Vectorized Similarity Search in Multi-modal Databases

Yaohai Zhou, Jo-Hua Wu, Yu-Ruei Chang, Yi-An Chen
UCLA Master of Engineering Department
Los Angeles, CA, 90024
zyh828482@gmail.com, {zoeywu0930, ryan88421, anniechen091}@g.ucla.edu

Abstract

We build a multimodal database including texts and images, which can query images by texts, and vice versa. Textual and visual embeddings are extracted by the techniques of CLIP. kNN and ANN are used as similarity search strategies. We study the performance of this approach and achieve achieved 96% precision rate on MS-COCO dataset. Online demo is available at Demo(GPU) and Demo(CPU).

1 Introduction

There are many kinds of data in the world we live in, including texts, images, web pages, voice, video, etc. With such massive data, the big challenge is to extract information from it. The similarity search system is one of the applications that helps people to organize multimodal data and discover useful information faster. For example, through similarity search engine, people can find out where to buy their desired product by inputting the picture of product. Or they can search for pictures by just inputting words or sentences.

To realize multimodal similarity search system, we divided our work into three major parts in 1: (1) embedding, (2) similarity search, and (3) user interface. The summary of each part is as follows:

- We introduce Contrastive Language-Image Pre-training (CLIP)(8) to embed texts and images into high-dimensional vector spaces and fine-tune pre-trained model to achieve better performance.
- We implement k-Nearest Neighbor (kNN) and Faiss Approximate k-Nearest Neighbor (ANN) to conduct multimodal similarity searches, which include image to text, image to image, text to image, and text to text functions.
- We use an open-source python library Streamlit to construct user interface. It makes everyone can access to and test our work through web easily and dynamically.

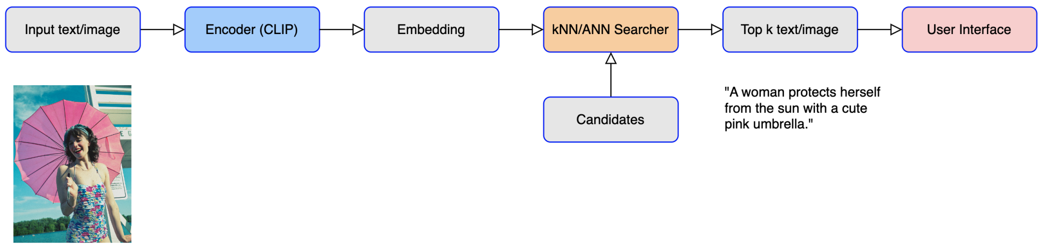


Figure 1: Flow Chart: The input text or image will be encoded by CLIP. Then by using the kNN or ANN similarity searcher, the top k most similar text or image will be retrieved and shown by our streamlit user interface.

Preprint. Under review.

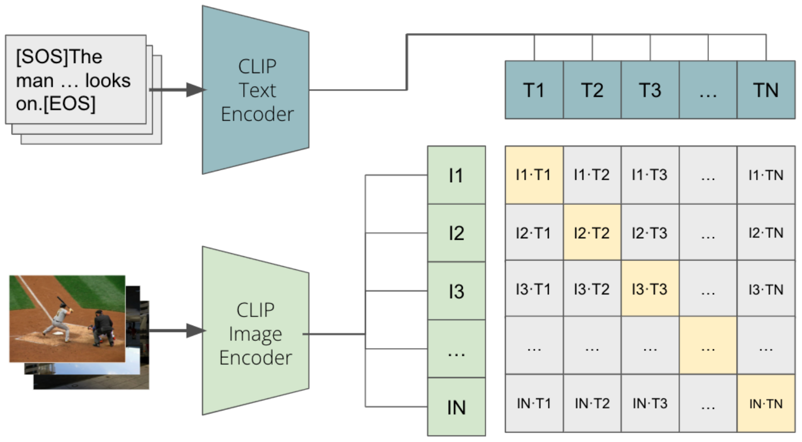


Figure 2: CLIP Architecture

2 Related Work

With the advent of deep learning and its high representative ability in information processing, people always want to explore the relationship between images and texts. First, Alex Krizhevsky et al. (10) achieved excellent performance on the ImageNet dataset (9). However, image context do not have a direct connection with text sequences. The task of providing a natural language description of the content within an image, which is called image captioning, lies at the intersection of computer vision and natural language processing. Vinyals, Oriol et al. (7) used a vision CNN and a language generating RNN to finish the task. However, generating the exact words of the text accompanying each image is a difficult task due to the wide variety of descriptions. Alec Radford et al. (8) found an easier way to build the bridge. It first embeds each image and text into high-dimensional vector and directly calculate their similarity. This way can output correct pairings of (image, text) and make it possible to build a multimodal database full of (image, text) pairs. When we input a new image, we can use k-Nearest Neighbor (kNN) to search the most similar texts as its captions.

To deal with kNN search time complexity issue, many solutions emerged. For hardware aspect, Garcia et al. in (1) proposed a CUDA implementation of the ldquobrute forcerdquo kNN search. Shenshen et al. in (2) presented a CUDA-based parallel implementation of kNN, CUkNN and studied various CUDA optimization techniques to maximize the utilization of the GPU. Besides GPU acceleration, Danopoulos et al. in (3) used FPGA-OpenCL platforms on the cloud to address complexity issue. As for algorithm aspect, Approximate k-Nearest Neighbor (ANN) search appeared for solving this issue. Andoni et al. in (4) studied many efficient ANN solutions that have been proposed in the last two decades.

3 Proposed Approach

3.1 CLIP

CLIP is a multimodal neural network architecture. It can process two kinds of input context which are images and their corresponding captions and gain their high-dimensional vectors. Then CLIP calculates the similarity between each image embedding vector and text embedding vector. Because each image has the highest similarity with its own caption, in the confusion matrix, the values on the diagonal should be the largest in each row and column. On this setting, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training samples.

3.1.1 Model

Architecture CLIP model is composed of two parts, which are text encoder and image encoder shown in 2.

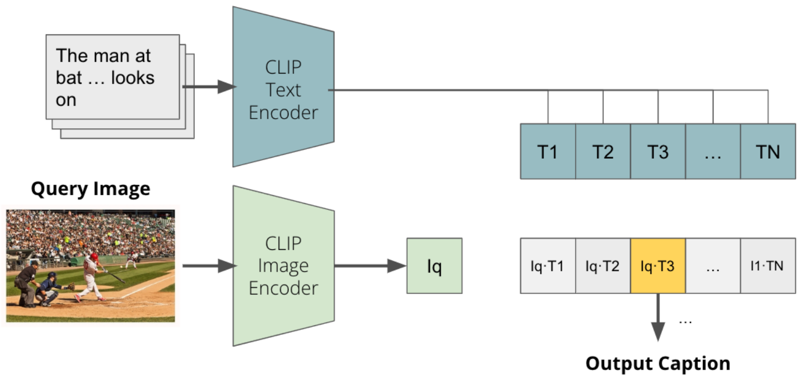


Figure 3: CLIP Prediction Process

The text encoder is a Transformer (5). First, we follow the original setting of CLIP and bracket each text sequence with [SOS] and [EOS] tokens. Then we utilize word embedding to tokenize each text into integers. These tokens are used as input to the Transformer. We calculate the activation map of the last token ([EOS]) as the feature representation for the whole text sequence. Finally, it is linearly projected into the multimodal embedding space.

The image encoder is Vision Transformer (ViT) (6). ViT first split an image into fixed-sized patches to form a text-like sequence and input them into a Transformer Encoder. After many self-attention and feed-forward layers, the output vector is regarded as the feature map for the image.

We generate the image and text embeddings using pretrained ViT-B/32 model and the built-in function `encode_image` and `encode_text`.

Distance Method There are many useful distance methods to calculate the similarity of two vectors, such as L2, cosine and dot product. We use cosine as our distance method. This method has two advantages. First, original CLIP also use cosine method. Second, the value of cosine function is always in range [-1,+1]. It is easily understood as a value for similarity and can be shown on our demo directly. Also when we finetune the pretrained model, small values in the confusion matrix can save much time and computing complexity.

