Content Based Image Retrieval System

Ankit Singh-IEC2018076, Shakti Majhwar-IEC2018006, Achal Singhal-IEC2018023 ,Rishab -IEC2018049
BTech.ECE 7th semester
Indian Institute of Information Technology Allahabad
Mentor:Ramesh Kumar Bhukya
Assistant Professor
Department of Electronics and Communication Engineering
rkbukhya@iiita.ac.in
Prayagraj,India

***Abstract*—In this paper, we implemented a real-time image recommendation system using ResNet and ANNOY. The aim of this project is to increase the efficiency of the traditional recommendation by decreasing the prediction time in a real-time environment drastically and also make a diverse output that can adapt to the changes int the interest of the users**

***Index Terms*—Transfer learning,Real-time recommendation,Image embeddings, DeepFashion, ResNet, ANNOY, Clustering, KNN**

Introduction

We want to build a recommender system that can automatically generate the set of items based on the recent history of the user. Our challenge was to make the system work well for the large dataset in real-time and also adjust to new products.Speed is essential in an e-commerce business.Previous methods work using the knn algorithm which is quite slow thus We'll be using transfer learning, approximate closest neighbors, and the built-in centroid detection in PyTorch to create our recommender.We break down four steps needed to construct the model: Dataset Distribution ,Preprocessing using the normalise and image embeddings , Model we used is the ResNet18 from Transfer Learning and FastAi hooks to retrieve them and at last for search we used ANNOY with the help of Centroid Detection.It help in find the nearest neighbors without pairwise comparison with each dataset thus it is most efficient fit to our model .

When customers search the products on the ecommerce website, they aren't able to get too many similar or equivalent products. So, the customers can miss the many nice clothes and it causes the loss of company also.

A. Transfer Learning

Transfer learning is a machine learning technique. In deep learning, transfer learning is the most used method.We start with pre-trained models in computer vision. Also, given the large computation and time resources, natural language processing problems. However, we must create neural network models.
Because transfer learning is linked to a variety of issues. Multi-tasking and notions wander, for example. It is not, however, solely a field of research for deep learning.
The basic principle of transfer learning is straightforward: take a model that has been trained on a large dataset and apply it to a smaller dataset. We freeze the early convolutional layers of the network for object recognition and just train the last few levels that produce a prediction. The concept is that the convolutional layers extract broad, low-level properties that apply across images — such as edges, patterns, and gradients — whereas the later layers recognise individual aspects within an image, such as eyes or wheels.

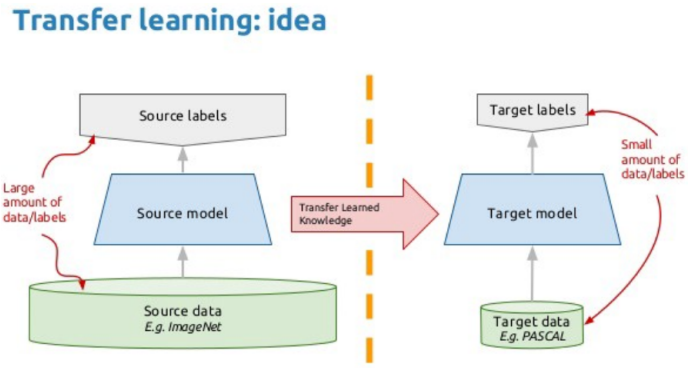


Fig.1: Transfer Learning idea

B. Embeddings

A mapping of a discrete, categorical, variable to a vector of continuous integers is called an embedding. Embeddings are low-dimensional, learnt continuous vector representations of discrete variables in neural networks. Because they can lower the dimensionality of categorical data and meaningfully represent categories in the transformed space, neural network embeddings are helpful.
The objective of neural network embeddings is threefold:

- In the embedding space, find the closest neighbours. These can be used to give suggestions depending on the user's preferences or cluster groupings.

- For a supervised task, providing input to a machine learning model.
- Concepts and relationships between categories can be visualised with this tool.

C. Clustering

Clustering, often known as cluster analysis, is a machine learning technique that divides an unlabeled dataset into groups. It is a method of grouping data points into multiple clusters, each of which contains similar data points. The objects with possible similarities are kept in a group with few or no connections to another.

