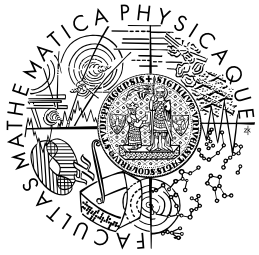

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Jakub Hajko

Comparison of joint-embedding deep networks for known-item search in image datasets

Department of Software Engineering

Supervisor of the bachelor thesis: doc. RNDr. Jakub Lokoč, Ph.D.

Study programme: Computer Science
Study branch: Artificial Intelligence

Prague 2024

I declare that I carried out this bachelor thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to express my gratitude to my supervisor, doc. RNDr. Jakub Lokoč, Ph.D., for suggesting this topic and for his help and guidance throughout this project. I would also like to thank Mgr. Ladislav Peška, Ph.D., and Patrik Veselý for their willingness and assistance. Last but not least, I want to thank my friends for their support and feedback.

Title: Comparison of joint-embedding deep networks for known-item search in image datasets

Author: Jakub Hajko

Department: Department of Software Engineering

Supervisor: doc. RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: As the production of multimedia continues to grow, the demand for effective multimedia retrieval methods increases. One critical task in this domain is known-item search within large unstructured collections of images using text queries. In recent years, this field has been dominated by deep networks trained to map both images and text in joint-embedding space. We evaluated multiple pre-trained networks, including CLIP, OpenCLIPs, ALIGN, and BLIP2, comparing their performance across various datasets. Additionally, we investigated how the amount of information provided within the text queries influences model performance. We also assessed the consistency of models' perceived image-image similarity with human judgments. Our findings indicate that OpenCLIP models excel in known-item search with queries and align well with human perception of similarity. Furthermore, we observed that providing more detailed information in text queries enhances model performance.

Keywords: multimedia, known-item search, joint-embedding deep networks

Contents

Introduction

Multimedia has become an essential part of our daily lives, seamlessly integrating into how we communicate, learn, and entertain ourselves. The amount of multimedia data continues to grow exponentially, and this trend shows no sign of slowing down. As the volume of multimedia data grows, so does the complexity of managing it. One very common problem is the ability to effectively retrieve relevant items from huge, often unstructured collections of images or videos based on the content of the searched item. We will focus mainly on retrieving images with text queries.

In 2021, OpenAI introduced the CLIP model[1], which revolutionized content-based known-item search within image collections. This model extracts features from both images and text and aligns them into a joint-embedding space. As a result, retrieval is enabled by computing the similarity between query and database features. Since then, several new models have been introduced, some of which extend and build upon the CLIP architecture, while others present completely new ideas.

We have chosen 9 pre-trained models and evaluated their performance across three different datasets. Each dataset presents a unique challenge: MVK (Marine Video Kit)[2] contains a collection of images from underwater videos, LSC[3] contains real-life images from lifelogging, and the Private Photos dataset features a gallery of phone photos from a real person. MVK and LSC datasets are utilized in the international retrieval competitions VBS and LSC. The findings from our comparison could potentially enhance retrieval systems used in these competitions.

We also conducted experiments to determine if providing more information in text queries is helpful for certain models and datasets. Additionally, we expanded on the findings from the RESET paper[4] to assess how closely models' understanding of image similarity aligns with human perception of similarity. This thesis is divided into four chapters: Basic Terms, Crossmodal Models, Implementation and Codebase, and Experiments. The first two chapters serve as an introduction to the findings upon which we build. The main contribution of my work is presented in the chapters Experiments and Implementation and Codebase.

1 Basic Terms

In this chapter, we will first introduce multimedia retrieval and related topics relevant to our work, as well as techniques for its implementation. Following this, we will introduce the two experiments conducted, along with their motivation and evaluation process. Ideas and knowledge introduced in 1.1 and 1.2 come from my supervisor's subject: Video Retrieval.

1.1 Multimedia Retrieval

Multimedia Retrieval is a field of computer science and information science focused on the process of obtaining relevant multimedia content—such as images, videos, audio, and text—from large, often unstructured datasets. The primary goal of Multimedia Retrieval Systems is to efficiently locate content that meets the user's needs based on queries, which can be expressed in natural language, example inputs, or structured formats. These systems employ various algorithms and techniques to index, search, rank, or retrieve the user-searched data. Common applications of Multimedia Retrieval include digital libraries, e-commerce platforms, social media, and video surveillance, all of which access vast amounts of multimedia content.

1.1.1 Known-Item Search in Image Datasets

In this work, we mainly focused on a subdomain of Multimedia Retrieval known as known-item search within image collections. Known-item search is a retrieval task where the user seeks a specific item they already know exists within the collection. This contrasts with ad hoc search, where users search for items based on general or broad queries without a specific item in mind. Known-item search is more precise and typically involves more specific queries tailored to locate a precise item or set of items.

The pursuit of known-item search within image datasets is motivated by several key factors. First, the exponential growth of multimedia production has led to vast collections of images, even in personal galleries, often containing tens of thousands of items. Efficiently locating specific images within these potentially unstructured collections can be incredibly time-consuming if done manually. This challenge becomes even more complex with large video collections, as videos are essentially sequences of images. Searching for specific scenes within hundreds of hours of video content highlights the importance of effective known-item search techniques. Moreover, the significance of known-item search is underscored by its prominence in international competitions like VBS and LSC [5, 6, 7, 8, 9, 10, 11, 12, 13].

1.1.2 Content-based Image Retrieval

We have introduced the concept of known-item search, but have not yet discussed the methodologies for its implementation. To retrieve an item from a database, the first consideration is the form of the query. The type of query is task-dependent; however, in general, queries can be based on metadata or the

content of the images. Metadata-based queries, although straightforward and achievable through simple metadata or attribute filters, are not the focus of this discussion. Instead, we will concentrate on content-based queries, which are more complex and interesting. Content-based image retrieval involves queries related to the actual content of the images within the collection. These queries can take various forms, including:

- **Text Query**: A textual description of the image content.
- **Query by Sketch**: A hand-drawn sketch approximating the desired image.
- **Query by Example**: An example image provided as the query.

The essential requirement for content-based queries is the ability to measure the similarity between the query and the data (images). This measurement allows the retrieval system to return the most similar items to the query. To generalize this process, we define the *Similarity Search Model*[14].

The Similarity Search Model involves:

- **Feature Extraction**: Identifying and extracting relevant features from both the query and the database images. Features are usually represented as high-dimensional vectors. Recently, deep neural networks have been the leading feature extraction methods [7, 8, 9, 10, 12, 13].

$$f_e : \mathrm{DB} \to \mathbb{R}^n$$

- **Similarity Measurement**: Applied on extracted feature vectors. Usually, a distance function or similarity function is used. A very common method is cosine similarity.

$$\delta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$$

- **Ranking and Retrieval**: Ranking the images based on their similarity scores and retrieving the most similar images to the query.