Prediction After building multimodal embedding vector database, a useful application is that it can be used to query texts by inputting an image. Following the process shown in 3, CLIP image encoder embeds the query image into a new vector Iq. Then we load all text embedding vectors from the database and calculate the similarity between each text vector and Iq. Finally, we can output the caption with the highest similarity value.

3.1.2 Finetune

CLIP is trained on WebImageText dataset, which contains 400 million (image, text) pairs. In order to get a better performance on MS-COCO dataset, we finetune the pretrained model on it.

We regard it as a classification problem and use CrossEntropyLoss to update model parameters. Given a batch of N (image, text) pairs, CLIP output $N \times N$ possible (image, text) pairings. In the i-th row, the i-th column should have the largest value. First apply the softmax function to all values in this row. Then apply -log function to the i-th column as i-th loss. Then we sum all loss on each row. Finally, we use AdamW optimizer with a very small learning rate equals to 5e-8 and weight decay equals to 1e-3. We finetune the model for 5 epochs until its validation accuracy converges.

3.2 Similarity Search

For this part, we introduce k-Nearest Neighbor (kNN) and Approximate k-Nearest Neighbor (ANN) searching approaches. Due to the high time complexity of kNN, we further accelerate kNN with

GPU. We tested four scenarios (Image to Text, Text to Image, Image to Image, Text to Text), and the result is shown in 4. The GPU acceleration leads to a 3 times faster overall run time. With this acceleration, ANN: Faiss ANN(100, 5) is still 36% faster than kNN with only 3% accuracy drop. The rest of comparisons for efficiency and performance are shown in 5 and 6. And more details about these two search algorithms will be provided in the following subsections.

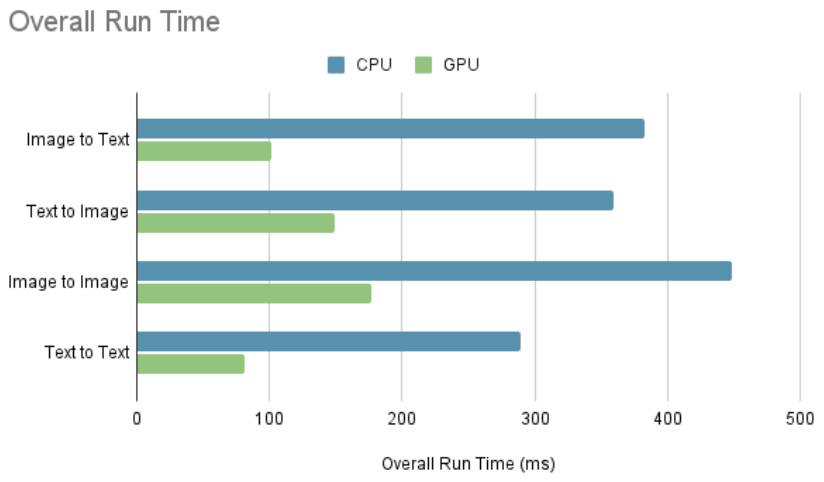


Figure 4: kNN Efficiency Comparison (CPU v.s. GPU)

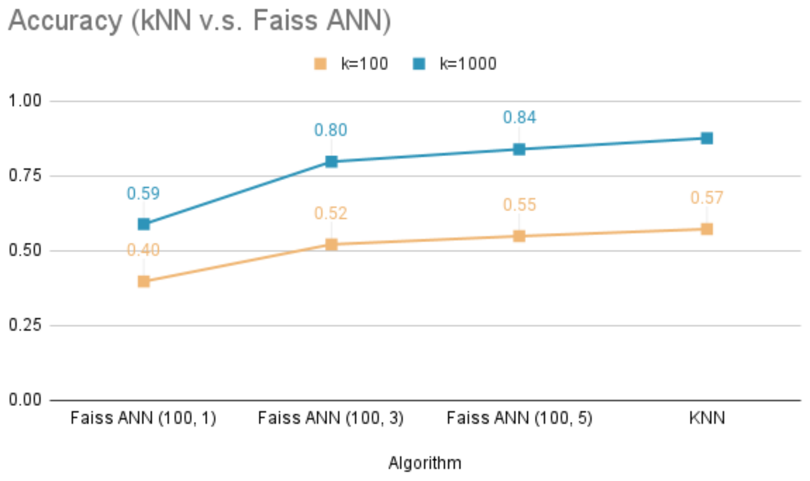


Figure 5: Accuracy Comparison (kNN v.s. ANN)

3.2.1 kNN Search

Let $\mathbf{q}$ be the query vector, and $\{\mathbf{c}_i\}_{i=1}^{n}$ be the search candidates, where n is the total number of candidates. First, the algorithm computes similarity scores with all candidates by chosen distance or similarity method (L2, dot product, or cosine). The similarity scores $s(\mathbf{q}, \mathbf{c}_i)$ are computed as

$$s(\mathbf{q}, \mathbf{c}_i) = \begin{cases} -\sqrt{(\mathbf{q}-\mathbf{c}_i)^T(\mathbf{q}-\mathbf{c}_i)}, \text{ if similarity method = L2} \\ \mathbf{q}^T\mathbf{c}_i, \text{ if similarity method = dot product} \\ \frac{\mathbf{q}^T\mathbf{c}_i}{\|q\|\|c_i\|}, \text{ if similarity method = cosine} \end{cases}$$

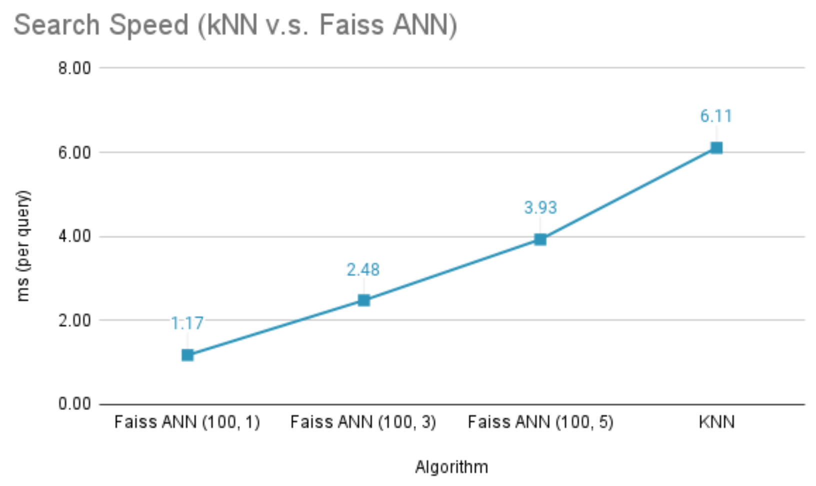


Figure 6: Search Speed Comparison (kNN v.s. ANN)

Then the algorithm ranks $\{c_i\}_{i=1}^{n}$ by similarity scores $\{s(\mathbf{q},\mathbf{c_i})\}_{i=1}^{n}$ with descending order and return the first k candidates as final answers. Therefore, the time complexity of kNN algorithm is $nd + k\log n$, where d is the dimension of $\mathbf{q}$.

3.2.2 ANN Search

We utilize Meta Faiss IVF algorithm as our ANN model. To begin with, it partitions all candidates into m groups and computes the center point of each group as representative. For each query $\mathbf{q}$, it first computes similarity scores with m representatives and selects the most similar t ones. Then, only the candidates within these t groups will be considered to calculate similarities. Like kNN, the most similar k ones will be returned. To analyze, the time complexity of ANN is $(t/m)nd + md + t\log m + k\log(tn/m)$. Since n normally dominates all the other variables, $1 - t/m$ will approximately be the rate of acceleration.

3.3 Streamlit

For user interface, we implemented the package Streamlit, an open-source Python library. Considering the easy use for users, the interface is divided into three parts, it is shown in7.

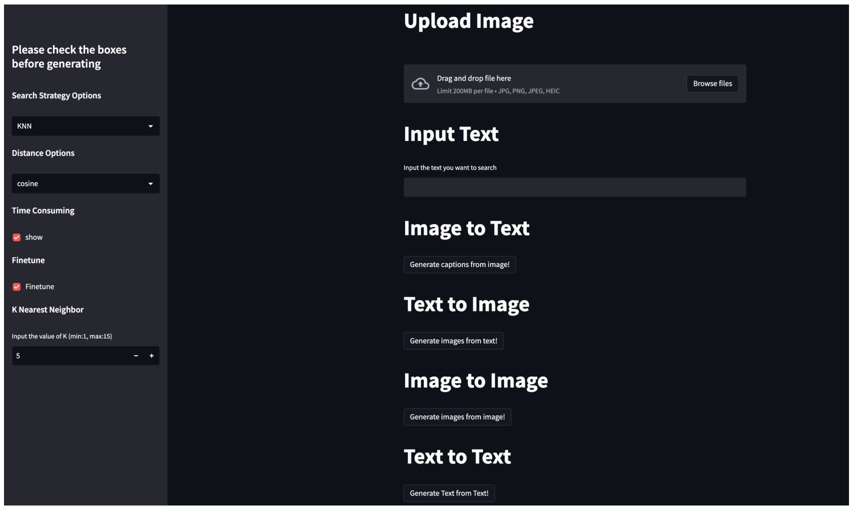
For the first part, users can select searching strategies including KNN search and ANN search based on Meta Faiss IVF algorithm, and also choose whether the outputs are fine tuned based on GPU. It also contains distance options such as cosine, L2 and dot product. Users may also adjust the k value or determine whether they wish to show the executing time. The second part is for users to upload image or input the text they want to search for. As for the last part, we allowed users to utilize four kinds of tools, including Image to Text, Text to Image, Image to Image and Text to Text.

To further demonstrate the performance of our model, we compare the output of finetuned model using cosine as distance option comparing to the original captions from the training set. As shown in 1, the output is on high accuracy.

4 Experimental Evaluation

4.1 Dataset

MS-COCO training dataset and validation dataset have totally 82783 images and 40504 images separately. Each image has more than 5 captions. We randomly sample 80000 (image, text) pairs from training and validation dataset separately and get the embedding vectors to build the main and



Please check the boxes before generating

Search Strategy Options: KNN

Distance Options: cosine

Time Consuming: [✓ show]

Finetune: [✓ Finetune]

K Nearest Neighbor

Input the value of K (min:1, max:15): 5

Drag and drop file here

Limit 200MB per file • JPG, PNG, JPEG, HEIC

Input the text you want to search

Figure 7: Streamlit User Interface

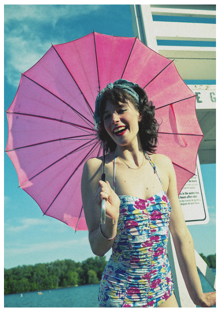
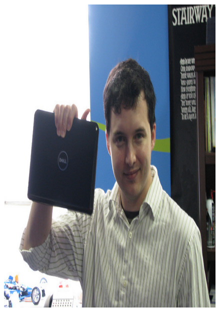
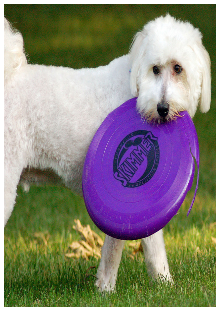
Input Image			
Standard	A woman in a floral swimsuit.holds a pink umbrella.	A man standing up and holding a small, closed laptop computer in one hand.	A white dog holding a purple frisbee in it's mouth.
Ours	A woman protects herself from the sun with a cute pink umbrella.	A man showing a small laptop type device.	A dog has a purple frisbee in its mouth

Table 1: Standard Output vs Our Output

test database. The main dataset can be used to finish Image to Text and Text to Image functions. And we also use main dataset to finetune our model. The test dataset is used to evaluate our finetuned model.

4.2 Evaluation

4.2.1 Metrics

For the evaluation part, we adopt precision rate to evaluate model performance. Precision rate is the ratio of true positives to the total of the true positives and false positives. Precision rate is defined as follows,

$$Precision\ rate = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{\sum_{i=0}^{N} R_i}{Total\ number\ of\ samples}$$

where $R_i = 1$ when the correct answer is contained within k neighbors; $R_i = 0$ when k neighbors don't contain the correct answer.
The higher the precision rate, the better performance the model has achieved.

