Ian Whitehouse

Dr. Bei Xiao

Deep Learning in Vision

May 7th, 2024

Outfit Recommender Project

Introduction

For my project, I worked on an outfit recommendation system which could match images of clothing to suggested outfits. This was motivated by the question "what goes with this shirt?," which is, in essence, the same thing as recommending an outfit. I structured my solution to this problem using ideas from image retrieval or image search problems. Similar to facial recognition, I used machine learning models to generate embeddings, which allowed me to search for similar images. Then, because every image I used was an item-completed outfit pair, I simply returned the images of the complete outfits that used the items of clothing that were the closest match to the input item of clothing.

To get my dataset, I tried scraping websites using Selenium, however, I discovered that two of the websites I looked at, J-Crew and American Eagle, used the same CDN, Adobe Scene7. This CDN has a very simple API, and all I had to do was send HTTPS requests to their Scene7 server with an item's id, an item's color's id, and a one- or two-character code to specify which image I wanted. From there, it was rudimentary to download a list of the items they sold and request those from the API. However, while both companies had the images of only the item of clothing tagged with a consistent code, the images of the clothing with a full outfit were inconsistently tagged with a couple different codes. Initially, I planned to use a fully connected network to map between the embeddings of the individual items of clothing to the images of the full outfits, but this turned out to be really difficult, and the inconsistency with the images of the full outfits made it nearly impossible. Therefore, one of the elements of this project I spent the most work on was cleaning the data and developing heuristics to determine which images of a model wearing the outfit were the best and most consistent for my machine learning model.

Model

I used 3 different models to generate embeddings. The first one, I-JEPA, I have wanted to use on a project for a while because its creator, Yann LeCun, has argued that it represents a new paradigm in self-supervised learning [1]. Similarly to a masked autoencoder, this model learns to generalize by recreating masked areas of an image, however, I-JEPA's novel premise is that it recreates the masked areas in latent space

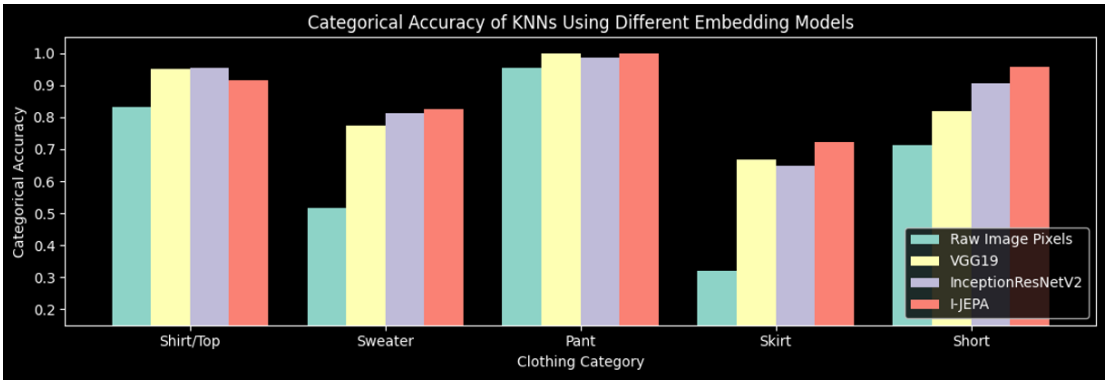
instead of in the final image space. LeCun has argued that this is revolutionary because the model can learn to ignore unnecessary detail, such as the background, while in latent space, allowing it to develop a more semantic understanding of its task [2]. Integrating I-JEPA turned out to be one of the hardest parts of this project, however. Despite presenting results where they had inferenced I-JEPA to generate encodings, the version of the model that LeCun's team put on GitHub lacked any ability to inference. After searching for solutions to this problem, I had to fork the repository and write my own inference pipeline before I could use its embeddings. In the end, I was successful, but the lack of information on how they actually tested the model meant that I had to make assumptions about their process. This model was pretrained on ImageNet and I did not fine-tune it to my dataset.

I also used InceptionResNetV2, which was a SOTA CNN-based model when it was released in 2017 [3]. This model is a 164-layer model that is based on the inception family of models while also implementing ideas from residual networks. I used Keras to download this model and removed the final stage of fully connected layers, taking embeddings directly from the convolutional output [4]. This model was also pretrained on ImageNet.