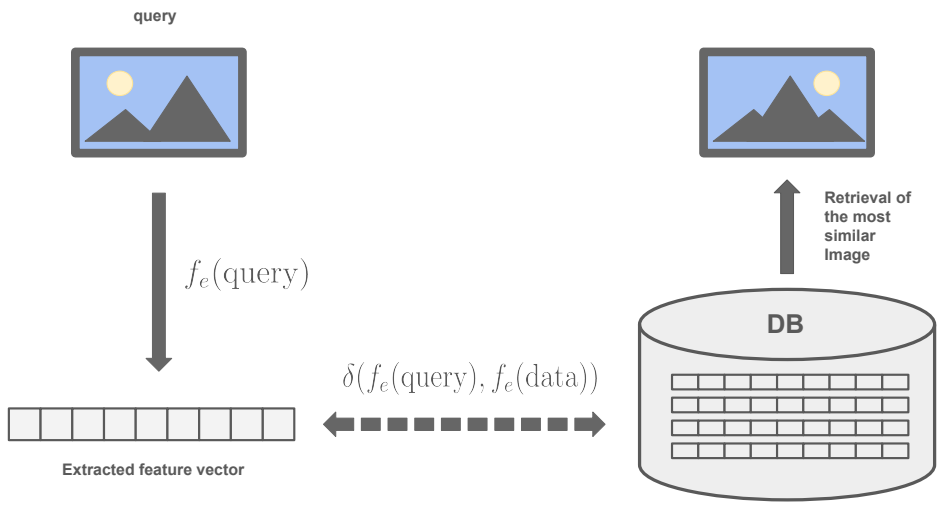

Figure 1.1 Sketch of the Similarity Search Model.

1.2 Known-Item Search with Text Query

In this experiment, our goal was to compare the performance of selected deep joint-embedding networks for known-item search within an image collection using text queries. To do this, evaluation datasets and text-image pairs for randomly sampled subsets of the datasets are needed. Both will be explained in more detail in the subchapter 4.1.

1.2.1 Evaluation Process

1. **Feature Extraction:** Extract feature vectors for the entire dataset and for the text queries using the same pre-trained model.

2. **Similarity Computation:** For each query, compute the cosine similarity between the query's feature vector and each image's feature vector in the dataset.

3. **Ranking:** Sort the dataset items based on their similarity scores, from most to least similar to the query. Determine the rank of the correct image in this ordered list. The rank is the position of the corresponding image in the sorted sequence.

4. **Rank Aggregation:** Repeat the above steps for all text-image pairs in the experiment. Record the rank of the correct image for each query.

1.2.2 Evaluation Methods

To compare the performance of the models, we will use two main methods:

1. **Cumulative Graphs:**

    - These graphs plot the proportion of queries for which the correct image is found within the top N ranks.
    - The x-axis represents the rank (e.g., top 1, top 5, top 10), and the y-axis represents the cumulative percentage of correctly retrieved images.

2. **Scatter Plots:**

    - Scatter plots will be used to compare pairs of models.
    - Each point represents a query, with the x and y coordinates corresponding to the ranks assigned by two different models.
    - This helps visualize the relative performance of the models for each query.

By following this evaluation pipeline, we can comprehensively assess and compare the effectiveness of different deep joint-embedding networks in retrieving relevant images based on text queries.

1.3 Image-Image Similarity

The inspiration for this experiment came from the paper "RESET: Relational Similarity Extension for V3C1 Video Dataset" [4]. The authors suggest that the quality of the image-image similarity model, especially in the context of images from videos is essential for some applications [15, 16, 17]. They also propose that there might be a semantic gap between human and deep networks' understanding of video keyframe similarity. This difference could be due to humans focusing on a highly variable and contextually dependent set of visual clues, which might differ from the visual clues extracted by the deep network. There is some evidence to support this [18]. Consequently, they constructed the RESET dataset, which will be explained in more detail in the subchapter 4.1. The dataset contains (query, candidate1, candidate2) triplets that were human-annotated (showing which candidate is more similar to the query). This can be used as a ground truth for measuring the consistency of human and model understanding of similarity. In the paper, the authors compared multiple feature extraction methods and measured the agreement ratio between model and human judgments. For the feature extraction, they used various techniques ranging from simple color histograms to more complex methods such as versions of CLIP or other deep networks. In our experiment, we will build upon the testbed from their experiments and extend the results with our selected pre-trained deep neural networks.

1.3.1 Evaluation Process

1. **Feature Extraction:** Extract feature vectors for the entire dataset with the model.

2. **Similarity Computation:** For each triplet, compute the cosine distance between the target image and the candidates.

3. **Consistency Computation:** Compute the agreement ratio between the models' and humans' judgments.

2 Cross-Modal Models

In recent years, content-based known-item search in image collections has been dominated by joint-embedding deep neural networks [7, 8, 9, 10, 12, 13]. In this chapter, we will delve into four different cross-modal networks, examining their architectures and training processes in greater detail. The specific versions of these models, which were used in our experiments, will be specified later.

2.1 CLIP

CLIP (Contrastive Language-Image Pretraining)[1] developed by OpenAI is a groundbreaking neural network intended to comprehend and connect visual and textual information. Using a dual-encoder architecture, CLIP processes images and text separately through specialized encoders, which map these inputs in joint-embedding space. CLIP uses a vast dataset of image-caption pairs scraped from the internet and then learns to align textual and visual representation through a method called contrastive learning[19]. This innovative training approach enables CLIP to demonstrate strong performance in various downstream tasks such as zero-shot classification and cross-modal retrieval without the need for task-specific fine-tuning.

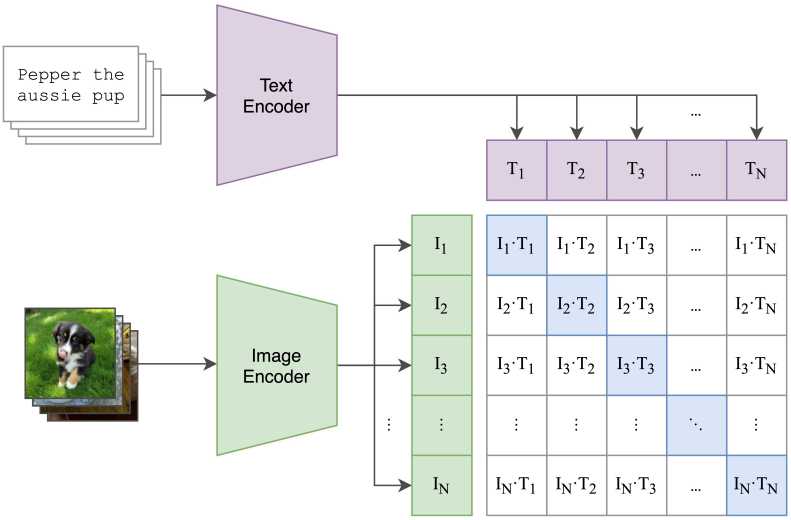

Figure 2.1 Architecture of CLIP model and process of contrastive learning on image-text pairs [1].

