*Human Interaction and Emerging Technologies (IHIET 2023),* Vol. 111, 2023, 168–174
https://doi.org/10.54941/ahfe1004021

# CLIP-Based Search Engine for Retrieval of Label-Free Images Using a Text Query

**Yurij Mikhalevich**

Lightning AI, Dubai, UAE

## ABSTRACT

In January 2021, OpenAI released the Contrastive Language-Image Pre-Training (CLIP) model, able to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the Internet. This model enables researchers to use natural language to reference learned visual concepts (or describe new ones), enabling the zero-shot transfer of the model to downstream tasks. One of the possible applications of CLIP is to look up images using natural language queries. This application is especially important in the context of the constantly growing amount of visual information created by people. This paper explores the application of the CLIP model to the image search problem. It proposes a practical and scalable implementation of the image search featuring the cache layer powered by SQLite 3 relational database management system (RDBMS) to enable performant repetitive image searches. The method allows efficient image retrieval using a text query when searching large image datasets. The method achieves 32.27% top-1 accuracy on the ImageNet-1k 1.28 million images train set and 55.15% top-1 accuracy on the CIFAR-100 10 thousand images test set. When applying the method, the image indexing time scales linearly with the number of images, and the image search time increases minorly. Indexing 50,000 images on Apple M1 Max CPU takes 19 minutes and 24 seconds while indexing 1, 281, 167 images on the same CPU takes 8 hours, 31 minutes, and 26 seconds. The query through 50,000 images on Apple M1 Max CPU executes in 4 seconds, while the same query through 1, 281, 167 images on the same CPU executes in 11 seconds.

**Keywords:** Artificial intelligence, Computer vision, Natural language processing, Image search, Transformers

## INTRODUCTION

Contrastive Language-Image Pre-Training (CLIP) is a neural network trained on a variety of (image, text) pairs. It can be instructed in natural language to predict the most relevant text snippet, given an image, without directly optimizing for the task, similar to the zero-shot capabilities of GPT-2 and 3 (Radford et al., 2021). Authors of CLIP found that CLIP matches the performance of the original ResNet50 on ImageNet “zero-shot” without using any of the original 1.28M labeled examples, overcoming several significant challenges in computer vision.

The capabilities of CLIP also allow researchers to perform image searches using natural language queries. This article explores this particular application of CLIP.

The rapid rate at which the data grows makes this research especially relevant. With the rise of digital devices and the Internet, more and more images are being produced and shared online daily. This exponential growth in data makes it increasingly difficult to find specific images manually. An efficient image search solution can help users and businesses quickly find the images they need amidst this growing sea of data. Additionally, as more photos are being shared online, protecting intellectual property and online security becomes increasingly important. An efficient image search solution can help identify and remove images that violate copyright laws or contain sensitive or inappropriate content. Therefore, with the growing amount of data and images being created, an efficient image search solution is becoming increasingly relevant and necessary.

## RELATED WORKS

There exist multiple methods and solutions for image search.

Traditional image search is based on bag-of-features defined for each image, usually containing a set of labels describing the image. The labels are either specified manually (Fiedler, Bestmann, & Hendrich, 2019), or filled in by the camera (Tesic, 2005), mobile phones (Kim, Lee, Won, & Moon, 2011), or editing software, or generated by an image recognition algorithm (Krizhevsky, Sutskever, & Hinton, 2017), or filled in by a combination of these approaches (Lee, Chen, & Chang, 2006). Then, a search algorithm is applied to these features to perform the image search. The nature of the search algorithm depends on the problem and on the nature of the labels. For example, if we are dealing with GPS coordinates, we can use a distancebased search algorithm that will look up pictures geographically related to the query (Zhang, Hallquist, Liang, & Zakhor, 2011). If we are dealing with text labels, we can use a text-based search algorithm to look up pictures based on text queries. For this, we can choose from a variety of different techniques ranging from substring matching to a search algorithm based on lemmatization (Balakrishnan & Lloyd-Yemoh, 2014) to using word embeddings (Günther, 2018) (Kenter & De Rijke, 2015). We can get more useful results in a practical setting if we combine multiple approaches and features, like GPS coordinates, text labels, the date picture is taken, the camera model, focal distance, etc (Ismail, 2011).