Finally, I used VGG-19, an older convolutional network that won the ImageNet challenge in 2014 [5]. This model was 19 layers deep and I also removed the fully connected networks before I used it to generate embeddings. This model was also trained on ImageNet and I included it because it was significantly smaller and cheaper than the other two models, so I believed that it could make an interesting baseline.

Within the embeddings space, I used KNNs from the Scikit-Learn library to search for similar images [6]. I also applied KNNs directly to the image pixels as a baseline.

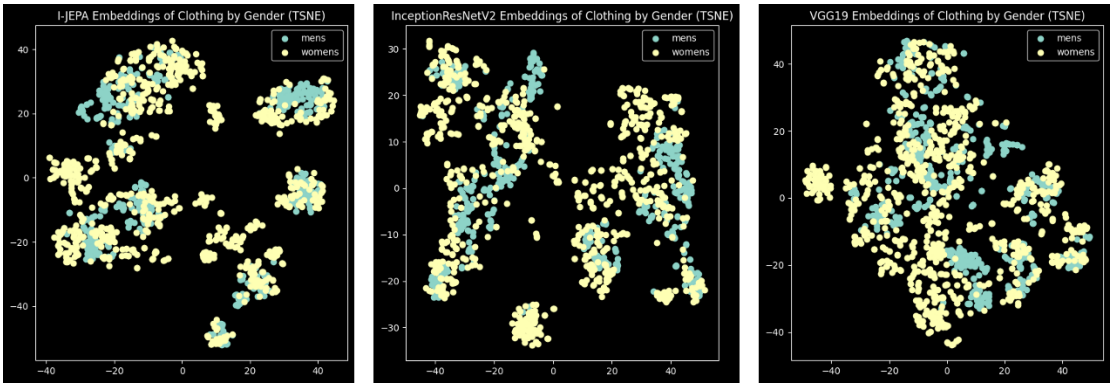
Results



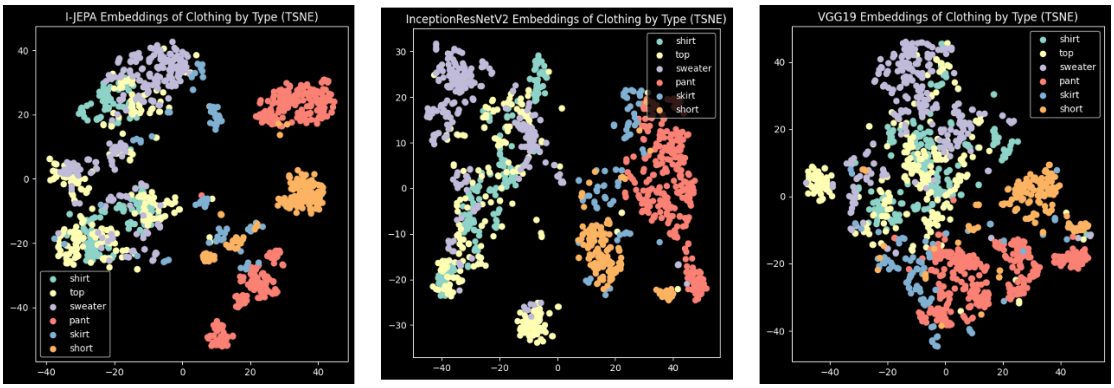
Quantitative results generated by using each model's embeddings to predict a novel image's category

Since outfit quality or clothing similarity are subjective, I used the clothing item category to quantitatively measure the accuracy of each embedding. For three of the five categories of clothing, I-JEPA performed significantly better than the other models. VGG-19

and InceptionResNetV2 performed equally, with a virtual tie on one category, and each model beating the other at two different categories. The baseline, a KNN trained directly on image pixels did the worst, but its accuracy was similar to the other three models on three of the five categories. All of the models seemed to perform well with an abundance of data, and they all performed the worst on the skirt class, which had significantly fewer instances.