2.1.1 Architecture

CLIP employs a dual-encoder architecture consisting of an image encoder and text encoder. Image encoder is usually based on ResNet [20] or Vision Transformers (ViT) [21] architectures. These encoders are trained to extract visual information such as shapes, colors, and textures, forming high-dimensional feature vectors that capture the content of the image in a semantically meaningful

way. Text encoder is usually based on transformer-based models [22], such as BERT (Bidirectional Encoder Representations from Transformers) [23]. This encoder is trained to transform the input text into a high-dimensional feature vector space, which is shared with an output of the image encoder. The feature vector captures the semantical meaning of the text.

2.1.2 Contrastive Learning Mechanism

The core of CLIP's training paradigm is using a contrastive loss [19], which encourages the model to bring matching image-text pairs closer to each other in the joint-embedding space while pushing non-matching pairs further apart. The first batch of N image-text pairs is passed through corresponding encoders to form text and image embeddings. Then the contrastive loss is computed by considering the cosine similarity between each pair of image and text embeddings.

2.1.3 Training Process

The CLIP model was trained on a vast dataset consisting of 400 million image-text pairs collected from the internet. The encoders were trained from scratch, without any prior pretraining. The training process took place over 32 epochs, using large batches of image-text pairs to improve the model's learning efficiency.

2.2 OpenCLIP

The CLIP (Contrastive Language-Image Pre-Training) model, developed by OpenAI, was released in early 2021 [1]. Alongside the release of the original research paper, the codebase was made open source. Since then, major companies like Google and Apple, as well as online research communities, have explored pre-training their own versions of CLIP. These variations involve different datasets, image, and text encoders, training frameworks, and even slightly modified objective functions. This open-source initiative has been dubbed OpenCLIP. Covering all the variations would be beyond the scope of this work, so we have selected six OpenCLIP versions for our experiments:

- `ViT-SO400M-14-SigLIP-384`
- `ViT-L-16-SigLIP-384`
- `ViT-B-16-SigLIP-512`
- `DFN5B-CLIP-ViT-H-14-378`
- `ViT-g-14_laion2b_s34b_b88k`
- `ViT-H-14_laion2b_s32b_b79k`

2.2.1 OpenCLIP Versions Trained on WebLI Dataset

Name	Number of Parameters (in millions)	Shape of Output Embedding
ViT-SO400M-14-SigLIP-384	877.96	1152
ViT-L-16-SigLIP-384	652.48	1024
ViT-B-16-SigLIP-512	203.79	768

Table 2.1 Selected OpenCLIP versions trained on the WebLI dataset.

Three of the six models we selected were developed by Google's research team and trained on the WebLI dataset [24]. As the names suggest, these models utilize the innovative SigLIP objective function described in the "Sigmoid Loss for Language Image Pre-Training" paper [25], which also contains detailed descriptions of these models. Like CLIP, these models use a dual encoder architecture: the language encoder is transformer-based [22], and the image encoders are ViT-SO400M [26], ViT-L-16, and ViT-B-16. Google used the Big Vision codebase [27] for training, and all three models were subsequently converted to the PyTorch format. These publicly available pre-trained models were found and downloaded from Hugging Face [28, 29, 30].

WebLI (Web Language Image), described in the PaLI paper[24] is a multilingual image-text dataset developed by Google to support their vision-language research. This extensive dataset, built from publicly available images and text on the web, includes up to 10 billion images and 12 billion alt-texts, covering 109 languages.

2.2.2 OpenCLIP Version Trained on DFN5B Dataset

Name	Number of Parameters (in millions)	Shape of Output Embedding
DFN5B-CLIP-ViT-H-14-378	986.71	1024

Table 2.2 Selected OpenCLIP version trained on the DFN5B dataset.

This model was developed as a byproduct of the paper "Data Filtering Networks" by Apple and the University of Washington [31]. In this paper, Data Filtering Networks are introduced as small networks designed to automatically filter large pools of uncurated data. The best-performing dataset, DFN-5B, consisting of 5 billion images filtered from a pool of 43 billion uncurated image-text pairs, was selected and used for pre-training the OpenCLIP model with the ViT-H-14 Vision Transformer [21]. The AXLearn [32] code framework was used for training, and the model was subsequently converted to the PyTorch format. This publicly available pre-trained model was found and downloaded from Hugging Face [33].

2.2.3 OpenCLIP Versions Trained on LAION-2B Dataset

Name	Number of Parameters (in millions)	Shape of Output Embedding
ViT-g-14_laion2b_s34b_b88k	1366.68	1024
ViT-H-14_laion2b_s32b_b79k	986.11	1024

Table 2.3 Selected OpenCLIP versions trained on the LAION-2B dataset.

The remaining two models were trained on LAION-2B which is an English subset of the LAION-5B dataset [34, 35]. LAION-5B is a massive, publicly available dataset comprising 5.85 billion image-text pairs and is designed to support large-scale research in machine learning. Both models were introduced in the paper "Reproducible Scaling Laws for Contrastive Language-Image Learning," [36] which provides a detailed explanation of the training process. These publicly available pre-trained models were sourced from Hugging Face [37, 38].

2.3 ALIGN

ALIGN (A Large-scale Image and Noisy-text) [39] is a model developed by Google Research designed and pre-trained to align visual and textual representations within high-dimensional joint-embedding space, enabling it to be used for cross-modal retrieval. Similarly to the CLIP model [1] it leverages the dual encoder architecture with contrastive loss [19]. The key difference lies in the fact that ALIGN achieves this through large-scale pre-training on noisy web data, which were not preprocessed or filtered.

2.3.1 Architecture

The ALIGN model consists of two main components: an image encoder and a text encoder. The image encoder, typically a convolutional neural network (CNN) like EfficientNet [40], processes input images and generates fixed-size image embeddings. On the other hand, the text encoder uses a transformer-based architecture, similar to BERT [23, 22], to process input text and produce text embeddings.

2.3.2 Training Process

Both the image and text encoders are trained simultaneously using a contrastive learning objective [19]. This objective encourages the model to bring related image-text pairs closer together in the joint embedding space while pushing unrelated pairs apart. The key innovation of ALIGN lies in the data used for pre-training. ALIGN was trained on a massive dataset of image-text pairs collected from the web. This dataset is inherently noisy, containing various levels of text quality and relevance to the corresponding images. Despite the noise, the sheer scale of the data enables ALIGN to learn robust representations.

Figure 2.2 Example image-text pairs from the original ALIGN paper [39].

2.4 BLIP2

BLIP-2 (Bootstrapping Language-Image Pre-training) [41] introduces an innovative vision-and-language pre-training approach that utilizes off-the-shelf frozen, pre-trained image encoders [21] and frozen large language models [42]. The gap between modalities is bridged with a lightweight Q-Former (Querying Transformer), effectively addressing the rising costs associated with vision-and-language pre-training. The model is capable of a variety of zero-shot vision-language tasks such as instruct image-to-text generation, visual question answering, image captioning, and even text-image retrieval.

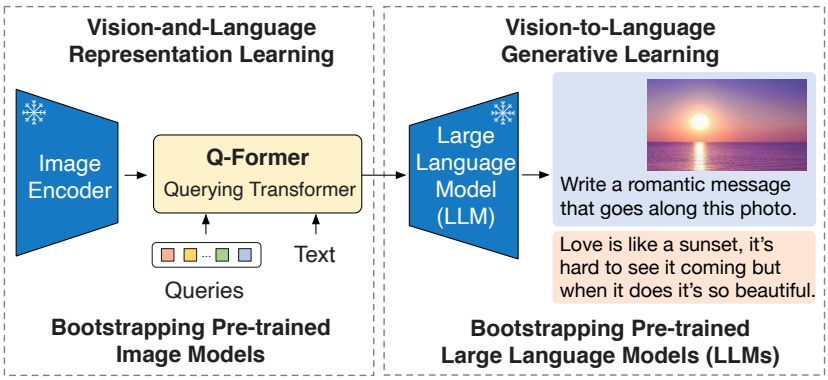

Figure 2.3 Overview of the BLIP2 Model from the original paper[41].

2.4.1 Architecture

The BLIP-2 model introduces a sophisticated architecture that utilizes frozen, pre-trained components to minimize computational overhead. The architecture consists of three main components: a frozen image encoder, a frozen language model (both acting like black boxes), and a lightweight querying transformer, which is the key innovation of this approach.

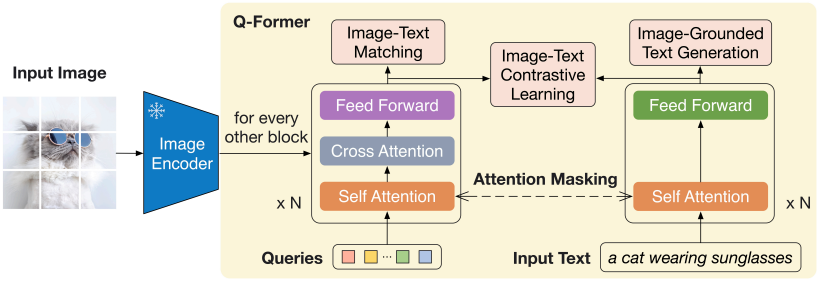

Figure 2.4 Architecture of the BLIP2 Model from the original paper[41].

Frozen Image Encoder: The image encoder is a pre-trained model, such as a Vision Transformer (ViT) [21], that processes input images into high-dimensional feature representations. By keeping this encoder frozen, the model makes use of the robust visual features learned from extensive pre-training.

Frozen Language Model: Similarly, the language model is a pre-trained transformer[22], such as GPT[42] or BERT [23], which remains frozen during the integration process. This model is responsible for generating or understanding text for downstream tasks such as visual question answering or image captioning. It is not needed for retrieval tasks, as there is no need for text generation, but nonetheless, it is a part of the architecture.

Q-Former: The Q-Former architecture consists of two transformer[22] sub-modules that share self-attention layers. The first sub-module, an image transformer, interacts with the frozen image encoder to extract visual features. The second sub-module serves as both an encoder and a decoder for text, functioning as a text transformer. A fixed number of learnable query embeddings are used as input to the image transformer. These queries interact with each other through self-attention layers and engage with frozen image features via cross-attention layers, which are inserted every other transformer block. Additionally, the queries can also interact with text through the same self-attention layers. Different self-attention masks are applied at the self-attention layer to manage query-text interaction, depending on the pre-training task. Q-Former is initialized with the pre-trained weights of BERTbase, while the cross-attention layers are randomly initialized. The model comprises 188M parameters, with the queries themselves considered part of the model parameters.

2.4.2 Training Process

Pre-training process of the BLIP-2 model is divided into two stages. The first is called Vision-Language Representation Learning where three pre-training objectives are optimized and the second is Vision-to-Language Generative Learning.

Vision-Language Representation Learning

In this stage, we focus solely on the architecture involving the frozen image encoder and Q-Former. Inspired by BLIP [43], three objective functions are optimized: Image-Text Contrastive Learning (ITC), Image-grounded Text Generation (ITG), and Image-Text Matching (ITM). The goal is to train Q-Former to ensure that its learnable queries extract visual representations highly informative of the text.

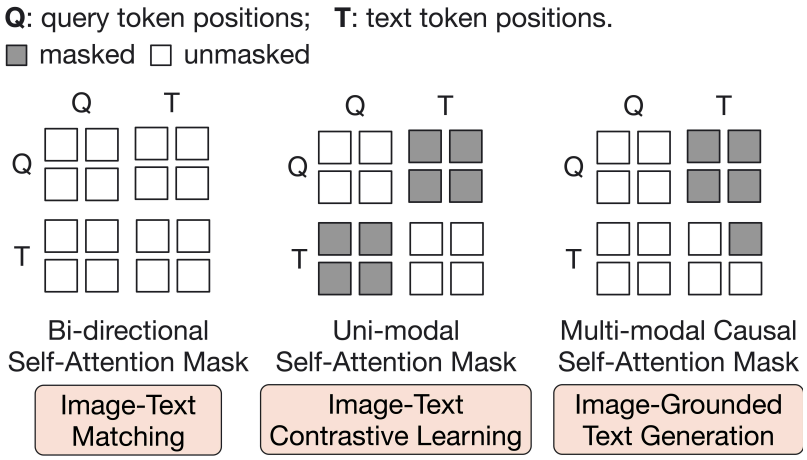

Figure 2.5 Image of self-attention masking strategy for each objective from the original paper [41].

- **Image-Text Contrastive Learning (ITC):** Aligns image and text representations by maximizing their mutual information through contrastive learning [19], comparing positive image-text pairs against negative ones. The highest similarity score between query outputs and the [CLS] token embedding is used.
- **Image-grounded Text Generation (ITG):** Trains Q-Former to generate text based on input images. Queries extract visual features and pass them to text tokens via self-attention layers, using a multimodal causal self-attention mask.
- **Image-Text Matching (ITM):** Learns fine-grained alignment between image and text through a binary classification task predicting whether image-text pairs are matched or not. It uses a bi-directional self-attention mask, with query embeddings capturing multimodal information.

