction

The E-Commerce explosion has begun, and so is the commercialization of AI techniques in improving customer experience. One of the biggest challenges faced by a store is to provide a more efficient browsing experience. It takes a lot of time for the customer to explore and select the items they were looking for. With our project we are able to provide an experience that can make this process faster and simpler. The idea is that the customer can upload an image of a person wearing a clothing that they desire to buy. We compare this image with all the clothing products in store in order to suggest best match or most relevant picks. The two important caveats to using our model are:

- It requires a lot of computational resources.
- It is limited to a subset of clothing products i.e., mostly upper garments and dresses only.

The advantages of using our model are:

- There can be any number of products. Apart from giving the closest product match, our model also gives products similar to the input.
- There is no need to retrain the model if new products are added into the catalogue. It easily includes the new products in its results.
- Better user experience since the user can now buy clothing products without rustling through the entire catalogue if they have an image of a person wearing it.

1. Background

We are using the LookBook dataset. This dataset has 77,546 images and 8,726 clothing products. For each product there are different images of a person wearing the clothing product. But the images belong to the same person wearing it, the picture is taken at different angles and with different poses and props (Refer Fig. 1)

1.1 Dataset

Below are steps and considerations we made for preparing our dataset.

- From Fig. 1 we can see that the images are of varied dimensions. This was a challenge for us. We took the average height of all images - 726 and average width of all images - 350 which threw memory error. To overcome that we had to try various dimensions like 300x150 which showed no improvement over our baseline 100x100 dimensions. Hence, we used the dimensions 100x100.
- We found out from our experiments that the model performs better if we have enough sample images of a person wearing the clothing in different poses and at different angles. Hence, we considered the products with at least 10 person images associated with it. This gives us 2,556 products with more than 10 associated images of person wearing a product.

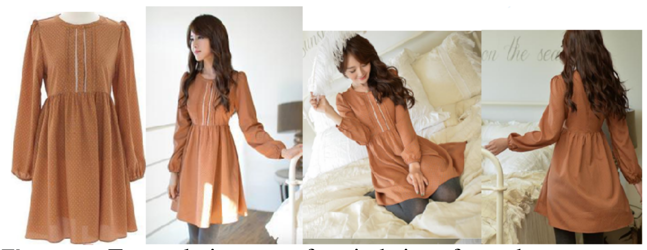
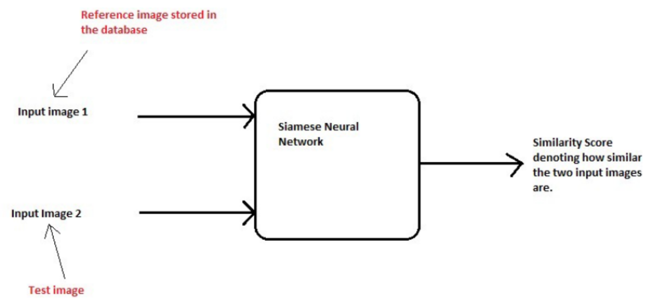


Figure 1. Example images of varied sizes from the LookBook dataset. A clothing image is associated with multiple images of a person wearing it.


Reference image stored in the database
Input image 1
Input Image 2
Test image
Siamese Neural Network
Similarity Score denoting how similar the two input images are.
Figure 2. Siamese Network[1]

1.2 Model

The traditional approach to this problem would be a standard classification model that takes the image of the person as input and outputs the softmax probability for all the products. The problem with this approach is that it needs lots of training and retraining in case of new products being added to the store.

Instead of classification, the **One-Shot Siamese Network**[1] takes the reference image of the person for a product and produces a similarity score indicating the chance the person is wearing that product. We wrap the similarity score in a Sigmoid function to obtain a standard output in range [0, 1]. Hence, the goal of Siamese Network[1] is to learn a similarity function that can find the similarity effectively. Interestingly enough, we do not need too many samples for each product and it works well with new test images. (Refer Fig. 2)

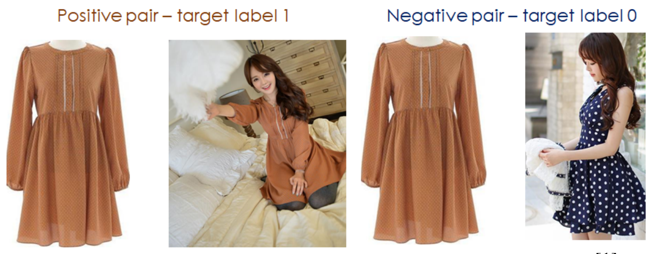
2. Methodology and model

The Siamese Network[1] requires input to be in pairs while the target label to be 0/1.

2.1 Data Preprocessing

The steps taken to prepare the data:

1. **Positive Pairs** are created. A Positive Pair is formed by taking an image of a clothing product and a person wearing that clothing product. The target label for such a pair is set to 1 indicating similarity.
2. **Negative Pairs** are created. A Negative Pair is formed by taking an image of a clothing product and a person wearing some other clothing product. The target label for such a pair is set to 0 indicating dissimilarity.(Refer Fig. 3)


Positive pair – target label 1
Negative pair – target label 0
Figure 3. Example Input pairs for Siamese Network[1]

3. To keep the train dataset balanced, we have considered equal number of positive and negative pairs for a given product. Since our input data set has at least 10 reference person images associated with a product image, we take 10 positive pairs and 10 negative pairs for a product.
4. We split the products to Train and Test. This gives us 2000 Train products i.e. 40,000 train samples while Test has 556 products.
5. Later, we vectorized the images using Pillow, resized images to 100x100, and shuffled the train data set.

2.2 Model Architecture

The intuition behind this model is to train a network over each of the two images in a pair. There are two parts in this network:

1. **Part One:** We obtain the embedding for an input image. It is a series of Convolution layers followed by linear layers.

   Below is the description of the network:

   (a) First Convolution 2D Layer:
   - 3 input and output layers
   - kernel size = 3
   - He initialization of the weights
   - Leaky ReLU
   - MaxPool Layer kernel size = 2

   (b) Second Convolution 2D Layer:
   - 3 input and 10 output layers
   - kernel size = 3
   - He initialization of the weights
   - Leaky ReLU
   - MaxPool Layer kernel size = 4

   (c) Batch Normalization

   (d) DropOut p = 0.5

   (e) Linear Layer with hidden size 2,000

(f) Tanh function

(g) Linear Layer that outputs a 1000 dimension embedding

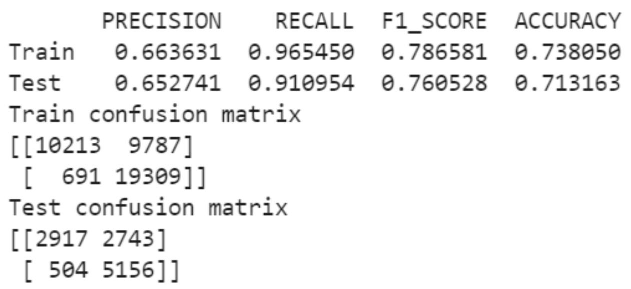
2. **Part Two:** The second part of the network does below;

- It accepts an input tuple which contains a pair as (product image, person image). This again can be a negative pair or a positive pair with target label set to 0 or 1 respectively.
- The pair (product image, person image) is passed through the network generating an embedding for both the images (as explained above).
- Now, it calculates the similarity score of the two embeddings using *Cosine Similarity*.
- Thus the output is now a score for each of the input tuple or a pair of product-person samples. This is passed through a Sigmoid layer that fits the scores between 0 and 1.
- Once we obtain the scores for the input data, we use *Binary Cross Entropy Loss* to calculate the loss of this score against the output labels which are 0 or 1.
- We obtain the final output prediction as 1 if the score is greater than or equal to 0.5 else 0 otherwise. This is used to evaluate the accuracy score.

2.3 Challenges faced

We faced several challenges before coming up with the final network.

