Image Based Product Search

Bhupesh Hada
Data Science Graduate Student
State University of New York at Buffalo
bhupeshh@buffalo.edu - bhupeshh

***Abstract*—This document represents a comprehensive effort to build the best possible solution that could associate the user-provided images of the products with the seller-used images. This solution can aid the customer product search process and at the same time help the e-commerce websites to enhance their customer experience and thereby bring in more website traffic and hence more revenue.**

I. Project Overview

- The primary objective of the algorithm is to accurately identify matching products from the collection of seller photos when presented with a search query. In the project, I categorize product images into two groups: user photos and seller photos. User photos usually depict products captured in cluttered settings using a smartphone camera. These images significantly contrast with seller photos, which aim to showcase products for online marketplaces. Also, user photos contain object-bounding boxes that highlight the desired products within user photos. Therefore, I utilize these images alongside their respective bounding boxes as search queries.
- There are a few state-of-the-art solutions present in the image product search domain. One of the most popular solutions is the use of CNN architectures like VGG and EfficientNet to perform feature extraction and learn the similarity between the images. Such solutions use pre-trained networks on large-scale datasets like ImageNet. Another solution makes use of SIFT operator to identify the descriptors and key points in the provided images and find the overlap across the images to get the similarity between them.
- In the solution, I started by developing a baseline solution that used resnet18 CNN architecture as the underlying model to find the association between the seller and the user image. This association is measured using the mean average precision (mAP) score. After getting the baseline accuracy I used sequential iterations of advanced models - arcface and XBM. Since the model run is sequential I used the weights obtained from the previous iteration as an input to the current iteration. Each iteration differs with respect to the resolution of the images and the size of the training data.

II. Approach of the Solution

- The solution used resnet18 and convnext_xxlarge CNN architecture. The convnex_xxlarge architecture is a transformer-inspired CNN architecture that has shown significant improvement in terms of model performance. Convnext_xxlarge uses Patchify stem - the kernel size and the stride are of the same size which ensures non-overlapping convolutions, whereas in resnet18 we have overlapping convolutions. Also, ConvNeXt architecture uses GeLU (Gaussian Error Linear Unit) instead of ReLU for the activation function which prevents the model from dealing with empty neuron problems.
- With ConvNeXt architecture I have used arcface loss function which is used in learning discriminative learning in the embedding space. Also, I have used contrastive learning loss (XBM) which offers larger memory to make more pair comparisons, allowing me to add more data while training.

A. Algorithm implementation

- As part of the model-building process, I have created a ẏml file for each of the model runs. This file contains values of the parameters that are used while running the model. It contains parameters like training and validation path, evaluation data path, number of classes for classification, embedding size, input resolution, model architecture, optimizer, etc.
- Secondly, I have transformed the input image in accordance with the input resolution provided in the ẏml file.
- For the images in the validation dataset, we have extracted the part of the image that is bounded by the box and then converted them into the required resolution.
- Subsequently, I have loaded the specific model with the required parameter in the ẏml file.
- Once, the model is loaded, I have looped over the specified epoch values and start training the model and calculate the error on the validation dataset, and record its value.
- During the training process, I saved the weights obtained from each run and used the weights from the final epoch run into the next model run.
- Similar to the first model run, I have created a separate ẏml file for all the subsequent models and used the weights obtained in the previous model to start.
- In the solution, I started by developing a baseline solution that used resnet18 CNN architecture as the underlying model to find the association between the seller and the user image. This association is measured using the mean average precision (mAP) score. After getting the baseline accuracy I used sequential iterations of advanced models

- arcface and XGB. Since the model run is sequential I used the weights obtained from the previous iteration as an input to the current iteration. Each iteration differs with respect to the resolution of the images and the size of the training data.

B. Pros and Cons

- One of the major advantages of using the ConvNeXt architecture is that it is built on top of Resnet therefore it addresses if not all then most of its shortcomings and therefore guarantees to give better performance.
- In the solution the models are implemented in a sequential manner, therefore in addition to error decreasing with each epoch run for a given model, it will mostly decrease with every model run as well.
- The biggest challenge I have observed is with the time taken by each model. They are computationally expensive. For, some model runs it has also taken more than a day.

C. Own Code

- I have coded the flow of execution of the model. This architecture or flow is used by all the models - Resnet, ConvNeXt with arcface, ConvNeXt with XBM to prepare a classification model. The flow involves performing appropriate transformations on the images like resizing and flipping to introduce data augmentation in the training data. The code also helps in choosing the appropriate parameters like optimizer, scheduler and criterion which are required for the training phase. Further, I have written the training code and the code to validate the weights obtained after training through error metrics. Also, I have created the evaluation code which loads the weights from the final model and calculates the embeddings of the test data to find the association between the seller and user images using mAP.
- The self-coded part forms the complete architecture of the solution as we can use this architecture to run different models with their specific parameter values. Also, once all the models have run, I can load the weights obtained from the final model run $modelrunsaresequential$ and test our model performance on our testing data.