It accomplishes this by identifying similar patterns in the unlabelled information, such as shape, size, colour, and activity, and classifying them according to the inclusion or exclusion of those patterns. It is an unsupervised learning method, which means the algorithm receives no supervision and works with an unlabeled dataset.

Following the application of this clustering technique, each cluster or group is given a cluster-ID, which the ML system can utilise to facilitate the processing of huge and complicated datasets.

For example, We'll use it in this example to automatically locate groupings of golfers based on their characteristics, and then interpret these clusters to provide customised instruction.

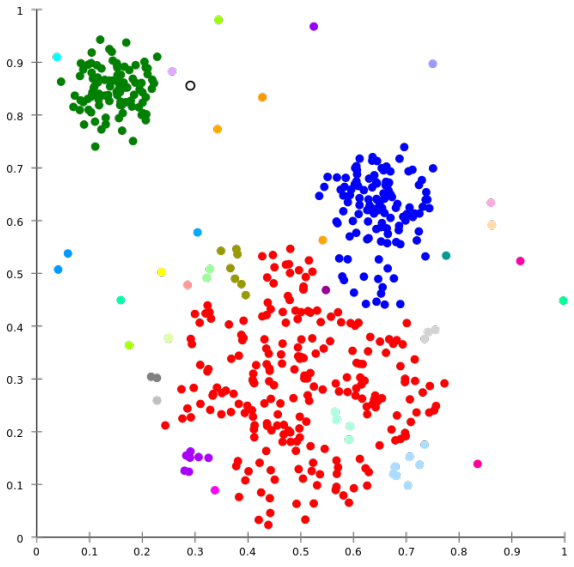


Fig.2: KNN

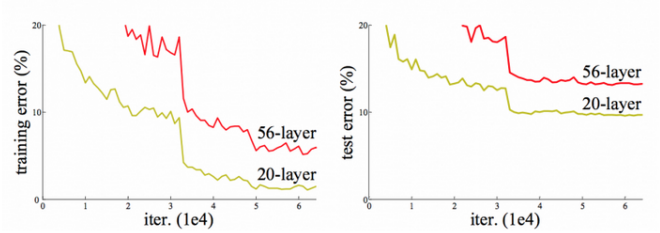
I. Dependencies

A. Image embeddings using Transfer learning

We can use Embeddings which is vector representation of images .It also project similar image vectors closer to each other.

For Image Embeddings vectors we need the second last layer of the CNN.CNN is convolutional Neural Network is a neural network used to classify images . Our dataset has 46 categories with 280k+ samples thus it is better to use pre-built and pre-trained CNN models applying transfer learning as most of knowledge gained is transferable and only last few layers need to trained.As the name states Transfer Learning take a model trained on a large dataset and transfer its knowledge to a smaller dataset .This way we can save lot of computation time .

Now we have to decide which ResNet model can be the best fit to our model .It may be better to use a higher layer model such as resnet50,resnet152 but colab provides limited computational and thus resnet18 is the best fit .
Below figure represent the resent model's training and testing error



RESNET Training error (left) and test error (right)

B. FastAi

Fastai is a deep learning library that provides students with high-level components that can quickly and easily deliver cutting-edge results in standard deep learning domains, while providing researchers with Low-level components can be mixed and linked to create new approaches.It simplifies training fast and accurate neural nets using modern best practices.It aims to do both things without significant compromises in terms of ease of use, flexibility, or performance.

We have used fastai hooks to retrieve image embeddings as this makes the process of retrieval easier and faster.We have used hooks to retrieve intermediate model values in forward or backward passes. I used Pytorch hooks to generate feature vectors for all images of trains and real data sets.

C. Clustering: KNN ANNOY

The k-nearest neighbor (kNN) algorithm is one of the most common classification methods. It is a Supervised learning algorithm.It needs to calculate the distance of the unmarked objects to all the marked objects in the training set. The majority rule is used to determine the class name taking into account the weight of the distance. This gives heavier weight to closer neighbors, depending on the distance.