None of these approaches allows us to search for images using a text query without having to specify the features beforehand. The techniques described above also do not allow performing a text-based search for a concept not present in the prepared image labels, even if this concept is present in the image itself. These are the problems that OpenAI's CLIP (Radford et al., 2021) enables us to solve.

## METHOD

With CLIP's text transformer, it is possible to convert a text query to an n-dimensional vector (where n differs depending on the CLIP model used). With CLIP's image transformer, it is possible to convert an image to an n-dimensional vector. Then we can calculate the dot product (Equation 1) of the normalized query vector and each of the normalized image vectors. If we then sort the images by the decreasing dot product and take top-k images, we will get the k images that match the query the most.

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{1}$$

While this approach works reasonably fast on a few images, it will scale poorly with the increase in the number of images. Especially if there is no access to GPUs to run the CLIP model on. To make the querying scalable, the solution proposed in this paper caches the image vectors and, when the user executes repeated queries, only computes the query vector. This approach allows users to execute repeated queries quickly, without having to wait for the image to be processed each time that they query.

The solution also allows adding new images to the cache to avoid recomputing the whole cache when the image catalog is updated.

Having these mechanisms in place allows the users to use CLIP in practical applications of image search.

## IMPLEMENTATION

The method described above is implemented in the Python utility called rclip (Mikhalevich, 2023). rclip provides an easy-to-use CLI interface that allows users to search images within any directory on a computer where rclip is installed. Search within nested directories is also supported. To use it, the user should open the terminal, navigate to the directory containing the files that they want to search through, type "rclip <search query>," and hit "Enter."

The solution uses OpenAI's clip library (Radford et al., 2021) to load the model and compute the feature vectors. rclip uses ViT-B/32 version of the CLIP model. The code of the feature computing methods is shared below:

```
def compute_image_features(self, images: List[Image.Image]) ->
np.ndarray:
   images_preprocessed = torch.stack([self._preprocess(thumb) for
thumb in images]).to(self._device)
   with torch.no_grad():
     image_features = self._model.encode_image(images_preprocessed)
     image_features /= image_features.norm(dim=-1, keepdim=True)
   return image_features.cpu().numpy()

   def compute_text_features(self, text: List[str]) -> np.ndarray:
     with torch.no_grad():
       text_encoded=
self._model.encode_text(clip.tokenize(text).to(self._device))
       text_encoded /= text_encoded.norm(dim=-1, keepdim=True)
     return text_encoded.cpu().numpy()
```

rclip features the image vector cache implemented using SQLite 3 RDBMS (Hipp, 2021). The image vector is computed and added to the index only if the cache does not already contain an entry for a given image. The vectors are cached by image paths. This allows the user to execute repeated queries over the same image catalog without waiting for the image vectors to be recomputed.

Here is how the structure of the cache is defined:

```
self._con.execute('''
  CREATE TABLE IF NOT EXISTS images (
    id INTEGER PRIMARY KEY,
    deleted BOOLEAN,
    filepath TEXT NOT NULL UNIQUE,
    modified_at DATETIME NOT NULL,
    size INTEGER NOT NULL,
    vector BLOB NOT NULL
  )
''')
```

The rclip source code is published on GitHub under the MIT license: https://github.com/yurijmikhalevich/rclip.

## PERFORMANCE

rclip was benchmarked using two different CLIP models, ViT-B/32 (smaller CLIP model) and ViT-L/14@336px (larger CLIP model), on a NAS running

Intel(R) Celeron(R) CPU J3455 @ 1.50GHz. Table 1 shows how rclip performs when indexing and searching through 269 photos when running on this CPU.

As Table 1 shows, the ViT-L/14@336px performance will not scale well, which makes rclip unusable in practical scenarios when running CLIP on low-level and mid-level consumer CPUs. This is why rclip uses ViT-B/32.

Running rclip indexing with ViT-B/32 on 72,769 photos on the same NAS powered by Intel(R) Celeron(R) CPU J3455 @ 1.50GHz took 23 hours.

Performing a query over 72,769 photos takes 56 seconds.

To give a better understanding of how rclip performance scales, Table 2 shows how rclip performs when indexing and searching through 50k images and 1.28m images on the Apple M1 Max CPU. As Table 2 shows, the indexing time scales linearly with the number of images when the search time increases only slightly, even when going from searching through 50 thousand images to searching through 1.28 million images.

**Table 1.** Indexing and search performance on Intel(R) Celeron(R) CPU J3455 @ 1.50GHz using different CLIP models.

| Model | Indexing time | Search time |
|---|---|---|
| ViT-B/32 | 3m56.626s | 0m18.064s |
| ViT-L/14@336px | 125m0.507s | 3m19.742s |
| *Difference* | *x31.70* | *x11.06* |

**Table 2.** Indexing and search performance on Apple M1 Max CPU using ViT-B/32.

| Dataset | # of images | Indexing time | Search time |
|---|---|---|---|
| ImagNet-1k validation set | 50k | 19m24.750s | 0m4.04s |
| ImagNet-1k train set | 1.28m | 8h31m26.680s | 0m11.49s |
| *Difference* | *x25.62* | *x26.35* | *x2.84* |

## SEARCH QUALITY

rclip achieves 32.27% top-1 accuracy and 45.26% top-5 accuracy rate on the ImageNet-1k (Russakovsky et al., 2015) 1.28 million images train set and 55.15% top-1 and 81.34% top-5 accuracy on the CIFAR-100 (Krizhevsky, Hinton, et al., 2009) 10 thousand images test set. Worth noting that rclip performs better on the ImageNet-1k dataset when the text prompt is constructed as "photo of class" instead of "class." See Table 3 for more details.

To get a better understanding of rclip's performance, see Figures 2, 3, 4, and 5, which show search results for a search performed on a demo set of 142 images (see Figure 1).

**Table 3.** rclip search quality.

| Model and prompt | Top-1 accuracy | Top-5 accuracy |
|---|---|---|
| ImageNet-1k 1.28m prompt: "class" | 31.17% | 44.80% |
| ImageNet-1k 1.28m prompt: "photo of class" | 32.27% | 45.26% |
| CIFAR-100 10k prompt: "class" | 55.15% | 81.34% |
| CIFAR-100 10k prompt: "photo of class" | 53.13% | 78.71% |