Vision-to-Language Generative Learning

In this stage, the Large Language Model (LLM) is connected to the Q-Former. The fully connected layer is used to linearly project the output query embeddings into the same dimension as LLM's input language embeddings. In the previous stage, the Q-Former was pre-trained to extract visual representations that are

informative for language processing. It acts as an information bottleneck, delivering the most relevant details to the LLM and filtering out unnecessary visual information. The authors of the paper experimented with two types of LLMs: encoder-decoder-based LLMs and decoder-based LLMs. For encoder-decoder-based LLMs, prefix language modeling loss was used. The prefix of the text was combined with the visual information and inputted to the LLM, while the suffix was used as the generation target. For decoder-based LLMs, language modeling loss was utilized.

2.4.3 BLIP-2 Retrieval Tasks

When conducting a known-item search using a joint-embedding space, there is no need for text generation, so a frozen Large Language Model is unnecessary. To enhance the performance of BLIP-2, the authors of the paper suggest that the initial training stage can be utilized to further fine-tune the model. The pre-trained model we utilized was obtained from the LAVIS project [44], and for assistance with implementation, we referred to the article [45]. The model utilized is BLIP2, adapted for feature extraction with the ViT-L/14 image transformer.

3 Implementation and Codebase

3.1 Language, Libraries and Resources

The project is implemented using Python. Below are some of the most crucial libraries used in this project.

- **Pillow (10.3.0)**: Python Imaging Library for opening, manipulating, and saving image files.
- **Pandas (2.2.2)**: An essential library for data manipulation and analysis.
- **NumPy (1.26.4)**: Fundamental package for numerical computations with support for arrays and matrices.
- **Matplotlib (3.8.4)**: Comprehensive library for creating static, animated, and interactive visualizations in Python.
- **Torch (2.3.0)**: Also known as PyTorch, it is a deep learning framework that provides a flexible and efficient platform for building neural network models.
- **Transformers (4.41.0.dev0)**: A library by Hugging Face that provides pre-trained models and tools to implement state-of-the-art natural language processing (NLP) and computer vision tasks.
- **Timeit** : A simple library to measure the execution time of small code snippets, useful for performance testing.
- **Tkinter**: Standard Python interface to the Tk GUI toolkit, used for building graphical user interfaces.

For the computation of image feature extraction and retrieval evaluation, I used the computational resources provided by the Czech National Grid Infrastructure, MetaCentrum, and Google Colab.

3.2 Codebase

We have organized our code in a project called M3RT (Multi-Modal Models Retrieval Tasks). The source code is conceptually divided into four parts: ***retrievers***, ***utils***, ***scripts***, and ***notebooks***. The ***datasets*** directory is where our datasets and labels are situated, while the ***saves*** directory is the place for saving all experiment results, plots, and image encodings. The structure is as follows:

```
M3RT
├── src
│   ├── retrievers
│   ├── scripts
│   └── utils
│       ├── labelling_app
│       └── plot
├── notebooks
├── datasets
│   ├── photos
│   ├── marine
│   └── lsc
└── saves
    ├── image_features
    ├── cumulations
    ├── ranks
    ├── plots
    └── reset_csvs
```

3.2.1 Directory 'retrievers'

Files:

- **`retriever.py`**
- **`clip_retriever.py`**
- **`openclip_retriever.py`**
- **`align_retriever.py`**
- **`blip2_retriever.py`**

The Retriever class, which is defined in **`retriever.py`**, encompasses the entire functionality of the retriever system used in our experiments. It serves as the superclass to all other specialized retriever subclasses. The class allows for loading image features from a specified path, computing and saving cumulation, and ranks for building plots and handling runtime messages printed to the terminal.

Since each model used in our retriever system may have a different interface, we have structured our classes accordingly. The subclasses ALIGNRetriever, BLIP2Retriever, OpenCLIPRetriever, and CLIPRetriever, extend the Retriever class and implement their own constructors. These constructors load the pre-trained model and define functions `encode_images()` and `encode_text()`. The

`encode_images()` function allows us to save the created image embeddings in a specified output path. The process of image encoding is handled in batches to prevent memory overflow. In addition, the OpenCLIPRetriever allows us to select the version of the pre-trained model we use. The computation also makes use of CUDA, allowing for processing on GPUs. This structure enables us to separate implementations of the retriever system for different models, keeping it clean and modular for potential future extensions.

3.2.2 Directory 'scripts'

Files:

- **`encode_images.py`**
- **`compute_cumulations.py`**
- **`compute_ranks.py`**
- **`extend_reset_csv.py`**

This directory contains scripts for higher-level tasks such as creating image encodings and computing cumulations and ranks. Each script accepts arguments where you can specify the dataset, type of labels, model, and version of the model if it is openclip. They instantiate the retriever system with the given parameters and save the results to the ***saves*** directory with appropriate filenames. The scripts also measure and report the time taken for encoding and computing tasks, providing a clear interface even for remote execution. The script **`extend_reset_csv.py`** is slightly different from the rest. It is used to compute distances for all triplets and extend the CSV file, which is then used in the RESET paper's codebase[4].

3.2.3 Directory 'notebooks'

Files:

- **`baseline_CLIPvsALIGN.ipynb`**
- **`overallJudgementConsistency.ipynb`**

This directory contains Jupyter notebooks. The **`baseline_CLIPvsALIGN.ipynb`** was our first implementation, from which all other code evolved. The **`overallJudgementConsist`** file is part of the RESET paper[4] codebase, but we adapted it to extend it with our models.

3.2.4 Directory 'utils'

Subdirectories:

- ***labelling_app***
- ***plot***

The ***labelling_app*** subdirectory contains a simple GUI application built with Tkinter for easing the dataset labeling process. It consists of:

- **`LabellingApp.py`**: Implements the GUI, which displays an image and provides two text boxes for entering short and long text labels. It includes `"Save"` and `"Sample Again"` buttons. The `"Save"` button appends the labels to a CSV file and displays a new random image, while the `"Sample Again"` button just displays a new random image without saving.
- **`LabellingEngine.py`**: Contains the class responsible for selecting random images from a specified dataset and writing the labels to a CSV file.

The ***plot*** directory contains all the code used to create our plots.

- **`plot_cumulative.py`**: Contains `plot_cumulative_graph()` function, which plots the cumulative graph for specified models.
- **`scatter_plots.py`**: It allows us to add computed ranks for models and then it creates scatter plots for all combinations of models.
- **`complementary_cumulation.py`**: It allows us to add computed cumulations and then it creates plots for all combinations of models.
- **`plot_grid.py`**: Takes scatter plots and cumulative graphs as input, it also needs mask images for diagonal and it prints them in the grid.

3.2.5 Directory 'saves'

This directory is designated for storing the results of our computations and plots. We have established naming conventions for the files to maintain a clean and structured organization. The subdirectories ***image_features***, ***cumulations***, and ***ranks*** contain pickle files with stored results. File names are structured as follows: first, the model name is mentioned, followed by the model version if the model is openclip. Then, the dataset name is included, and finally, if it is a ranks or cumulation file, the type of labels is also added. The subdirectories ***plots*** and ***reset_csvs*** contain plots, usually in the form of PNG files, and CSV files used in the RESET codebase[4].