D. Online Resources

- I have taken the help of the online github repository to understand the parameters and implementation of the ConvNeXt CNN architecture with arcface and XBM loss, as it is a relatively new CNN architecture.
- https://github.com/IvanAer/G-Universal-CLIP/tree/main

III. Experimental Protocol

A. Dataset

- I used a human-labeled product image dataset named "Products-10k" for training the arcface model, which is so far the largest production recognition dataset containing 10,000 products frequently bought by online customers, covering a full spectrum of categories including Fashion, 3C, food, healthcare, household commodities, etc. Moreover, large-scale product labels are organized as a graph to indicate the complex hierarchy and interdependency among products. I also used the 'Amazon Review' dataset to use the larger size of the data XBM model.
- I also used the development test with images and ground truth labels. The test set contains 2 files: gallery.csv and queries.csv. gallerycsv defines the database of images from marketplaces. Each row contains the following information:seller_img_id - unique int32 identifier of product image that is used in result ranking NumPy array; img_path - path to the product image in the "data" folder. queriescsv defines a set of user images that will be used as queries to search the database. Each row contains the following information: user_img_id - unique int32 identifier of user image that is used in result ranking NumPy array; img_path - path to user image in "data" folder; bbox_x, bbox_y, bbox_w, bbox_h - bounding box coordinates of the product in the user image.

B. Success Evaluation

- For every image present in the queries.csv file, we found the top 1000 images from the gallery.csv which are very closely associated with it.That association is measured using the mAP value. We use the below formula to calculate the mAP. (AP@n = 1 $\frac{}{GTP\sum\_k\hat{n}P@k\times rel@k)}$
- where GTP refers to the total number of ground truth positives, n refers to the total number of products you are interested in, P@k refers to the precision@k and rel@k is a relevance function. The relevance function is an indicator function that equals 1 if the product at rank k is relevant and equals 0 otherwise. K = 1000 in our case.

C. Compute Resources

- I have used my PC, kaggle, and Google Collab as my compute resources with the default specifications which are freely available.

IV. Results

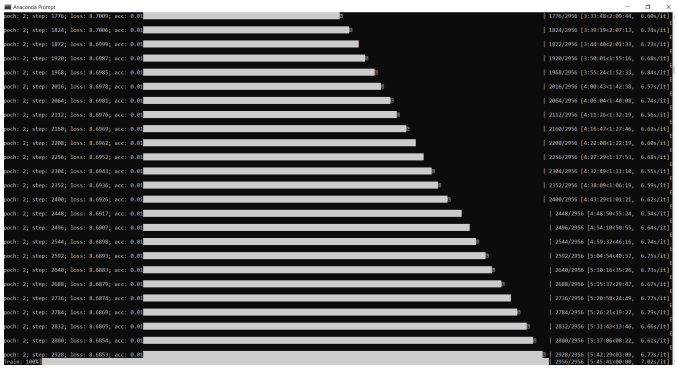

Fig. 1. Epoch run for model training

- The above figure shows the model training phase of the resent18 model, We can see that the loss is subsequently decreasing with each step. However, at the end of the 3rd epoch, the loss is till high 8.6853. Therefore, resulting in lower mAP score on the test dataset.
- mAP value is used to determine the performance of the model. The baseline model which used Resent18 gives mAP = 0.1415 whereas using the arcface and XBM model in succession gives mAP = 0.6147
- From the above mAP values we can say that the ConvNeXt CNN architecture with arcface and contrastive loss performs much better in product recognition. This is because of two main reason: 1 Multiple models are run in succession with changes like increasing the resolution and increasing the data that help improve the training of the model. 2 The model used is better in terms of performance as ConvNeXt is built on top of Resnet and further it used a better loss function which aided in getting better features in the embedding space.

V. Lesson Learned

- The primary goal was to come up with an algorithm that could effectively match user-generated query images (with provided object bounding boxes) to the relevant products in the database of seller photos. This involves overcoming various visual variations to achieve accurate matches. For this, several data manipulation techniques were used like adjusting the resolution, and at the same time augmenting the images to make it more robust to get a correct balance between good accuracy and the complexity of the model that could also effect the resources used.
- In future computer vision problems, I will always consider coming up with the simplest solution with the simpler algorithm available and then try to find out where this algorithm lacks and how a more complex algorithm can address the issues in the primitive model and also at what will be the cost for going ahead with the complex models. Also, it is always better to play around with the images while reading them and before pushing them for training.

References

[1] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, Saining Xie. A ConvNet for the 2020s

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In ICCV, 2019.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In CVPR, 2017.

[4] Github Repo. https://github.com/IvanAer/G-Universal-CLIP/tree/main

[5] https://www.kungfu.ai/blog-post/convnext-a-transformer-inspired-cnn-architecture

[6] http://carpentries-incubator.github.io/capstone-novice-spreadsheet-biblio/04-db/index.html

[7] https://www.easydatatransform.com/normalizedata.html

[8] https://community.dbdiagram.io/t/new-features-composite-foreign-key-and-import-from-sql-server/628

[9] https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms