Visual Semantic Search - Shoppin

Image based semantic search like Google Lens takes an image as input and retrieves most similar images based on its content.
The basic idea behind semantic image search is to represent each image as an embedding of features extracted by a pre-trained or task specific fine-tuned deep metric learning model. Then image retrieval can be performed by storing & comparing image embeddings via vector based similarity like cosine distance based indexing.

Dataset

Polyvore Outfits
This dataset contains 21,889 outfits from polyvore.com. The following metadata is available for each of the product in the outfits-
Title

- Description
- Category, Semantic Category
- Image

Approaches

I have proposed and implemented several approaches for visual semantic search-

1. **Supervised Learning - ResNet-50**
   A ResNet 50 model is used as a feature extractor. Last global average pooling layer is used to extract the vector which has 2048 dimensionality.
   This model is pre-trained on ImageNet data of 1000 classes for image classification task. So its not expected to do really well on the e commerce related images as such categories are not the major part of the ImageNet data.
2. **Supervised Learning - ResNet-50 fine-tuned**
   In this approach, to tackle the target data misrepresentation issue mentioned above, I fine-tune the last 75 layers of ResNet so that it learns the semantic meaning of the ecommerce related product images such as style, color, fabric pattern etc.
   Here the category label from the metadata is used as ground truth.
3. **Self supervised learning - CLIP**
   CLIP model is proven to be more expressive but the major reason I wanted to use this for our problem is due its self supervised nature meaning that it does not need image and category labels which is hard to get at web scale but it uses image, title pair to learn the representation.

Image and title pair is available on the web at huge scale and get to expand further from the internal dataset of any ecommerce company.

4. **Self supervised learning - CLIP fine-tuned**
   The model by openAI is trained on 400 million image-text pairs but sources of data is not revealed hence I fine-tuned clip on our own Polyvore data so that it can generalize to this as well.
5. **Object detection + Embedding**
   I've not implemented this approach end-to-end but I have given the major object localization part where we get the largest object present in the image.
   It can be integrated with all the 4 approaches above and guaranteed to improve the performance but at a cost of increased compute cost.
   But ideally this approach gives an idea about how we as a user can crop the relevant area(instead of an ML model) in the query image(like in Google Lens) after uploading and we can extract the embedding of that portion only.

Result of a SSD model trained on COCO dataset

Semantic Search Tech

Vector DB - Milvus

Why we need a vector DB - We can simply use Faiss similarity search index without a vector

DB but it will give a lot of false positives in real world images due to misscategorization and lack of the annotations issues.

Faiss vs Milvus-

Hence having a vector DB instead of just a similarity search index would allow us to store metadata along with the vector and use that metadata to improve the similarity search.
One most useful capability for us is - before doing similarity search for a given query image vector, limit the search space with all the products having same/similar categories only hence reducing false positives similar items.

Similarity Index

We choose IVF_FLAT as index and L2 as distance metric for Milvus

Evaluation

To evaluate the system against ground truth, we consider images in the same class as relevant items. You can follow steps below or design a benchmark by your own to compare search results with all relevant items in the dataset.

Evaluation metric - mAP

This benchmark measures the system with the metric of mAP, which is mean average precision over all query images.
For each query image, we expect to get images of the same class from the dataset.

For compute efficiency evaluation we also calculate **QPS** metrics(query processed every second)

Benchmarking Results

- ResNet pre-trained
  mAp: 0.42
  qps: 29.79
- ResNet fine-tuned
  mAP: 0.46
  qps: 29.5
- CLIP
  mAP: 0.49
  qps: 37.54
- CLIP fine-tuned
  mAP: 0.6
  qps: 17.017

Qualitative results

The fist image in the row is the query image.

CLIP fine-tuned

CLIP pre-trained

ResNet fine-tuned

ResNet pre-trained

Code

Code repository - https://github.com/ranjeetthakur/visual_search/tree/main

1- **visual_similarity_shoppin.ipynb**
The main notebook for end-to-end visual search - dataset creation, feature extraction, vector DB setup, inference, evaluation
2- **resnet_polyvore_finetuning.ipynb**
This notebook fine-tunes ResNet on the Polyvore dataset. The fine-tuned model is again used in the main notebook mentioned above.
3- **object_detection.ipynb**
This notebook has the main object location pipeline implemented using Tensorflow Hub's SSD detection model for lightweight and fast inference. I compared it with FastRCNN as well but that is very slow compared to SSD given its 2 stage detection architecture.
4- **clip_fine_tuning_polyvore.ipynb**

This notebook fine-tunes CLIP model in Polyvore image, title pairs to improve the performance of visual search.
I exhausted all free credits in Google Collab, Hence I was not able to do the full tuning hence I have compared the performance of a similarly fine-tuned version on Farfetch outfits dataset that is similar to Polyvore.

Production Choices

Low latency vs High accuracy tradeoff
For a better user experience, I would go ahead with a smaller model which can do inference quite fast while still being decently accurate.
Further accuracy can be easily improved with metadata filters, adding data augmentation while training.
Also we can train these lighter architectures in a self supervised way needing less labeled data.

Embedding dimensionality reduction
ResNet embeddings are 2048 dimensional, For inference perspective I would reduce the dimensionality to 512(may be, will evaluate multiple dimensions).

Metadata filtering
We can easily build metadata enrichment models using iFashion, MMFashion, Polyvore, Farfetch public datasets and use that info to improve the accuracy while doing vector similarity search in Milvus or Qdrant using filter feature.

Object Detection choices
An object detection pipeline is essential as we would have cases where images contain full body images hence multi ecommerce products are visible causing search accuracy to go down due to mismatch.
I would propose a batch offline pipeline which reads all the hot products available in inventory to be inference with detection of major objects in them using light weight architectures like Yolo and SSD to avoid compute intensive models like FastRCNN.
An ondemand API can be exposed in case non-popular items are requested for detecting the objects in them.