4 Experiments

In this chapter, we will start by introducing the datasets and text labels used to evaluate our experiments. After that, we will present the results of our research. Our experiments included *Known-Item Search with Text Query* and *Image-image Similarity*, which are discussed in detail in 1.2 and 1.3. We tested and compared nine different joint-embedding networks during the evaluation:

1. **CLIP ViT-B/32**
2. **BLIP-2** (with the ViT-L/14 image transformer)
3. **ALIGN**
4. **OpenCLIP ViT-SO400M-14-SigLIP-384**
5. **OpenCLIP ViT-L-16-SigLIP-384**
6. **OpenCLIP ViT-B-16-SigLIP-512**
7. **OpenCLIP DFN5B-CLIP-ViT-H-14-378**
8. **OpenCLIP ViT-g-14_laion2b_s34b_b88k**
9. **OpenCLIP ViT-H-14_laion2b_s32b_b79k**

4.1 Testbed

For our experiments total of 4 datasets were used. Three of these—namely the Private Photos dataset, MVK (Marine Video Kit)[2], and LSC'24 (Lifelog Search Challenge)[3] dataset were used within our testbed for the *Known-Item Search with Text Query* experiment. For the *Image-image Similarity* experiment, we employed the testbed from the paper 'RESET: Relational Similarity Extension for V3C1 Video Dataset' [4], which included a human-annotated RESET dataset consisting of more than 17000 (query, cnadidate1, candidate2) triplets. Each of these datasets was selected for its unique characteristics and relevance to our research goals. In the following section, we will delve into a detailed examination of these datasets, as well as the text labels that were used to measure performance in our experiments.

Below are brief explanations of the datasets we used:

1. **Private Photos Dataset:** Collection of photos from private phone gallery.
2. **MVK (Marine Video Kit)[2]:** Collection of videos(converted to images) from underwater environments, capturing variety of sea flora and fauna.
3. **LSC'24 (Lifelog Search Challenge)[3]:** Collection of real life images generated by one active lifelogger.

4. **RESET (RElational Similarity Evaluation dataseT)[4]:** Collection of over 17,000 similarity annotations for query-candidate-candidate triples of video keyframes taken from the V3C1 video collection which is a subset of V3C(Vimeo Creative Commons Collection)[46].

4.1.1 Text Labels For Images

To measure the performance of *Known-Item Search with Text Query* experiment, text-image pairs are needed. Since external, publicly available text annotations for our datasets are somewhat limited, I custom-created the text labels for all three datasets. This allows us to simulate real-life user text queries for retrieval from image datasets. Although designing text labels from a single user is suboptimal, it should still provide a reasonable approximation of real-life users. Additionally, it allows us to use the testbed to extend the text label set for other users in the future. Custom creation of text labels also gives us an option to experiment with the amount of information provided within the text label. For each dataset, we created two types of text labels:

- **Short text labels:** Image is described with a minimal amount of words, usually up to one short sentence.
- **Long text labels:** Image is described with more than one sentence usually describing the content of the image in more detail, for example specifying the position, color, and shape of objects within the Image.

Short label:
Sunset above Prague.

Long label:
Orange sunset above Prague castle, river canal in the front.

Short label:
Scuba diver on a blue background.

Long label:
Scuba diver with a grey gas tank with bubbles above him. He is in left bottom corner and his feet are not visible. On the background there is blue color.

Short label:
Taking a picture of a lake.

Long label:
Hands holding a phone taking a picture of a lake. There is a hill behind the lake.

Figure 4.1 Examples of short and long text labels for images sampled from each of our datasets: Private Photos Dataset (left), MVK (middle), and LSC'24 Dataset (right).

Note that all three datasets are different and require different types of vocabulary during the labeling process. While the Private Photos Dataset and LSC'24 primarily capture real-life situations, the MVK dataset is highly domain-specific,

featuring a wide variety of underwater recordings. To those without prior knowledge of this specialized terminology, these recordings might appear quite similar. To simulate real-life users during the labeling process of the MVK dataset, we avoided using domain-specific terminology to describe the images. For example, instead of specifying the exact species of fish, we described its appearance. However, this approach resulted in longer text labels for the MVK dataset compared to the Private Photos Dataset and LSC'24.

During the labeling process, we randomly sampled 100 images from each of our three datasets. In the MVK dataset, some images were not suitable for the retrieval task and likely wouldn't be targeted for retrieval, such as very blurred images or those showing nothing but blue color. In these instances, we randomly selected another image from the dataset and continued with the labeling process. For each image, both a short and a long text label were created, resulting in a total of 600 text labels.

4.1.2 Private Photos Dataset

- **Name**: Private Photos Dataset
- **Source**: Custom-created from a personal collection of images
- **Date Range**: October 2022 to July 2023
- **Total Images**: 534
- **Format**: JPEG

Composition and Origin The Private Photos Dataset was developed as a lightweight baseline for cross-modal model evaluation. Most images were captured using an iPhone 13 Pro Max, but the dataset also contains screenshots and downloaded images. The collection provides a diverse visual representation of student life in Prague, with additional photos from various locations.

Content Description The dataset captures an image gallery of a student residing in Prague. It includes everyday activities and objects, social gatherings, cityscapes, nature, and miscellaneous personal moments.

Data Preparation

- **Selection**: Images were handpicked to ensure a representative sample of daily activities and diverse scenes.
- **Quality Adjustment**: Image quality was deliberately reduced to optimize dataset size and allow faster processing.
- **Formatting**: All selected images were converted to JPEG format and numbered sequentially.

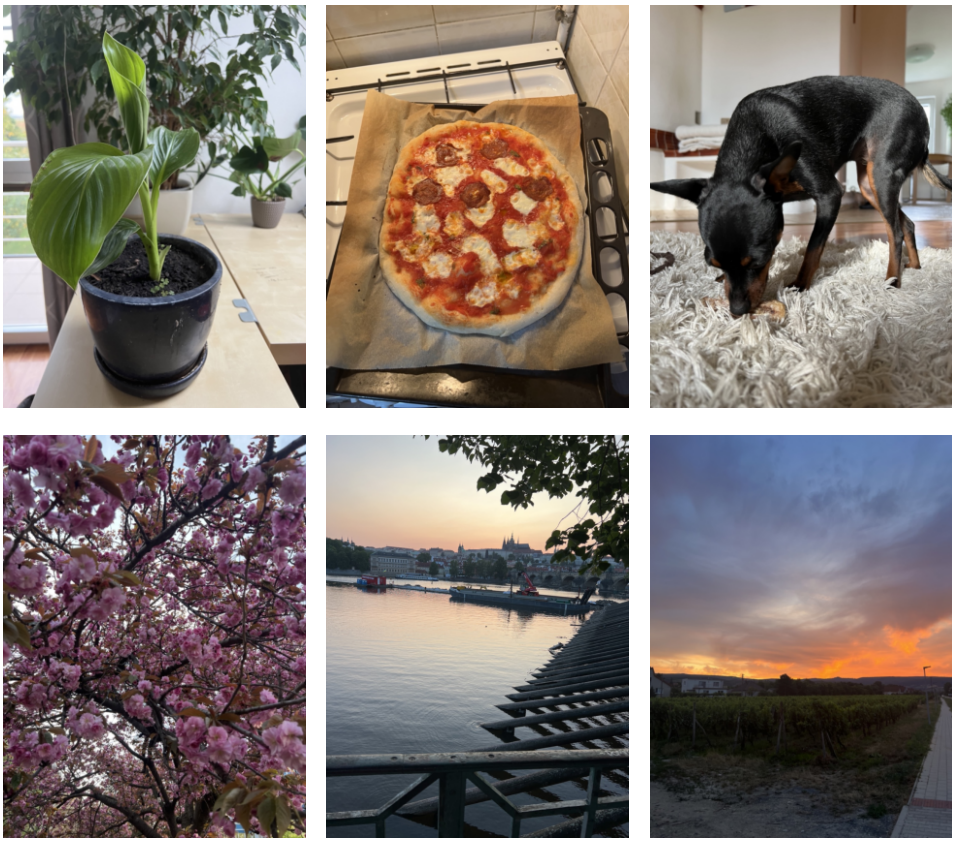

Figure 4.2 Sample of images from the Private Photos Dataset.

4.1.3 MVK(Marine Video Kit)

Nontraditional Domain-specific datasets present an interesting challenge for state-of-the-art general-purpose models. That is why we selected the Marine Video Kit dataset[2] featuring a wide collection of videos of marine environments. This dataset was also utilized in an international known-item retrieval competition VBS[10].

Usage in our Experiments In our experiments, we used a subset of the full MVK dataset, which includes 9,413 images. This subset offers a sufficiently large sample for our purposes.

Composition and Origin The dataset is composed of single-shot videos taken by moving cameras in various underwater environments. These videos feature a wide variety of underwater flora and fauna, as well as diving gear. Since the videos are not post-processed, they vary in quality, with some being blurred or captured in low-light conditions. The full MVK collection includes 1,379 videos, ranging from 2 seconds to nearly 5 minutes. The average video length is 29.9 seconds, while the median is 25.4 seconds. The videos were recorded across 11 different regions from 2011 to 2022.

Figure 4.3 Sample of images from the MVK dataset.

4.1.4 LSC'24(Lifelog Search Challenge)

LSC (Lifelog Search Challenge)[3] is a workshop where teams compete to develop the leading Lifelog retrieval tool. The workshop has been held since 2018, with the most recent challenge taking place at the ACM IMCR'24 conference on June 10th, 2024 in Phuket, Thailand. The datasets in the LSC contain life log data captured over an extended period from a single individual, consisting of real-life shots from their experiences. For our experiments, we used a smaller subset from the LSC'24 dataset, which includes 9566 images captured during the first 10 days of January 2020, providing a significant dataset for our evaluations.

Figure 4.4 Sample of images from the LSC'24 dataset.

4.1.5 RESET(RElational Similarity Evaluation dataseT)

RESET(RElational Similarity Evaluation dataseT)[4] is a dataset consisting of 17026 human-annotated (query, candidate1, candidate2) triplets collected during the spring of 2023. The annotations indicate whether candidate1 image is more similar to the query image than candidate2 image, according to human judgment. A total of 84 participants were recruited to create this dataset. The participants were reasonably diverse in terms of age, education, and knowledge of machine learning. The images for the dataset were sourced from the publicly available V3C1 dataset[46]. As mentioned in section 1.3, this dataset can be utilized as a benchmark for assessing the consistency of human and model understanding of similarity in image collections.

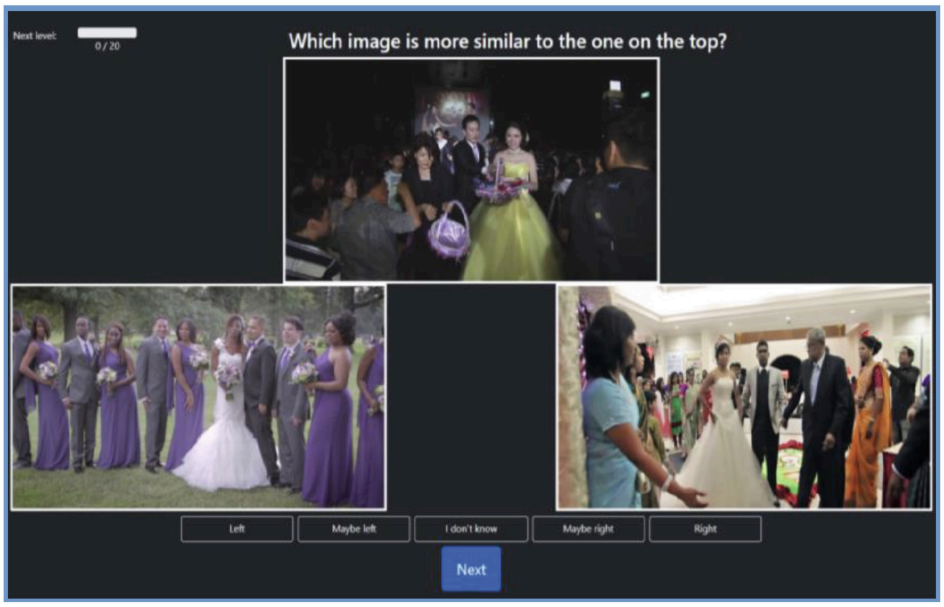

Figure 4.5 Screenshot illustrating the data collection process as described in the original paper [4].

4.2 Known-Item Search with Text Query

In this experiment, we will examine how well the selected models performed in known-item search with text queries. All nine selected models were evaluated under the same conditions on all three datasets and with both short and long text labels. First, in Figure 4.6, we will examine the cumulative graph results. Subsequently, in Figures 4.7 4.8 4.9 4.10 4.11 4.12, we will select the top 5 performing models for specific datasets and label types, and compare them in pairs. The results will be presented in a 5x5 grid of plots.

4.2.1 Cumulative Graphs Plot

The figure below shows six cumulative graphs featuring two sets of labels (short and long) for each of our three datasets. Each curve on the graph represents the performance of a specific model. In the cumulative graph, the x-axis indicates the rank, and the y-axis represents the cumulative percentage of retrieved images. This allows us to see what percentage of our text queries were retrieved within the top k rank. A steeper and higher line on the graph indicates better model performance. For clarity, each model was assigned a unique color during our experiments. Additionally, for the "Private Photos" dataset, we limit the x-axis to rank 50, as the dataset contains only 534 images. For the MVK and LSC datasets, each containing approximately 10000 images, we limit the x-axis to rank 500.

