Clothing Products Recommendation: Convolutional Autoencoders and VAE for image compression

Zoey (Li-Yen) Yang

1 Introduction

My project aims to address pertinent challenges within the fashion industry. Specifically, within online clothing retail platforms, a prevalent strategy for increasing sales involves providing customers with tailored clothing recommendations and thereby enhancing the likelihood of a purchase. A common approach is to suggest products that are similar with customers' past purchases or views. Consequently, it becomes important to devise a method to quantify product similarities. However, images are high dimensional pixel data and it can therefore be difficult to quantify their similarities.

This project is inspired by a Kaggle competition: `https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations`, which is provided by a clothing company, H&M. The original competition was designed to predict customers' future purchases based on their purchase history, and the dataset was composed of clothing or fashion images along with the purchase record of the customers. My project focuses on the methodology of finding similar products for each given product. To do so, I first explored a convolutional autoencoder to reduce product images into a latent space of lower dimensions that can capture the distinctive features of each product's image. I then tried training a variational autoencoder (VAE) based on the same structure for comparison. Finally, different similarity distance metrics were explored between latent representations to find similar products based on K-nearest neighbors.

2 Related Work

Related work in the field of image embedding techniques using machine learning methods has achieved significant advancement, reflecting a growing interest in leveraging machine learning for efficient image representation. Various studies have explored the application of autoencoders for compressing data to its representative latent space. Among different architectures of autoencoders, convolutional neural networks have been shown to achieve superior performance in terms of extracting image information given their ability to extract local features in the images. Yang et al. 2002 [1] proposed a method to compress images based on varations of PCA. Zhang et al. 2018 [2] used a convolutional autoencoder to sucessfully denoise and compress images of human faces. Pintelas et al. 2021 [3] shows an example of a convolutional autoencoder designed to extract image information to be used for downstream classification tasks. The studies showed a significant improvement on the classification task using the compressed image from the autoencoder compared to using raw images directly. Finally, Nellas et al. 2021 [4] proposed a method using convolutional VAE to compress images for the purpose of clustering, which shows the potential of VAE for image dimension reduction.

3 Approach

3.1 Dataset Collection, Overview, and Preprocessing

My dataset was collected from the kaggle competition: H&M Personalized Fashion Recommendation. The original dataset is composed of images of fashion products including clothes, shoes, hair ties, and purses from a clothing company H&M. A smaller subset of 5000 images from the original dataset was curated for the training and evaluation purpose of the project. After creating a subset of the dataset, I then did some preprocessing steps

for the images. The images in the dataset do not have a uniform pixel size, but the majority of them have a pixel size of 3x1166x1750 (RGB x width x height). So I rescaled all the images to 3x64x43 to reduce their size while maintaining a similar proportion of width x height. Then, I used zero padding to transform them into square images with a pixel size of 3x64x64. Finally, these RGB images are converted from an integer representation to a floating point representation by scaling the pixel values to the range of 0 to 1.

3.2 Problem and Solution Formulation for Feature Extraction of the Images

Common techniques to extract image representations includes PCA and autoencoders. However, PCA is limited to linear transformations when reducing the dimensions. Given a sample x_i where $i \in [m]$, denote E(x) as the encoder, D(x) as the decoder, and V as the eigen vector matrix of the data correlation matrix, then, as proven in class, PCA returns $E^* = V^T$ and $D^* = V$. However, images are high-dimensional data that may be very complicated. They have 3 dimensions, where the first dimension represents the RGB colors and the second and third dimensions represent visual locations within the image. Given the fact that the same object with a slight translation or rotation can result in a matrix that is significantly different, it may be hard for PCA to capture and learn that they are the same object. Therefore, I chose to use a convolutional autoencoder to address the problem with non-linear functions in the autoencoder and to help maintain translation invariance by the convolutional neural network.

The problem of parameterizing the autoencoder is forumlated as follows: Let m be the number of samples (images) in the training set, where each sample has RGB float values and a pixel size of 64x64 and can therefore be denoted as $x \in \mathbb{R}^{3\times64\times64}$. Let E be the encoder and let D be the decoder of an autoencoder. The ERM problem of the autoencoder can then be stated as:

$$(E^*, D^*) = argmin_{E,D} \sum_{i=1}^{m} \|x_i - D(E(x_i))\|^2 \tag{1}$$

E and D are parameterized by neural networks and can be written as the following formula: $E(x) = f_k^E o ... f_2^E o f_1^E(x)$, where $f_i^E = a(w_i x + b_i)$ is the i^{th} layer of the encoder neural network applied with a non-linear activation function $a(y)$. Similarly, $D(x) = f_k^D o ... f_2^D o f_1^D(x)$, where $f_i^D = a(w_i x + b_i)$ is the i^{th} layer of the decoder neural network applied with a non-linear activation function $a(y)$.

Autoencoders, however, might also have their limitations. In an autoencoder, the latent representation of the data is mapped by a deterministic function, which may lead to overfitting. Second, the latent space encoded by the encoder is not regulated, and theoretically, the latent space may be uninformative due to the lack of regularization in the latent space. Thus, a variational autoencoder is also tested in this project to see it can perform better than the autoencoder. Denote $P(x)$ as the PDF of the data and $P(z)$ as the PDF of the latent space. In VAEs, we can approximate $P(z|x)$ with a function $q(z)$, which is assumed to be have a Gaussian distribution $N(\mu_x, \sigma_x)$. The problem now then becomes finding μ_x^*,σ_x^* that best approximates $P(z|x)$ [5].

$$\begin{aligned}
(\mu_x^*, \sigma_x^*) &= argmin_{\mu_x,\sigma_x}\ KL((q(z), p(z|x)) \\
&= argmin_{\mu_x,\sigma_x}\ (\mathbb{E}(log\ q(z) - \mathbb{E}(log\ \frac{p(x|z)p(z)}{p(x)})) \\
&= argmax_{\mu_x,\sigma_x}\ (\mathbb{E}(log\ p(x|z)) - KL(q(z), p(z)) \\
&= argmax_{\mu_x,\sigma_x}\ (\mathbb{E}(-\frac{\|x - D(z)\|^2}{2c}) - KL(q(z), p(z)))
\end{aligned} \tag{2}$$

Thus, the ERM problem of the variational autoencoder can be stated as:

$$(E^*, D^*) = argmin_{E,D} \sum_{i=1}^{m} \|x_i - D(E(x_i))\|^2 + \beta KL[N(\mu_x, \sigma_x)||N(0, I)] \tag{3}$$

where KL denotes the KL divergence function and μ_x and σ_x denote the mean and standard deviation of the latent distribution, respectively. β is a term to control the weight of the contribution on the optimization trade-off between the reconstructed error and the regularization of the latent space.

3.3 Model Architecture for Feature Extraction

A few different autoencoders architectures were explored. Table 3.3 below shows the architecture used in the final model that has the best performance. (2D Conv denotes a 2D Convolutional layer, 2D ConvTrans denotes a 2D Transpose Convolutional layer, ReLU denotes a ReLU Activation layer, and Sigmoid denotes a Sigmoid Activation layer.)

Layers	Kernel Size	Stride	Padding
2D Conv	3x3x16	(2,2)	(1,1)
ReLU	-	-	-
2D Conv	3x3x8	(2,2)	(1,1)
Sigmoid	-	-	-
2D ConvTrans	3x3x16	(2,2)	(1,1)
ReLU	-	-	-
2D ConvTrans	3x3x3	(2,2)	(1,1)
Sigmoid	-	-	-

Table 3.3: Layers and specific parameters used in the final model

VAE models were constructed with the same architecture mentioned above, except that the latent vector is sampled using μ and σ parameterized by the encoder.

3.4 Model Training for Feature Extraction

The Adam optimizer, a stochastic gradient descent method that is based on adaptive estimation of moment, is used to optimize the model parameters. For autoencoders, the loss for a sample is defined as the reconstruction loss $\|x_i - D(E(x_i))\|^2$. For VAEs, the loss for a sample is defined by the addition of the reconstruction loss and KL divergence loss between the latent distribution and an isotropic unit Gaussian Distribution defined as: $\|x_i - D(E(x_i))\|^2 + KL[N(\mu_x, \sigma_x)||N(0, I)]$

Reconstruction loss is used for the autoencoders. Reconstruction loss + KL divergence loss is used for the VAEs. All models are trained with 30 epochs. Different hyperparameters were explored and evaluated for model performance. After the training procedure, the model with the best performance was used to extract image latent representations.

3.5 Similarity Metrics used for Nearest Neighbor Search

After obtaining the image representations, the next problem is to measure the similarity distance between different image representations. Ideally, if we define a good way to measure the distance between the image representations, different products with similar properties will be closer with each other in the distance space defined by the similarity distance metrics. Three different similarity metrics were explored and evaluated in this project. Given a pair of image representation vectors u and v, the distance metrics are defined as:

1. Cityblock Distance: $\|u - v\|_1$
2. Euclidean Distance: $\|u - v\|_2$
3. Cosine Distance: 1 - $\frac{u \cdot v}{\|u\|_2 \|v\|_2}$

Using the similarity distances, a nearest neighbor search is then performed to find the k most similar latent representations. Given a product of interest, the recommendation system will recommend the k most similar products that correspond to the k nearest latent representations of the products.

4 Results

4.1 Model Performance of autoencoders and VAEs

Using the model architecture mentioned in the method section, the autoencoder performs much better than the VAE. During the training of the VAE, the model was able to learn, but the reconstruction error remained

signigicantly higher than that of the autoencoder model. Thus, the autoencoder architecture was chosen and its hyperparameters were tuned. The table below shows the reconstruction loss of the autoencoder model trained with different hyperparameters:

Learning Rate	Batch Size	Training Error	Test Error
0.1	256	0.0056	0.0054
0.01	256	0.0031	0.0030
0.001	256	0.0058	0.0056
0.01	16	0.0005	0.0005
0.01	32	0.0009	0.0009
0.01	64	0.0015	0.0014
0.01	128	0.0017	0.0016

The results show that the most appropriate learning rate for the model is 0.01, and that in general, the smaller batch size gives better performance. Thus, the final model was trained with a learning rate of 0.01 and a batch size of 16. Figure 4.1A shows the reconstruction loss of the model training process. From the plot, it can be observed that the ADAM optimizer was able to reduce the loss of both training and testing samples. The loss calculated from the training and testing set was roughly the same, indicating that the model is able to generalize well.

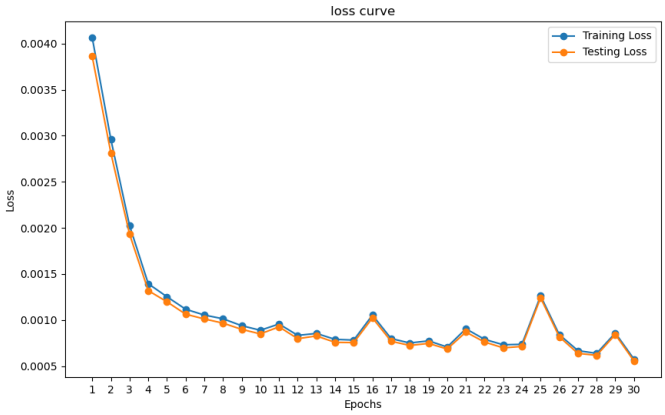

Figure 4.1 A: loss curve during the training process of the model trained with a learning rate of 0.01 and a batch size of 16

To visually examine the ability of the model to reconstruct images, three images from the test set were randomly selected for evaluation. Figure 4.1B shows the comparison of images that were fed into the encoder and the corresponding images that are reconstructed by the decoder:

Figure 4.1 B: Three different original product images are shown on the left. The corresponding reconstructed images from the autoencoder are shown on the right.

The above comparison shows that the autoencoder does have the ability to reconstruct the images. The clothing products remain the same shape after the reconstruction, although the color of the images are slightly deviated from the original colors and the reconstructed images are relatively blurry compared to the original images.

4.2 Recommendation Results based on Nearest Neighbor Search

Figure 4.2A-C shows a product of interest and the recommended products based on nearest neighbors with three different similarity metrics. The results show that the recommended products from three similarity metrics have similar patterns. For the first example product, cosine distance was able to capture the closest clothing style while Euclidean Distance and CityBlock Distance capture the product with the same color. For the second product (figure 4.2 B), Euclidean distance and CityBlock distance recommended very similar products, whereas cosine distance has relatively different recommendations. However, the color scheme and style between the recommended product and the product of interest is similar. For the third product (figure 4.2 C), the recommendation system picked up on the products that contain two pieces, but the pattern and the colors of the recommended products are not quite similar to the product of interest.

Figure 4.2 A: A product of interest (Product #1) is shown on the left. Recommended products are shown on the right, with the first recommended product shown as rank1 and so on.

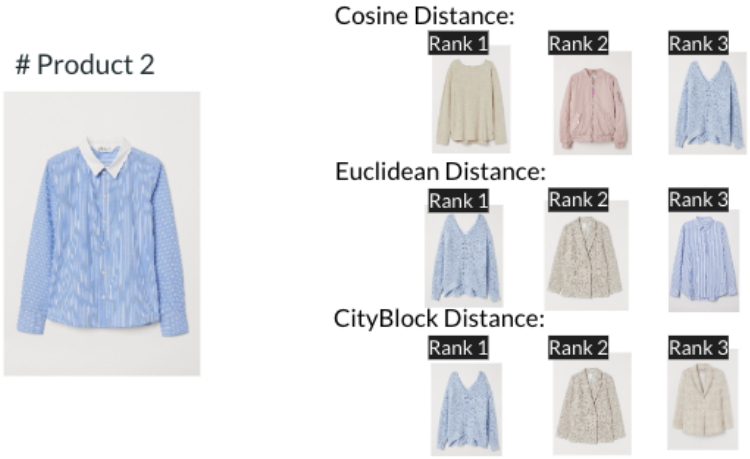

Figure 4.2 B: A product of interest (Product #2) is shown on the left. Recommended products are shown on the right, with the first recommended product shown as rank1 and so on.

Figure 4.2 C: A product of interest (Product #3) is shown on the left. Recommended products are shown on the right, with the first recommended product shown as rank1 and so on.

5 Conclusion

In conclusion, this project implemented and demonstrated a method to quantify image product similarities in the fashion industry. By leveraging convolutional autoencoders, the methodology effectively reduced high-dimensional image data into a lower-dimensional latent space, enabling the exploration of different similarity distance metrics. The recommendation system, based on a nearest neighbor search using three distinct similarity metrics, showcased its ability to suggest products with similar styles or features. However, it also has its limitation: the current recommendation system is not able to adjust its recommendations based on user preference on what kind of similarity it should pick up on. In the examples shown in the results, it may pick up on patterns, colors, or number of pieces of the clothings.

In terms of why the VAE was unable to reconstruct the image data well, one possibility is the improper β parameter used in the training process. As shown in equation (2), there exists a tradeoff between training VAE to reduce the reconstruction error and regularizing the latent space distribution. It is possible that β is set to be too large for the data. Another possibility is that VAE assumes Gaussian distributions for the $p(z|x)$ and $p(x|z)$. However, the distribution of the dataset might not be suitable for this assumption.

To sum up, autoencoder performs the best in this particular dataset. Future work could involve further examinations of the ability of the autoencoder. Since the dataset is unlabeled, the analysis of how well the image representation works is limited and subjective. If we can label the dataset into a few different classes, we can use clustering methods such as k-means clustering to cluster the latent image representations and analyze if products in the same class will be clustered together. Also, a more objective scoring measurement function of the recommendations can be extremely helpful in quantifying the quality of the recommendation system. Another interesting future work involves analyzing how and why the VAE failed to reconstruct the data in details. Finally, the other interesting question would be, is autoencoder the best way to reduce the image dimensionality for this task? Given an unlabeled image dataset, will the performance be better if the dimensionality is reduced by extracting the hidden layers of a CNN model that is trained to label similar images from a labeled dataset that is much bigger?

6 Code Availability

Code for this project is available at: `https://github.gatech.edu/lyang417/ml-recommendation-system`

References

[1] Yang, Jian, and Jing-yu Yang. "From image vector to matrix: A straightforward image projection technique—IMPCA vs. PCA." pattern Recognition 35.9 (2002): 1997-1999.

[2] Zhang, Yifei. "A better autoencoder for image: Convolutional autoencoder." ICONIP17-DCEC. (accessed on 23 March 2017). 2018.

[3] Pintelas, Emmanuel, Ioannis E. Livieris, and Panagiotis E. Pintelas. "A convolutional autoencoder topology for classification in high-dimensional noisy image datasets." Sensors 21.22 (2021): 7731.

[4] Nellas, Ioannis A., Sotiris K. Tasoulis, and Vassilis P. Plagianakos. "Convolutional variational autoencoders for image clustering." 2021 International Conference on Data Mining Workshops (ICDMW). IEEE, 2021.

[5] https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73