4.2.2 Comparable analysis between different distance methods

Each graph as shown in 8 indicates the precision rate measured by three distance methods (cosine, dot product and L2) from pretrained models on either training or validation dataset. As the value of k goes up, the precision rate is improved. As figure shows, for both Image to Text] and Text to Image functions, we got the highest precision rate when Cosine is used as distance method in both pretrained and finetune models.
And this result is in line with our expectation since CLIP technique also compute the cosine similarities of the image's encoding.

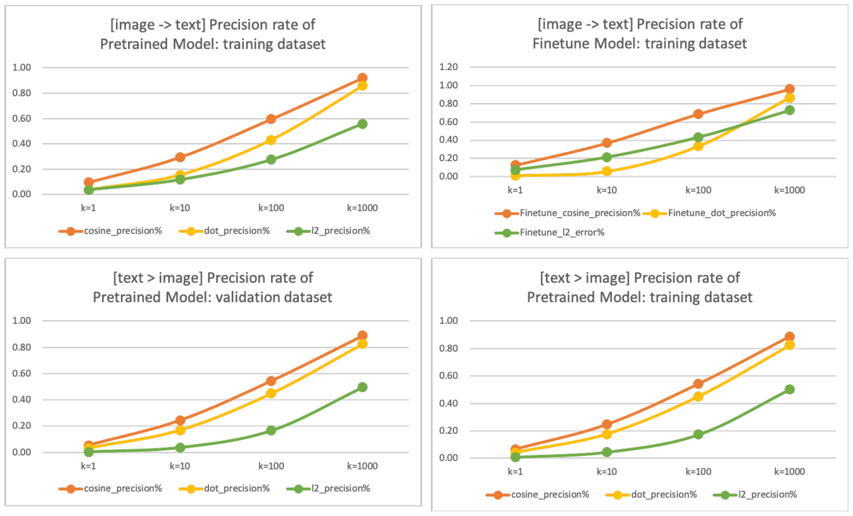


Figure 8: Precision rate measured by three distance methods from pretrained models

We compared our result with the precision rate of a random classifier, which is defined as k divided by total number of samples. As shown in 9, the precision rate is 10% higher than the random classifier when k=1, and twelve times higher when k=1000.

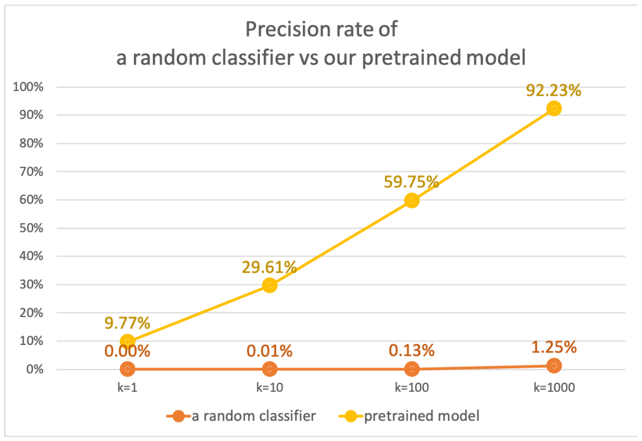


Figure 9: Precision rate of a random classifier vs our pretrained model

4.2.3 Evaluation of fine-tune model

When we first evaluated our pretrained models, we found the results were not satisfactory. For better performance, we finetunde the CLIP model on MS-COCO dataset. We used the training set to finetune the model and used validation set to evaluate its accuracy. As shown in 10, after 5 round finetune, the accuracy increases 6 %. And we also found finetune model is better than pretrained model in our demo.

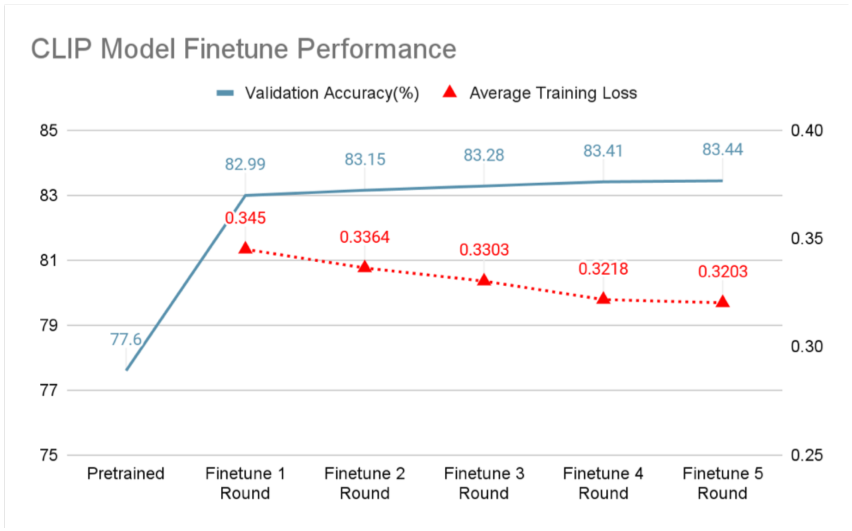


Figure 10: Precision rate measured by three distance methods from pretrained models

After we finetuned the CLIP model, we cut the error rate in half and achieved 96% precision rate in finetune models on both training and validation datasets for both Image to Text and Text to Image functions.

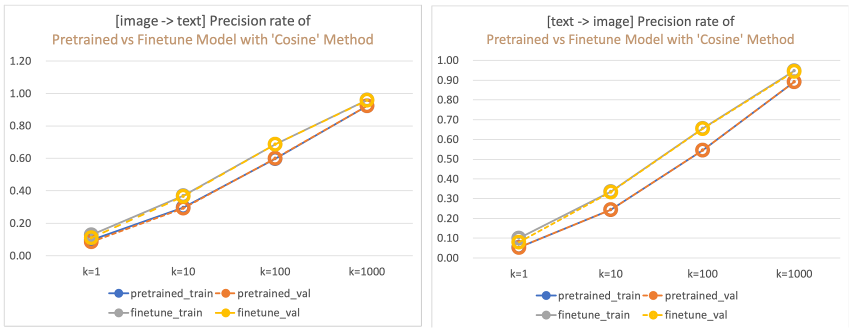


Figure 11: Precision rate of Pretrained vs Finetune Model with 'cosine' Method

5 Conclusion

We have presented the CLIP, a multi-modal vision and language model, specifically adapted to the task of word and image embedding. When each image and its corresponding caption gain their high-dimensional vector, we successfully adopted kNN and ANN to execute the multi-modal similarity search. We fine-tuned our model on MS-COCO dataset and achieved 96% precision rate on both training and validation dataset. To make it easier to use, we used streamlit to build a user query interface, which allows user to easily query similar images and texts.

Future work: At present, our model only takes into account one caption corresponding to one image. As a image has five correct captions in the original dataset, in the future we can consider using multi-classification instead of binary classification and redefine evaluation metrics. As a next step, we intend to incorporate labels into datasets and use them as a part of similarity evaluation. This should lead to additional performance gains as well as improved interpretability of the model.

Acknowledgments

The authors would like to thank Professor Quanquan Gu and the PhD student Weitong Zhang for providing us with a free GPU instance.

References

[1] Garcia, V., Debreuve, E., Barlaud, M. (2008, June). Fast k nearest neighbor search using GPU. In 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (pp. 1-6). IEEE.

[2] Liang, S., Wang, C., Liu, Y., Jian, L. (2009, September). CUKNN: A parallel implementation of K-nearest neighbor on CUDA-enabled GPU. In 2009 IEEE Youth Conference on Information, Computing and Telecommunication (pp. 415-418). IEEE.

[3] Danopoulos, D., Kachris, C., Soudris, D. (2019, July). Approximate similarity search with faiss framework using fpgas on the cloud. In International Conference on Embedded Computer Systems (pp. 373-386). Springer, Cham.

[4] Andoni, A., Indyk, P., Razenshteyn, I. (2018). Approximate nearest neighbor search in high dimensions. In Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018 (pp. 3287-3318).

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Atten- tion is all you need. In Advances in neural information processing systems, pp. 5998–6008, 2017.

[6] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[7] Vinyals, Oriol et al. "Show and tell: A neural image caption generator." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014): 3156-3164.

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. Proceedings of the 38th International Conference on Machine Learning, PMLR 139:8748-8763, 2021.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.