Because it memorises training samples rather than putting forth the effort to understand the data, the k closest neighbour method is a lazy learner.A typical machine learning (ML) algorithm that isn't lazy learns from the training data goes through the process of discovering important characteristics and their relative relevance levels. The learning process usually follows a pattern that is comparable to curve fitting.

The kNN approach, on the other hand, simply stores the training data points in a Kd-tree or other multidimensional indexing structures, then classifies the query data based on the k closest data points to the query point, thus the name.

- k-nearest neighbors requires no training.
- k-nearest neighbors can learn non-linear boundaries as well.
- k-nearest neighbors predicts just the labels. Sometimes, having probabilities allows you to integrate the model seamlessly with other probabilistic machinery

The algorithm's learning is:

- Instance-Based Learning: Here we are not learning weights from the training data to predict the output but using full training instances to predict the output of invisible data.
- Lazy learning: The model is not previously learned using training data and the learning process is postponed until the prediction is requested in the new instance.
- Non-parametric: in ANN there is no predefined form of the mapping function

1) Pseudocode for K Nearest Neighbor (classification)

This is the pseudocode for building the KNN algorithm from the ground up:

- Load the training data into the programme.
- Scaling, missing value treatment, and dimensionality reduction are all options for preparing data.
- Determine the best value for K.
- Predict a new data's class value: Calculate distance(X, Xi) from i=1,2,3,....,n, where X represents a fresh data point, Xi represents training data, and distance is calculated using your preferred distance metric. Sort the distances with the relevant train data in ascending order. Select the top 'K' rows from this sorted list. Find the most common class among these 'K' rows. This is the class you are expected to be in.

2) Summary: Annoy's algorithm

Time for preprocessing:

- Create a slew of binary trees. Split all points recursively by random hyperplanes for each tree.

Time to make a query:

- Each tree's root should be added to the priority queue.
- Search all the trees using the priority queue until we have search kcandidates.
- Duplicate candidates should be removed.
- Calculate the distances between candidates.
- Candidates should be sorted by their distance from each other.
- Return the ones at the top.

Approximate nearest neighbor implementation open-sourced by Spotify, to index and search feature vectors efficiently.Approximate nearest neighbor algorithms can be seen as a trade-off between accuracy and performance.

ANNOY tactics require constructing a Cluster Index first which essentially makes a decision with some X leaders with which we can locate similarities. Then, the cluster index is used to locate Approximate K-nearest neighbours for all given points.The cause why that is called 'approximate' is that the cluster constructing manner isn't always perfect.It is an approximate cluster index.Annoy is one of the maximum broadly used libraries for doing approximate nearest neighbours.

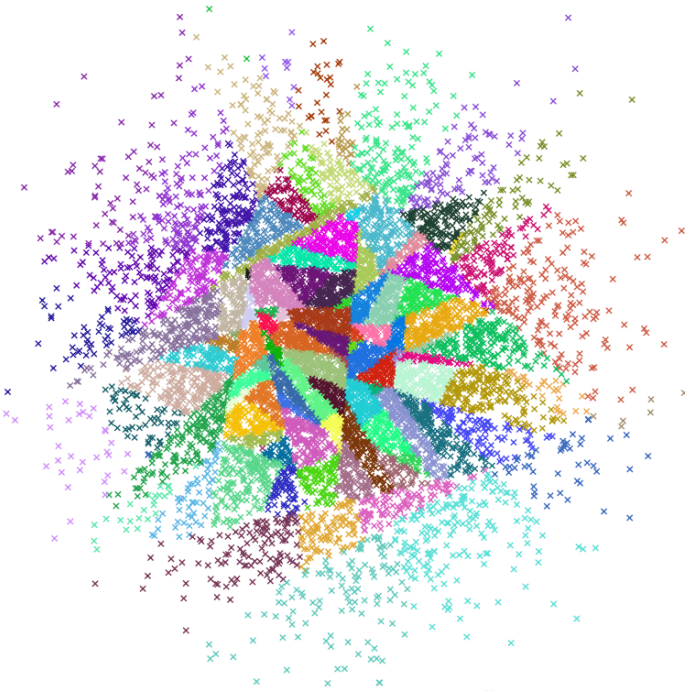


Fig.3: ANNOY

At each intermediate node of the tree, a random hyperplane is chosen, dividing the space into two subspaces. This hyperplane is selected by sampling two points of the subset and taking the hyperplane equidistant from them.

We do this k times to get a forest. k should be tailored to your needs, considering the trade-off you have between accuracy and performance.

Good! We end up with a spatially partitioned binary tree. The good news is that points that are close together in space are more likely to be closer together than in a tree. In other words, if two points are close together in space, it is unlikely that the hyperplane will separate them.

To find any point in this space, we can traverse the binary tree from the root. Each intermediate node (small squares in the tree above) defines a hyperplane, so we can determine which side of the hyperplane we need to continue on, and that determines whether we go down the node left or right child. Finding a point can be done in logarithmic time since that is the height of the tree.

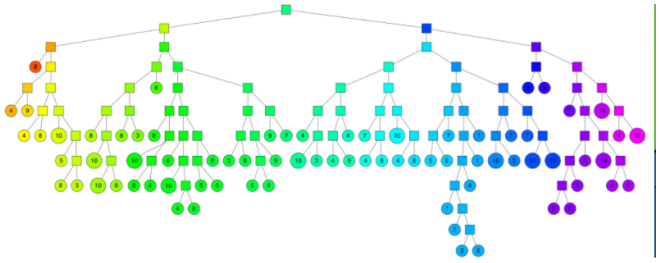
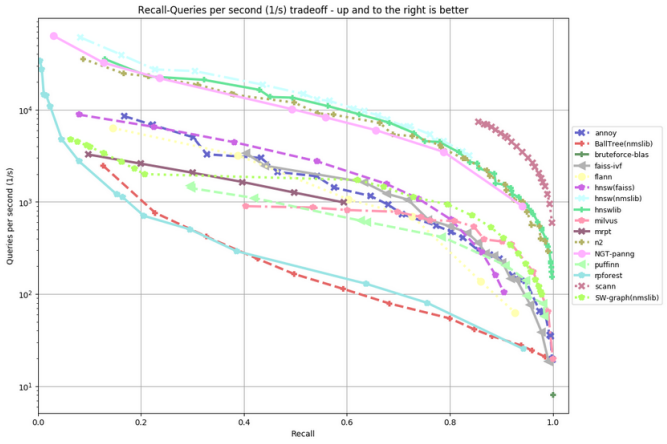


Fig.4: NTrees

II. Benchmarks Score of Annoy

- Result for annoy are pretty good as compare to various Panns,Nearpy, LSHForest .
  x-axis represents the Recalls
  y-axis represents Queries per seconds



1) KNN vs ANNOY

There is an issue we're facing with the traditional K-NN technique. We don't want to calculate all pairs, because storing distances for that many pairs and calculating genuine K-nearest neighbours at runtime for any object is quite challenging.

Approximate Nearest Neighbors comes into play in this situation. These argue that since the goal of the challenge is usually always to find the K closest neighbours,why bother calculating similarities between any two entities that have no chance of appearing in the neighbourhood?

III. Dataset Description

The dataset used for this project is DeepFashion open-sourced dataset by Liu, Ziwei and Luo, Ping and Qiu, Shi and Wang, Xiaogang and Tang, Xiaoou. It is a subset of a large dataset produced by the same authors. This dataset consists of over 280K images with a total of 46 categories.

The dataset consists of 1 image folder consisting of 5,621 sub-folders and 3 CSV files, consisting of information like:

- Image IDs
- Image paths in the folder
- 46 image categories mapping
- Image split into approximate 200K Train, 40K Test, and 40K Validation

IV. Model Description

A. Environment

The model is trained and tested on the basic Google Colab platform with python 3.6+ version.

B. Initial Dependencies Setup

- First, the important libraries to support the project are installed using the pip command.
- Then other dependencies are imported into the environment.
- Later the use of GPU is enabled to make the process fast.

C. Initial Dataset Setup

- Initially, the dataset is downloaded into the runtime environment from the Gdrive using gdown library.
- The dataset consists of one zip file of a folder consisting of images, which is to be extracted using zipfile library.
- The dataset also consists of 3 CSV files consisting of meta-data.
- All the CSV files are merged by only considering the relevant information for the model using the pandas and numpy library.
- The merged pandas Dataframe consists of 3 attributes: image_path, dataset_type, and category, along with the indexes as image IDs.

D. Image Pre-processing

- In the pre-processing part, first, the images are split into 3 parts: train, test, and validation as 200K, 40K, and 40K images each, with the help of information provided by DeepFashion.
- Later all the images are fixed to 224x224 size, which is the default size specified by the ResNet models for best results.
- Images are also normalized before dividing them into batches.
- The batch size is taken to be 128, as it gives the best accuracy possible for the chosen model and dataset.
- Now the dataset is ready to be fed to the ResNet models, to get trained.

E. Model Training

- First, the model is trained on the pre-trained model to retrieve the best learning rate for the model, using the fastai library.
- The optimized learning rate came out to be 0.01(1e-2), by observing the loss vs learning rate curve.

Fig.5: Loss vs. Learning Rate

- Now the model is trained for 8 epochs and the obtained top 1 accuracy came out to be 0.664400 and the top 5 accuracy came out to be 0.926100. The total time for training is approximate 6 hours.
- Now the model is saved on Gdrive in order to be used for later use.

F. Retrieving the image embeddings

- First, the model is downloaded from Gdrive and retrieved using fastai hooks.
- Now the image embeddings present in the 2nd last layer of the model are retrieved, and these feature vectors(i.e., image embeddings) are concatenated to the pandas dataframe.
- Now the model is ready to be fed to ANNOY(Approximate Nearest Neighbors Oh Yeah!), in order to get the recommendations.

G. Setting up ANNOY

- First, the image embeddings centroids are calculated using the mean.
- Now the ANNOY tree is populated, by considering 100 ntrees(the more the ntrees the better the approximation), and metric_choice as “angular”.
- Now the model is ready to give similar image recommendations using ANNOY.

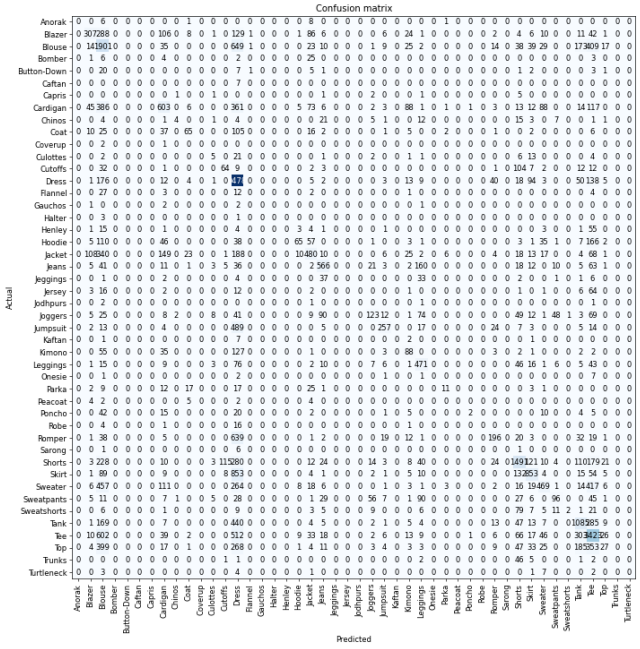
V. Train/Test Accuracy Table

- We have run the model for 7 epochs and check the accuracy for k=1 and k=5

epoch	train_loss	valid_loss	accuracy	top_k_accuracy	top_k_accuracy	time
0	1.471898	1.353760	0.602250	0.602250	0.897625	16:54
1	1.451260	1.388633	0.593200	0.593200	0.891875	16:19
2	1.440240	1.380244	0.594750	0.594750	0.894000	16:32
3	1.409721	1.315045	0.610425	0.610425	0.905450	16:39
4	1.335867	1.257601	0.634525	0.634525	0.911550	16:36
5	1.315253	1.238566	0.639000	0.639000	0.915575	16:48
6	1.275752	1.166933	0.658325	0.658325	0.923950	17:06
7	1.228926	1.135368	0.664400	0.664400	0.926100	17:09

VI. Confusion Matrix

- Here is the confusion matrix of the dataset
  x-axis represents the predicted value
  y-axis represents the actual value



Conclusion

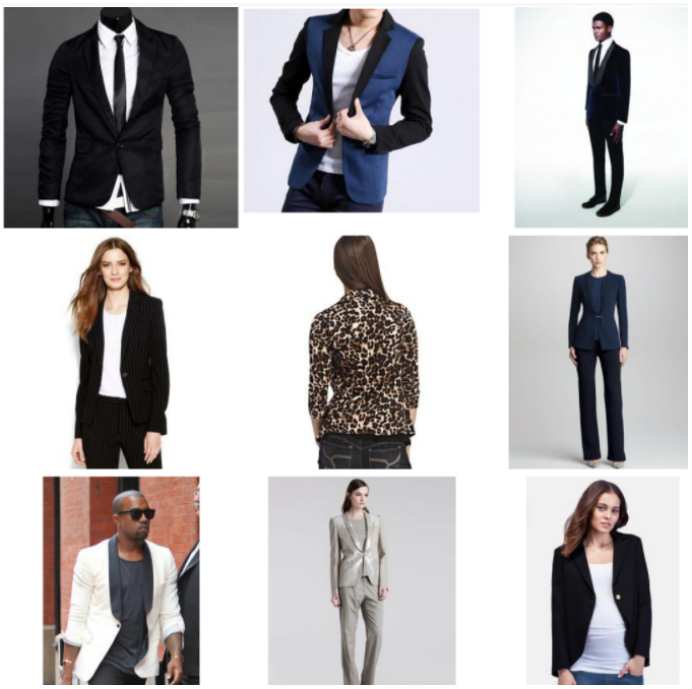
In this project, we presented the detailed analysis and implementation of Real-time personalized recommendation using ANNOY. We presented, that how extracting embeddings can drastically increase the accuracy and decrease the training and prediction time. Also using ANNOY made the prediction time as quick as a wink, i.e., in micro to milli seconds range. We further compared the traditional Clustering Algorithms with ANNOY and showed that whey ANNOY outperforms them. In the era of everything working on ML/AI, the users don't like to wait for anything, that is what we achieved in this project.

References

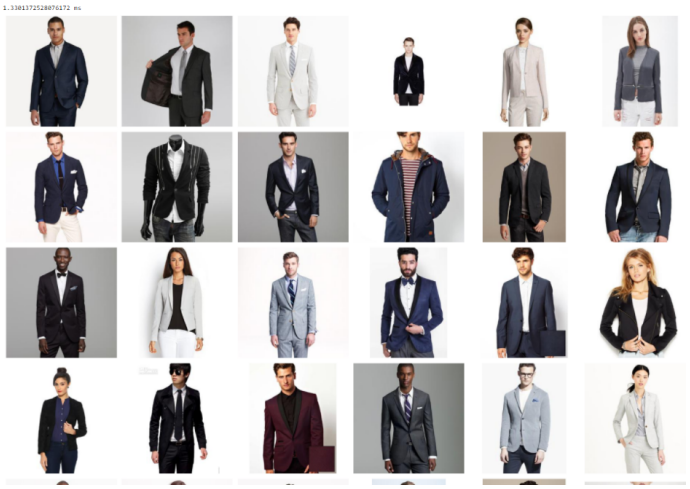
1. Liu, Ziwei and Luo, Ping and Qiu, Shi and Wang, Xiaogang and Tang, Xiaoou, "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.
2. N. N. Qomariyah and A. N. Fajar, "Recommender System for e-Learning based on Personal Learning Style," 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2019, pp. 563-567, doi: 10.1109/ISRITI48646.2019.9034568.
3. Y. Park, "Recommending Personalized Search Terms for Assisting Exploratory Website Search," 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2019, pp. 404-405, doi: 10.1109/JCDL.2019.00091.
4. A. Bodaghi and E. Homayounvala, "Personalization of interactive recommender systems for expert users," 2018 4th International Conference on Web Research (ICWR), 2018, pp. 58-62, doi: 10.1109/ICWR.2018.8387238.
5. Erik Bernhardsson, "Nearest neighbors and vector models – part 2 – algorithms and data structures", 01-10-2015[Online], Available: https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html [Accessed on: 15-09-2021]
6. Zeyuan Allen-Zhu, Yuanzhi Li, "What Can ResNet Learn Efficiently, Going Beyond Kernels?", 1 June 2021
7. Abhigya Sodani, Michael Levy, Anirudh Koul, Meher Anand Kasam, Siddha Ganju, "Scalable Reverse Image Search Engine for NASAWorldview", 10 Aug 2021
8. Trieu H. Trinh, Minh-Thang Luong, Quoc V. Le, "Selfie: Self-supervised Pretraining for Image Embedding", 27 Jul 2019
9. Will Koehrsen, "Transfer Learning with Convolutional Neural Networks in PyTorch"[Online], 27-11-2018, Available: https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce [Accessed on: 18-10-2021]
10. Rinu Gour, "Transfer Learning for Deep Learning with CNN", [Online], 28-11-2018, Available: https://medium.com/@rinu.gour123/transfer-learning-for-deep-learning-with-cnn-afa1fe0aa7e2 [Accessed on: 25-10-2021]

Results

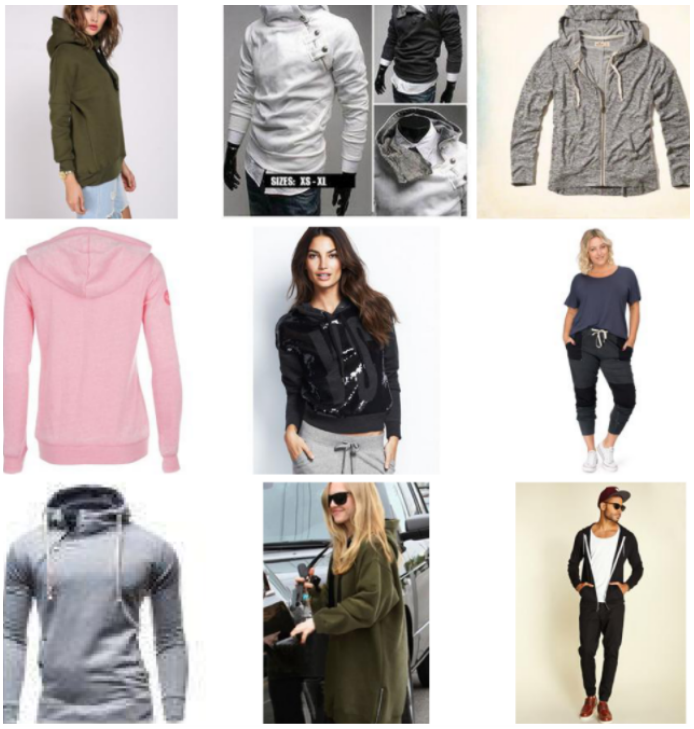
Here are few results of the final output of our model.



Input: blazer



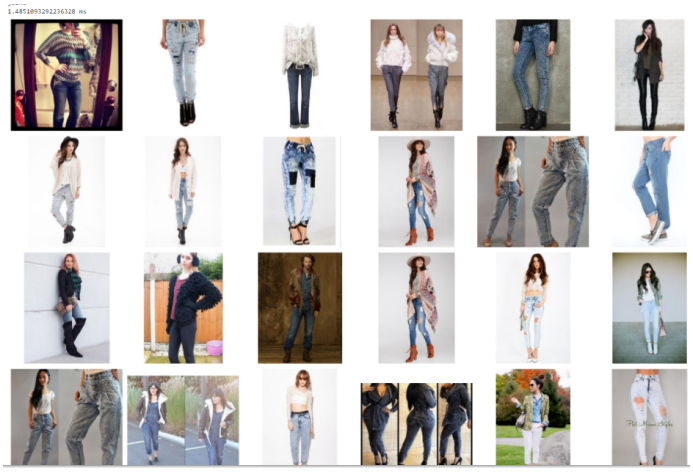
Output:blazer



Input: hoodie

Input: jeans

Output:hoodie



Output:jeans