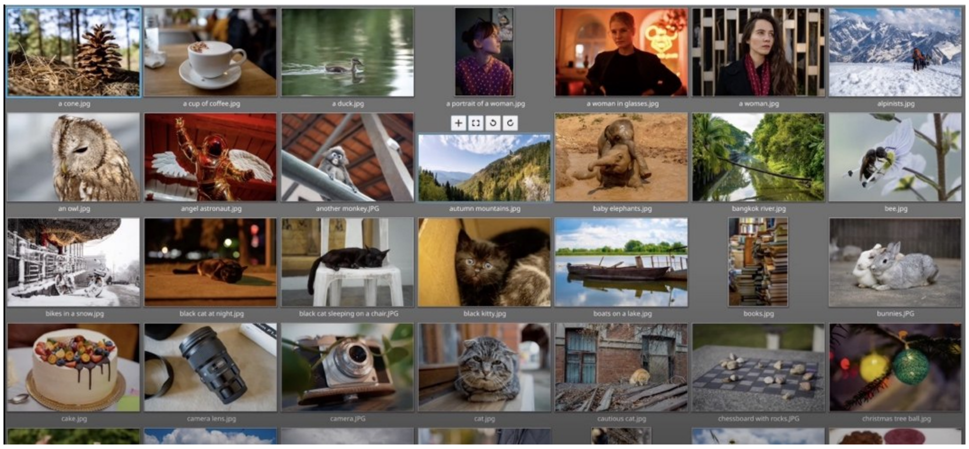

**Figure 1:** Demo set sample.

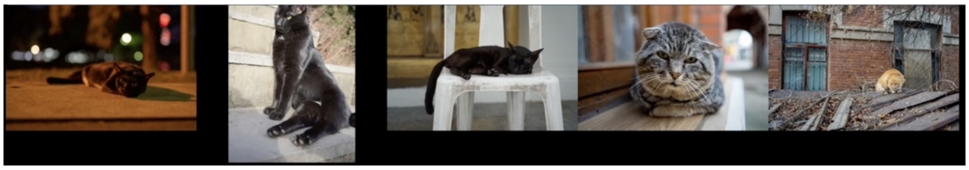

**Figure 2:** Search result for the query "cat".

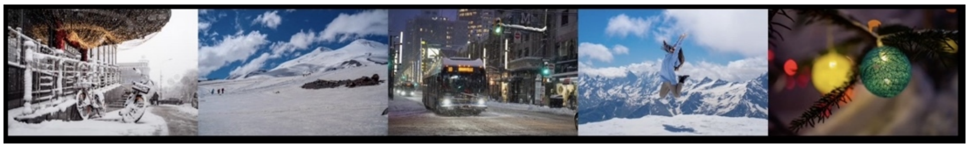

**Figure 3:** Search result for query "snow".

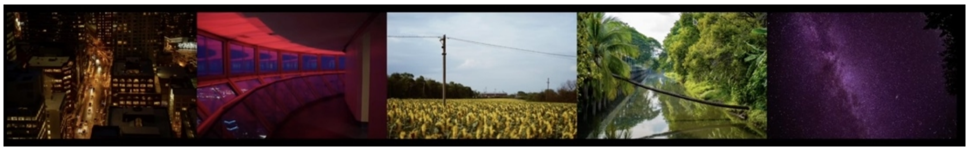

**Figure 4:** Search result for the query "leading lines".

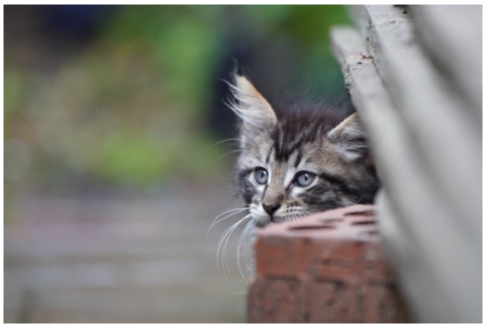

**Figure 5:** Top result for the query "a kitten peeking from behind a corner".

## CONCLUSION

rclip, a command-line utility for image search that is easy to use, was developed using OpenAI's CLIP model. The utility is highly efficient and practical, but there are still ways to optimize its performance further. Future plans include creating a distinct model for the CLIP text transformer and loading only it when users conduct a search that does not require indexing, omitting the loading of the CLIP vision transformer. Additionally, future plans include improving rclip by ensuring it does not re-index files when they are renamed. These upgrades would increase the utility's efficiency, but even with the current performance, rclip is an incredibly valuable tool.

## REFERENCES

Balakrishnan, V., Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances.

Fiedler, N., Bestmann, M., Hendrich, N. (2019). Imagetagger: An open source online platform for collaborative image labeling. In Robocup 2018: Robot world cup XXII 22 (pp. 162–169).

Günther, M. (2018). Freddy: Fast word embeddings in database systems. In Proceedings of the 2018 international conference on management of data (pp. 1817–1819).

Hipp, R. D. (2021). SQLite (Version 3.32.2) [Computer software]. https://www.sqlite.org/index.html

Ismail, M. M. B. (2011). Image annotation and retrieval based on multi-modal feature clustering and similarity propagation. University of Louisville.

Kenter, T., De Rijke, M. (2015). Short text similarity with word embeddings. In Proceedings of the 24th acm international on conference on information and knowledge management (pp. 1411–1420).

Kim, J., Lee, S., Won, J.-S., Moon, Y.-S. (2011). Photo cube: an automatic management and search for photos using mobile smartphones. In 2011 IEEE ninth international conference on dependable, autonomic and secure computing (pp. 1228–1234).

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

Lee, B. N., Chen, W.-Y., Chang, E. Y. (2006). A scalable service for photo annotation, sharing, and search. In Proceedings of the 14th acm international conference on multimedia (pp. 699–702).

Mikhalevich, Y. (2023). Rclip (Version 1.2.5) [Computer software]. https://github.com/yurijmikhalevich/rclip/tree/v1.2.5

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S.,... Sutskever, I. (2021). Learning transferable visual models from natural language supervision. arXiv. Retrieved from https://arxiv.org/abs/ 2103.00020 doi: 10.48550/ARXIV.2103.00020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S.,... FeiFei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3), 211–252. doi: 10.1007/s11263-015-0816-y.

Tesic, J. (2005). Metadata practices for consumer photos. IEEE MultiMedia, 12(3), 86–92.

Zhang, J., Hallquist, A., Liang, E., Zakhor, A. (2011). Location-based image retrieval for urban environments. In 2011 18th ieee international conference on image processing (pp. 3677–3680).