EXPLORING IMAGE AND VIDEO SEARCH

Ng Shi Qing Eugenia[1], Wu Jieyu Heidi[2], Xiao Xuhong[3]
[1]River Valley High School, 6 Boon Lay Avenue, Singapore 649961
[2]Nanyang Girls' High School, 2 Linden Drive, Singapore 288683
[3]DSO National Laboratories, 12 Science Park Drive, Singapore 118225

1. Abstract

With the greater reliance on mobile devices to capture images of our key moments in our everyday lives, it can be difficult to scroll through all those images just to find a specific image. CLIP is a pre-trained model that can understand the connection between the semantic meaning of words and the image features associated with it. Therefore, this paper explores the use of CLIP to make a more advanced image and video retrieval that can take in longer and more specific textual descriptions, with an accuracy that is better than other similar services.

2. Introduction

Recent technological advancements have paved the way for significant milestones in the development of cameras in mobile phones. As these cameras become increasingly sophisticated, photo-taking with such cameras is taking the world by storm due to its convenient nature. However, retrieval of these images is a potential hassle as photos are automatically stored in chronological order, and the time-consuming, impractical process of scrolling through countless images to find a specific one wastes users' precious time. This process can be mitigated via utilising machine learning technology to retrieve images.

This project aims to explore the use of CLIP (Contrastive Language-Image Pre-Training), a pre-trained model by OpenAI [1], to create an advanced image and video retrieval system that can firstly take in specific queries, such as textual descriptions of the focal objects in the image, and output relevant based on these queries; and secondly output similar images with a given input image.

Briefly, this project focuses on three different areas of image and video retrieval: using a text query to output relevant images, using an image query to output relevant images, and using a text query to output relevant videos.

3. Literature Review

3.1 Contrastive Language-Image Pre-Training (CLIP)

CLIP is the first multi-modal model to combine computer vision (CV) and natural language processing (NLP) technology and has produced results that surpasses many of the current state-of-the-art models. Its unconventional training method, which will be discussed at length later on, allows it to understand the semantic meanings of words. The CLIP model uses contrastive learning to train their model, a method which uses image and text transformers to embed (image, text) pairs, and tries to maximise the cosine similarity between correct (image, text) pairs and minimise the cosine similarity between wrong (image, text) pairs.[2] The unique ability of the model being able to understand words has rendered it useful in a variety of innovative projects like Image Generation[3] and Robotic Manipulation[4].

3.1.1 Training speed

The most prominent issue faced when training machine learning models is the extremely unpragmatic process of manually labelling training data, which utilises too much manpower and time. To provide an example, the dataset used for ImageNet models had required 25,000 workers to manually annotate 14 million images for 22,000 image object categories.[5] With

CLIP, which is trained on (image, text) pairs, individual images used for the training dataset no longer have to be manually labelled, allowing the dataset to be composed of images found via the internet and the text paired with it, such as tags and captions attached to the original image. By drastically reducing the manpower required to train the model, and the time required to collate the training dataset, the CLIP model can be trained more efficiently.

3.1.2 Zero-shot capabilities

As CLIP is more efficiently trained as compared to other models, its training dataset is maximised as the same amount of computational time and power can be invested into a larger dataset. As CLIP learns what each word means and can therefore understand a phrase or sentence, even if has not been trained on that particular phrase before, the large dataset of 400 million image-text pairs[2] allows the vocabulary size of the model to be maximised to understand a wide range of words. CLIP then makes use of a text transformer and feature extraction to map the meanings of the words to features in the images. The model can then better evaluate test cases and data that it has not been previously trained on, increasing its zero-shot capabilities.