- The accuracy initially started at 0.5 and got stuck with no progress with increase in number of epochs, we replaced the ReLU activation function with LeakyReLU. Although this showed some fluctuation between epochs, the accuracy unstable.
- We then added a dropout layer and reduced the learning rate which stabilized the output.
- Additionally, the Adam optimizer did not converge faster, hence we used the Stochastic Gradient Descent optimizer with 0.05 learning rate.
- The model takes about 5 hours to train for 15 epochs, hence a lot of time was required to tune the model and observe the pointers highlighted above.
- We also started with a Keras implementation which did not yield an accuracy above 0.5, since the network is not transparent enough we went on to try a PyTorch implementation.
- One of the bigger challenges was to run on Google Cloud. The problem was to read data from a zip file and to use CPU since more than one GPU was not available.



```
       PRECISION    RECALL  F1_SCORE  ACCURACY
Train   0.663631  0.965450  0.786581  0.738050
Test    0.652741  0.910954  0.760528  0.713163
Train confusion matrix
[[10213  9787]
 [  691 19309]]
Test confusion matrix
[[2917 2743]
 [ 504 5156]]
```

Figure 4. Evaluation results for Experiment1

3. Experiments and Results

As mentioned before the test data contains 556 products. The different ways in preparing the test sample data and the obtained results are discussed in this section. It is important to note one limitation of our dataset i.e., For a given product image there are images from only one person wearing the product albeit in different poses and angles. Yet, we cannot use these images to test if the model predicts the right product image during test phase; this will be considered as Data Leakage. Hence, we test the performance of the model on new products images and corresponding new person images wearing the product.

3.1 Experiment 1

The process followed to create the train dataset is followed here to prepare the test dataset i.e., For each of the 556 products, we take 10 positive pairs and 10 negative pairs. The target labels for the same is set to 1 and 0 respectively. If the output score is greater than or equal to 0.5 the output label is predicted to be 1 (product worn by the person is same as the product they are paired with) or 0 otherwise. Number of test samples = 556x20 = 11,320. We get 71.3 percent accuracy with an F1-Score of 0.76. (Refer Fig. 4), sample predictions (Refer Fig. 5)

3.1.1 Observation

From Fig. 4, we observe the precision is low when compared to recall. This shows that a lot of dissimilar images are being predicted as similar or in other words the False Positives are more. One probable explanation would be the confusion caused due to negative samples that appear to be as close as the positive samples.

3.2 Experiment 2

This experiment fully concurs to our problem statement. Given a person wearing a clothing product, we want to recommend the clothing that best matches the clothing worn by the person. As mentioned earlier, we recommend products in the test set only. Below is the process followed to obtain the results:

1. Form Negative and Positive Pairs and obtain the Sigmoid output from our Siamese[1] Network. For a particular person in a pair, we obtain the top recommendations.

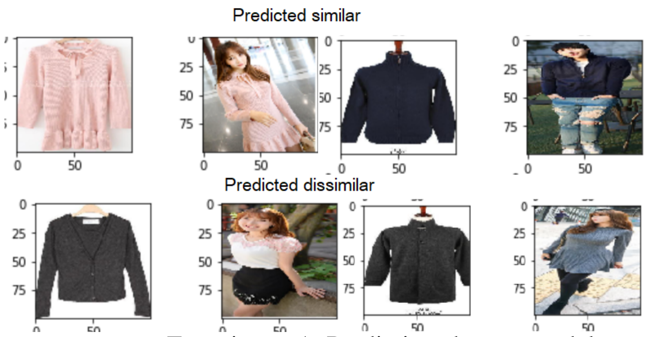


Figure 5. Experiment 1: Predictions by our model

No. of Products	Top 1	Top 5	Top 10	Top 20
50	3/50 = 0.06	18/50 = 0.36	28/50 = 0.56	43/50 = 0.86
300	1/300 = 0.003	13/300 = 0.043	32/300 = 0.1067	60/300 = 0.2

Figure 6. Accuracy: Number of Correct Predictions

2. We start with 50 products. Here, we take 50 person images corresponding to the 50 products. we make 2,500 pairs i.e., each person image has One Positive Pair and 49 Negative Pairs.

3. From the 50 pairs for a person image, we use maximum of the score from each of these pairs to obtain the Top N recommendations of product for that person. Refer Fig. 6 for results. Refer Fig. 7 for a snapshot of our Model Recommendations.

4. We conduct the same experiment (Steps 1 - 3) for 300 products i.e. 300x300 = 90,000 samples.

3.2.1 Observation

As highlighted in the observation from Experiment 1, the False Positives are high, this is not a favourable result for this experiment since we want to lessen the confusion for choosing the right product. Hence, we see good results for Top 5 Recommendations for 50 products with an accuracy of 0.36 whereas Top 10 products starts being significant for 300 products.

- The model needs to learn more patterns, it is not able to generalize over all products. Hence, more data may improve accuracy since we are using only 2,000 products to train currently. Example in fig8 we see that the letters on the clothing product are hidden by the person, hence it is hard for the model to find a good match. The right product is the last image in the series of predictions for the girl in the first image.

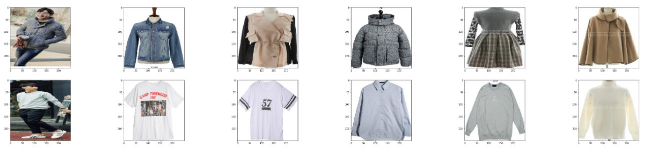


Figure 7. Example of 2 products: Top 5 Recommendations by our model

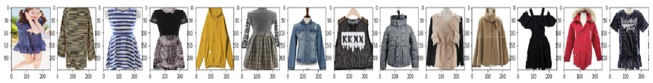


Figure 8. Example image with not so good recommendations

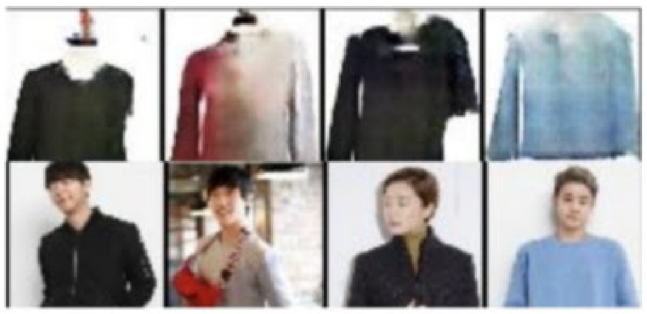


Figure 9. Results from PixelDTGAN[2] Model

- There are clothes that look different from when a person wears them, it is hard for the model to recommend such clothes as well.

Hence, we can say that if the person image clearly speaks about the product or if the image can be generalized, then the model does a good job.

3.3 Experiment 3

During our exploration on various models, we found out about PixelDTGAN[2]. It is a Deep Convolutional Generative Adversarial Network which encorporates Pixel-Level Domain Transfer. Using the same LookBook dataset, they were able to achieve good results with a qualitative and quantitative analysis of 82%. When looked closely into this model, we found out that while the model performs well, it fails to accommodate new products. We would have to retrain the GAN model on the new catalog which takes up a lot of computational resources as opposed to our Siamese Network[1] which doesn't require retraining when new products are encountered. Apart from that, PixelDTGAN[2] takes huge amounts of time i.e., around 1000 hours on a basic CPU to obtain good results. While the results obtained were significantly well, it fails to fulfill our purpose for this project. A Snapshot of the results from the PixelDTGAN[2] is shown in Fig. 9

4. Summary and Conclusions

The Siamese Network[1] does a good job in determining the similarity between a product and a person image pair. But the precision is somewhat low. To overcome this, We need more data that has variation too in order to train our model. This also needs more computational resources. This is especially required to improve our results for Experiment 2 which demands less False Positives. The PixelDTGAN[2] on the other hand cannot accommodate new classes and takes even more time to train than our current model. Hence, it is not recommended.

Acknowledgments

A special thanks to Professor Minje Kim for the help and guidance throughout the semester and for the project. Also, thanks to the AIs of this course who have helped us throughout the semester. Thanks a lot to Donggeun Yoo et al. for creating the LookBook dataset.

References

[1] Siamese Neural Networks for One-shot Image Recognition
[2] Pixel-Level Domain Transfer
[3] Signature Verification using a ”Siamese” Time Delay Neural Network
[4] Learning a Similarity Metric Discriminatively, with Application to Face Verification