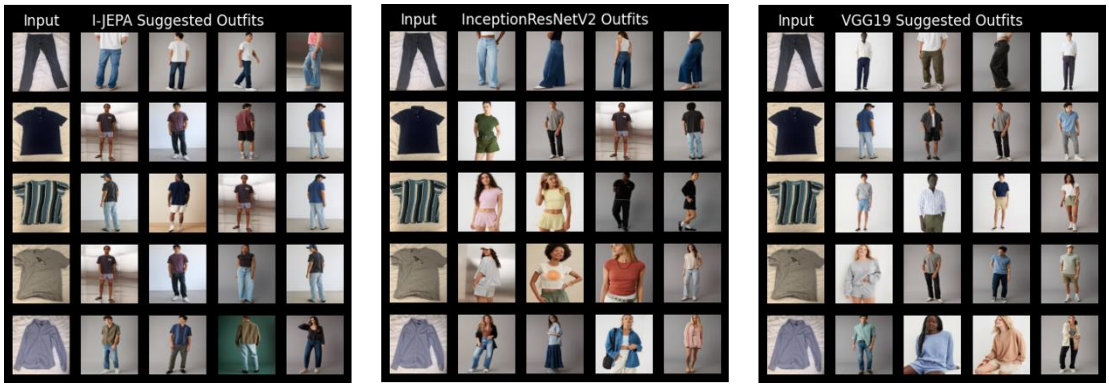
TSNE plots of each model's embeddings, colored by gender



TSNE plots of each model's embeddings, colored by clothing type

Looking at TSNE plots of the embeddings, the I-JEPA model appears to have done the best, developing clear clusters for pants, skirts, and shorts. However, the results are inconsistent, as the InceptionResNetV2 model was the only model that developed separate clusters for some sweaters and some tops, while the other models grouped them together. This is somewhat an issue with the dataset, as J-Crew clearly differentiated shirts from tops while American Eagle referred to women's shirts as tops (I combine them into one category for the quantitative results). Additionally, I-JEPA did the best as separating the different clusters, while InceptionResNetV2's clusters were close together and sometimes had unclear boundaries and VGG-19's clusters practically overlapped.

Items of clothing picked as similar to test images based on each model's embeddings



Outfits suggested from test images based on each model's embeddings

Finally, I included the outfits that were generated with each of the embedding models. Here, the results are especially interesting because none of the models performed well. On first glance, InceptionResNetV2 appeared to have performed the best as it was able to match the clothes I own to similar clothes, especially for my button-down shirt, which the other models struggled with. However, when looking at the outfit images that matched these shirts, all of them were on female models. In contrast, I-JEPA's embeddings had trouble finding similar looking items of clothing, but it somehow knew that the shirt was a men's shirt, and mainly returned male models. This is especially interesting when you consider that there were no gender-defined clusters in any of the embeddings, which was a feature I thought I would see and was surprised that the models hadn't picked up on.

Applications

While I applied this to outfit creation, there are evidently numerous applications for image search within image embeddings. I am particularly interested in triplet learning, which is a self-supervised process that encourages the model to learn the difference between pairs of images. While this learning turned out not to be appropriate for this task,

it is commonly used in facial recognition, and I would like to apply it to a project in the future.

Conclusions

Using these models, I learned the importance of embedding models for computer vision applications. While, in this case, I simply searched through the embeddings, I am interested in exploring them more. One idea I had was using the embeddings as the seeds of a generative model, allowing me to use any clothing item and generate an outfit based on it. Ultimately, to do this, the dataset would have to be significantly higher quality, as right now, the model would have to cope with different poses and camera angles. I also would be interested to see how the results would improve if the model was finetuned using images from the dataset.

However, I was disappointed in the quality of the outfits that I was able to generate. One of the items I uploaded was a blue polo shirt from Target. While this image was not in the training dataset, I know for a fact that there are multiple blue polo shirts in the dataset, making me disappointed that the model didn't find any of them. Additionally, the models struggled to match the t-shirt with the duck graphic or the striped shirts to similarly styled shirts. Looking at the matched items, only InceptionResNetV2 actually pulled out t-shirts with similar graphics to the duck shirt, despite the American Eagle website hosting hundreds of graphic t-shirts.

Note About the Dataset

This dataset was 1/10th the size of the original dataset. To download the original dataset, run *ae_download.py*, *jcrew_download.py*, and *remove_bg_and_crop.py*.

References

[1] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun and N. Ballas, "Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture," arXiv, 2023.

[2] Y. LeCun, Interviewee, Yann LeCun, Chief AI Scientist at Meta AI: From Machine Learning to Autonomous Intelligence. [Interview]. 25 May 2023.

[3] C. Szegedy, S. Loffe, V. Vanhoucke and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," arXiv, 2016.

[4] F. Chollet, "Keras," 2015.

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, 2014.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learing Research, vol. 12, no. 85, p. 2825–2830, 2011.