3.1.3 Model adaptability

Since the CLIP model learns the semantic meaning of words, it is able to accurately classify the relevant images even if it was not directly trained for that particular query. When compared to a different vision model such as ImageNet, the CLIP model can be used to classify objects that it had never encountered in its training dataset, whilst an ImageNet model can only classify objects that it was trained for and is unable to accurately predict objects that are out of its training data. This is clearly illustrated in a comparison done by OpenAI to compare the accuracy of the ImageNet ResNet101 model and the CLIP ViT-L model when classifying a range of test sets. The accuracy of CLIP was the same, or in most test sets, even more accurate than the ResNet101 model. CLIP outperformed the ImageNet model even when it was not trained on the original 1.28M labelled examples.[5] The results of said comparison can be seen in Fig 3.1. Its zero-shot capabilities let the CLIP model be used in a range of contexts without the need to be fine-tuned and re-trained using new training data, allowing it to be easily adaptable into a plethora of projects.

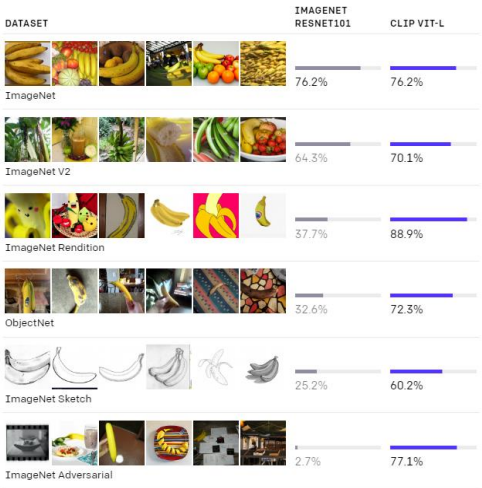

Fig 3.1 taken from https://openai.com/blog/clip/

3.2 Other image retrieval systems

Image retrieval systems do exist in the current technological scene, such as the built-in image retrieval system in the *Photos* app on iPhones and Google Photos (https://www.google.com/photos/about/). However, these systems often only utilise feature extraction to classify images, and hence can only classify images based on more generic descriptions, such as types of animals, locations, and monuments. These systems can also identify text that is present in images and, when the text query appears in the image in the form of text, will output the image as a relevant result. Another feature of such systems would be the detection and compilation of similar faces and objects into albums. However, these are generally ineffective as the images are categorised by features rather than captions or image descriptions, and therefore cannot be easily retrieved via a textual description of the image. False negative results are also common, which proves the lack of accuracy and precision of such systems. A comparison between this project's image and video retrieval system and Google Photos with the same dataset can be seen below. Fig 3.2 shows the results when the query "me walking down the road with the northern lights" is entered on Google Photos. While Fig 3.2 and Fig 3.3 is the output when the same query is entered in our image and video retrieval system.

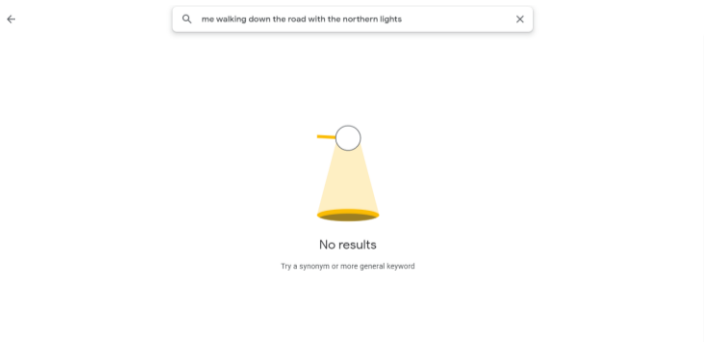

Fig 3.2

```
Enter query: me walking down the road with the northern lights
[(['YsJWJjNMTpo.jpg'], 0.3230439310507483), (['f4S3rQiCRSc.jpg'], 0.2656734090600756), (['t5GekS4FNVk.jpg'], 0.25076732311062333)]
```

Fig 3.3

Fig 3.4

The project's image retrieval system can understand longer, more descriptive, and contextualised queries, such as "a dog walking at the beach" or "me walking down the road with the northern lights". Using the CLIP model as the basis of this image and video retrieval system allows for a more accurate and versatile system and makes finding any image easier and more convenient.

4. Methods

4.1 Dataset storage

Google Drive was used to store the images and videos as it was a cloud-based storage platform and could significantly reduce the amount of storage space needed on the device. It would also make it significantly easier for the code to access the images, especially since our code was written in Google Colaboratory and our Google Drive could be mounted on it.

4.2 Saving preprocessed and encoded images

After preprocessing and encoding the images and videos, we decided to save it into a Comma-Separated Values (CSV) and pickle file respectively. We decided to do this as preprocessing and encoding every single image each time we run the code would be extremely inefficient and simply impractical. With a database size of 200 images, it took around 8-9 minutes to run each time we input a query. Therefore, in order to shorten the time taken for the code to run, each time there was a new image added to the Google Drive, it would be preprocessed and encoded once, before adding it to either a csv or a pickle file so as to quicken the retrieval process afterwards. The difference in file types when saving images and frames of a video is due to the limitations of the libraries we used. As we used the CV2 library to extract the frames from a video, when appending it to the CSV, it would be saved as a string instead of a Tensor object. Therefore, in order to resolve that issue, we decided to save the frames of the video to a pickle file.

4.3 Formula used in calculating similarity

The CLIP model was used to preprocess, encode and embed both the images in the Google Drive database and the input queries. CLIP has built in transformer models that encodes and embeds both the text query and image output onto a mathematical space as a vector. We decided to compare the similarity between the query and output image by calculating the cosine similarity. The cosine similarity formula allows us to find the cosine of the angle between both the text and image embeddings and prevents the Euclidean distance between both vectors from affecting the accuracy of the comparison. Other similarity measures such as by Euclidean Distance can be affected by the size of the data. If 2 data points have similar sizes but are describing completely different things, the similarity between these 2 data will be high due to the Euclidean distance between these points being small. In contrast, cosine similarity measures the angle formed between the 2 data points and takes into account the orientation of the data points and not the distance. The data points can have a big Euclidean distance but small angle, which can more accurately show that they are similar.[6]

4.4 Calculating cosine similarity

Below is the formula for calculating cosine similarity:

$$cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The cosine similarity is found by dividing the dot product of both the text and image vector by the product of the magnitude of both vectors.[7] The smaller the angle, the closer the value is to 1. To sum it up, the similarity between a text and an image can be compared, allowing us to make an accurate image and video retrieval system just by matching the meaning of the query to the features in the images. Fig 4.1 shows how the cosine similarity score is minimised for the wrong text description and maximised for the correct text description.

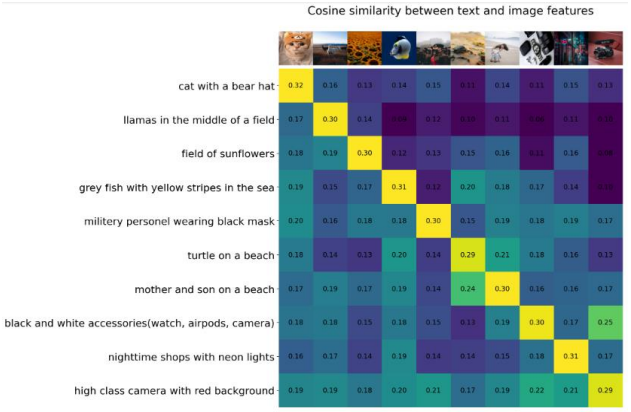

Fig 4.1

4.5 Difference between text query and image query

However, one interesting point to note is that when testing both text queries and image queries, we found that the model was significantly more confident when comparing image queries as compared to text queries. The difference in confidence level was generally between 0.5-0.6, with the average confidence level of an image query being 0.9 and the confidence level of an image query being 0.3. Fig 4.2 shows the output when the query is a text while Fig 4.3 shows the output when the query is an image.

```
Enter query: a field of sunflowers at sunset
[(['5dpSx2hBtj0.jpg'], 0.3026748891141233),
```

Fig 4.2

```
Enter filename: 5dpSx2hBtj0.jpg
[(['9zvw7In3vhA.jpg'], 0.9355786314361606),
```

Fig 4.3

4.6 Outputting relevant images

A confidence level threshold was also implemented as images with cosine similarity values below a certain level were irrelevant to the query inputted, which would not be useful to the user. Therefore, we decided to output the top 5 images with cosine similarity higher than 0.25 for text queries and 0.70 for image queries. If there are less than 5 images that exceed the threshold, then we will output a maximum of the top 5 images with cosine similarity higher than 0.25 for text queries and 0.70 for image queries. This difference in confidence level threshold is due to the observation mentioned earlier where image queries then have a higher confidence level presumingly because it is easier to compare the similarity between features than between semantic meaning of words and features. Fig 4.4 and Fig 4.5 is a block diagram of the process behind how the image and video retrieval system works.

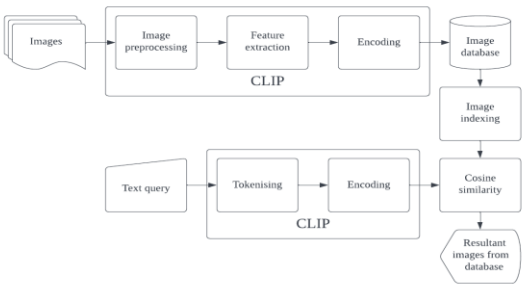

Fig 4.4

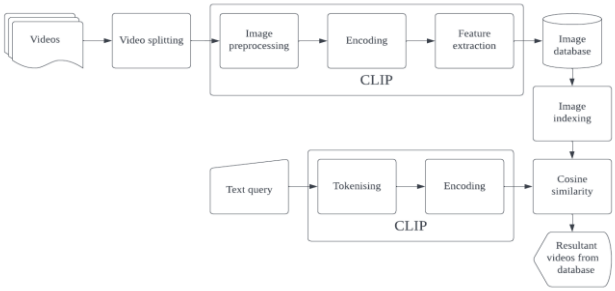

Fig 4.5

4.7 Handling different file formats

As we aim to contextualise our image and video retrieval system and to make it as user friendly as possible, when displaying the output images, we decided to use the IPython display module and Pyheif library. As images in iPhones can be stored as 2 formats, .heic (High Efficiency Image Container) and .jpg (Joint Photographic Experts Group), we wanted our code to be able to run even with these different file formats. Therefore, we also decided to use the Pillow-Heif library together with the Pillow library to open the image files before preprocessing it due to the Pillow library's inability to handle .heic file formats.

5. Results

5.1 Text query to image:

When the input text query is entered, the output would be the filename and the cosine similarity for each image output. Below are 3 examples of a text query to image output.

Example 1:

```
Enter query: cute cat in a sock
[(['IMG_5913.JPG'], 0.35764384915862374), (['IMG_5910.JPG'], 0.28867614486266946)]
```

Fig 5.1

Fig 5.2

Fig 5.3

Example 2:

```
Enter query: having a campfire at the beach by the sea during sunset
[(['FGZBXeQxDSM.jpg'], 0.3230367605892465), (['JZ-MwvgwnHw.jpg'], 0.2547232465909637)]
```

Fig 5.4

Fig 5.5

Fig 5.6

Example 3:

```
Enter query: man in red shirt and orange helmet cycling down a road  in the countryside
[(['nL2ZA7J-9Rs.jpg'], 0.32122120590886005)]
```

Fig 5.7

Fig 5.8

5.2 Image query to Image

The code will take in a filename as an input. When it is entered, the output would be the filename and the cosine similarity for each image output. Below is an example of an image query to image output. The first image of each example (Fig 5.10, Fig 5.13 and Fig 5.20) is the reference image input while the rest shows the output (the code only shows the latter).

Example 1:

```
Enter filename: IMG_5913.JPG
[(['IMG_5910.JPG'], 0.7686859490670324)]
```

Fig 5.9

Fig 5.10

Fig 5.11

Example 2:

```
Enter filename: 5dpSx2hBtj0.jpg
[(['9zvw7In3vhA.jpg'], 0.935578548481322), (['sWC1NIg93fg.jpg'], 0.9241300496129311), (['Qf4J7_YqBgY.jpg'], 0.9032054017569249)]
[(['oP-z4YcRXhs.jpg'], 0.8785586703063424), (['hT-yJurs95o.jpg'], 0.8766143475472589)]
```

Fig 5.12

Fig 5.13

Fig 5.14 *Fig 5.15* *Fig 5.16* *Fig 5.17* *Fig 5.18*

Example 3:

```
Enter filename: 9s8DUVb3y4U.jpg
[(['pdBsg-29fq8.jpg'], 0.747192572068082), (['RD_VnDHL4oM.jpg'], 0.7336918815879486)]
```

Fig 5.19

Fig 5.20 *Fig 5.21* *Fig 5.22*

5.3 Text query to Video:

When the input text query is entered, the output will be the video filename, the timestamp of the frame and the cosine similarity. Below are 3 examples of the text query to video output:

Example 1:

```
Enter a query: dog catches frisbee in midair in a forest
['mixkit-dog-catches-a-frisbee-in-mid-air-1557-medium.mp4', 1.0, 0.36504608]
['mixkit-dog-catches-a-frisbee-in-mid-air-1557-medium.mp4', 6.0, 0.34942657]
['mixkit-dog-catches-a-frisbee-in-mid-air-1557-medium.mp4', 3.0, 0.3491227]
['mixkit-dog-catches-a-frisbee-in-mid-air-1557-medium.mp4', 4.0, 0.33171195]
['mixkit-dog-catches-a-frisbee-in-mid-air-1557-medium.mp4', 5.0, 0.31167498]
```

Fig 5.23

Fig 5.24 *Fig 5.25* *Fig 5.26* *Fig 5.27* *Fig 5.28*

Example 2:

```
Enter a query: talking and laughing with friends during a Christmas dinner over  bread and wine
['mixkit-recording-christmas-dinner-for-social-networks-42147-medium (1).mp4', 4.0, 0.28044185]
['mixkit-christmas-dinner-with-friends-42144-medium.mp4', 4.0, 0.27639288]
['mixkit-christmas-dinner-with-friends-42144-medium.mp4', 14.0, 0.27282947]
['mixkit-christmas-dinner-with-friends-42144-medium.mp4', 12.0, 0.27170888]
['mixkit-christmas-dinner-with-friends-42144-medium.mp4', 2.0, 0.26953295]
```

\Fig 5.29

Fig 5.30 *Fig 5.31* *Fig 5.32* *Fig 5.33* *Fig 5.34*

Example 3:

```
Enter a query: owner using a yellow straw to play with a grey and white cat
['mixkit-pet-owner-playing-with-a-cute-cat-1779-medium.mp4', 14.0, 0.3214167]
['mixkit-pet-owner-playing-with-a-cute-cat-1779-medium.mp4', 6.0, 0.2996389]
['mixkit-pet-owner-playing-with-a-cute-cat-1779-medium.mp4', 10.0, 0.2772224]
['mixkit-pet-owner-playing-with-a-cute-cat-1779-medium.mp4', 8.0, 0.27604714]
['mixkit-pet-owner-playing-with-a-cute-cat-1779-medium.mp4', 4.0, 0.2739378]
```

Fig 5.35

Fig 5.36 *Fig 5.37* *Fig 5.38* *Fig 5.39* *Fig 5.40*

6. Discussion

6.1 Importance of image retrieval systems

We now live in an era where digital memories are getting increasingly popular, and to be able to retrieve these memories in the future, a more advanced image and video retrieval system needs to be in place. By using CLIP to make a more advanced image retrieval system, we can make lives more convenient for users by replacing unnecessary, impractical processes of manually scrolling through large numbers of images with an efficient model with low time complexity.

6.2 Effectiveness of CLIP for image retrieval

CLIP has proven itself to be a multiskilled model that is effective in making an image retrieval system. The results that we managed to achieve are significantly better than other image retrieval models, even without fine-tuning and retraining the model. It can handle more complicated queries that are not just nouns describing the object.

6.3 Limitations

However, there are still a few limitations to our code. Although we set the threshold for the relevant images at 0.25 and 0.7, there were still some false negatives outputs. This could be due to the significant difference in colour or features. Our untested threshold could also be a

potential source of error and a reason our model has inaccurate outputs and false negatives and false positives. In order to resolve this issue, we could find the lowest cosine similarity score of the images relevant to the query and use the average of that score among 20 random text inputs and set that to be the threshold.

Another limitation that could affect the accurate collection of data in this project is the small size of our database. We only have around 300 images in our database, which makes it incredibly difficult to find similar images to test the model with. Unfortunately, because of this, we are unable to test how the model fares in differentiating between similar images and what the difference is, if any, when comparing and calculating the cosine similarity of the images and input query.

6.4 Further improvements

If we were to expand this project further, we would definitely look into turning our code into a working web application or a mobile app for image and video retrieval. Currently, most people would use their phones to take photos and we think that it would have greater potential to improve people's lives if it were turned into a mobile app. We would also investigate into how to further improve the output of our text to image retrieval system. Currently, it only outputs the frame, video filename and the timestamp of the frame. If possible, we would compile the timestamps and output the video and the timeframe where the query is relevant. This way, it would be easier for the user to find and skip to the part of the video that is actually relevant to the user's query. This can save the unnecessary time taken to watch the whole video just to find the part of the video that the user is looking for.

7. Conclusion

To sum up this whole project and research paper, we demonstrated CLIP's effectiveness in understanding the connection between the semantic meanings of words and the features associated with it and how we can use it to construct a better image and video retrieval system than those in the technological scene now.

Acknowledgements

We would like to express our utmost gratitude to everyone who has helped us in this project. This project would not have been possible without Dr Xiao Xuhong, our mentor who guided us along this project and provided extremely valuable advice whenever we had any queries or problems. We would also like to thank Vera, Enrui and Qi Heng for their honest feedback and encouragement. Lastly, we would like to thank DSO National Laboratories (DSO) and the Young Defence Scientist Programme (YDSP) for granting us this opportunity to conduct the project and to be a part of this experience.

References

[1] OpenAI et al. (2021). CLIP. [Source Code]. https://github.com/openai/CLIP .\

[2] Radford, A. et al. (2021). Learning Transferable Visual Models From Natural Language Supervision, p1-4. https://arxiv.org/abs/2103.00020

[3] Ramesh, A. et al. (2022) Hierarchical Text-Conditional Image Generation with CLIP Latents. https://arxiv.org/abs/2204.06125

[4] Shridhar, M., Manuelli, L. & Fox, D. (2021). CLIPORT: What and Where Pathways for Robotic Manipulation. https://arxiv.org/abs/2109.12098

[5] Radford, A., et al. (2021, January 5). CLIP: Connecting Text and Images. OpenAI. https://openai.com/blog/clip/

[6] sharadarao1999. (2020, October 6). Cosine Similarity. GeeksforGeeks. https://www.geeksforgeeks.org/cosine-similarity/

[7] Karabibber F. (n.d.). Cosine Similarity. LearnDataSci. Retrieved January 2, 2023, from https://www.learndatasci.com/glossary/cosine-similarity/