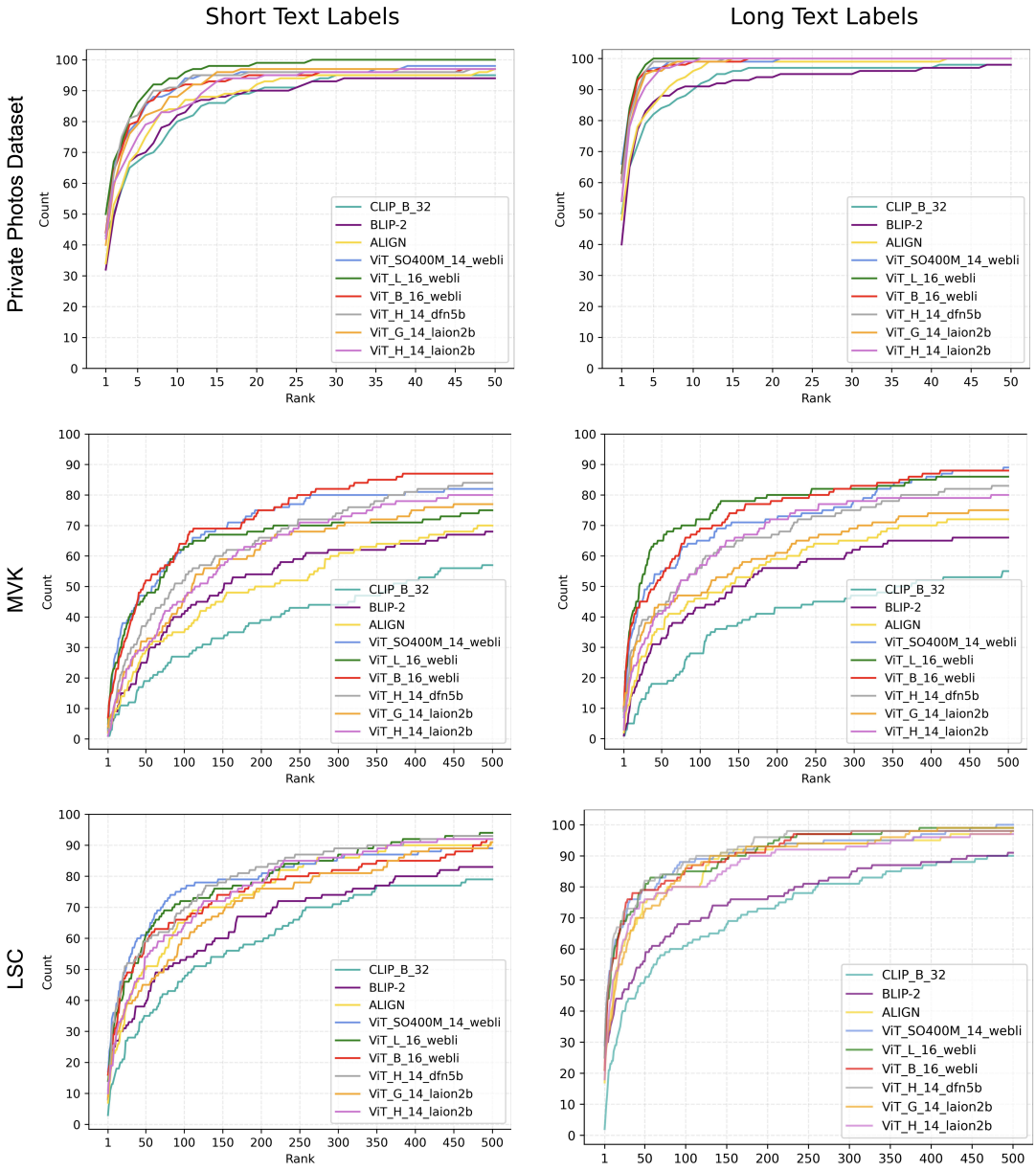

Figure 4.6 Cumulative Graphs

It is important to acknowledge that domain-specific datasets like MVK pose a slightly more difficult challenge for our models compared to less domain-specific datasets like LSC and the Private Photos Dataset, which capture more "real-life" situations. This is evident not only from observing the height and steepness of the curves but also from the fact that the best-performing model on LSC and short labels was able to retrieve over 60% of the queries within rank 50 and nearly 80% within rank 150. In contrast, for MVK, the retrieval rates were only slightly over 50% and almost 70% respectively. This trend was also evident for long labels, where the differences were slightly larger. It is also worth noting that the models performed better on the Private Photos Dataset compared to the other two datasets, as the best-performing model was able to retrieve 100% of the queries with a rank of 25 or better for both short and long labels. This is likely due to the significantly smaller size and complexity of the dataset.

An interesting observation is that long labels consistently perform better than short labels across all three datasets. Queries with longer text tend to result in much steeper curves. For the MVK dataset, the best performing model was able to retrieve nearly 70% of the test queries within the top 50 ranks for long labels, while for short labels, it was only about 50%. This trend was similar for the LSC dataset as well. Another notable point is that for long labels in the LSC and Private Photos Dataset, the differences in model performance appear to be smaller compared to short labels on the same datasets, wherein the model performances seem to vary more. However, this pattern does not seem to hold for the MVK dataset, where the differences between the models seem to persist even for long labels. This might be caused by the fact that if you provide sufficient information for most models on a simpler, less domain-specific dataset, they can retrieve better results.

After analyzing the model performances, it is evident that OpenCLIPs outperform other models across all six plots. Both base CLIP and BLIP2 models consistently underperform across all three datasets. ALIGN also struggles with the Private Photos Dataset and MVK dataset, however performs solid for LSC, especially for long labels. It's worth noting the significant lead of webli-trained OpenCLIPs over other models on the MVK dataset for both short and long labels. Webli-trained models also excel on the Private Photos Dataset, with ViT_L_16_webli performing best for both short and long labels, as well as ViT_SO400M_14_webli showing strong performance on the LSC dataset. Dfn5b trained model keeps up with webli-trained OpenCLIPs and even outperforms them in some cases, except for the MVK dataset where it seems to struggle. On the other hand, Laion2B-trained models deliver average performance across all three datasets.

4.2.2 Grid Plots

In the following plots, we wanted to test what pairs of models might have the potential for performing good when used in combination for known-item search with text query. An example utilization of such a pair would be using a more effective model for retrieval first, and if that doesn't work, switching to the second model.

For each dataset and text label type, we created a 5x5 grid plot featuring the 5 best-performing models. The upper triangle of the plot contains scatter plots for all combinations of models, while the lower triangle contains cumulative graphs for all combinations. The scatter plots have both the x and y axis in log

scale. Each point represents a text query and its position is determined by the ranks of model 1 and model 2. Points on the diagonal have the same rank for both models. A larger number of points on the diagonal may indicate a lower potential for using the models in combination. In the cumulative graphs, there

are three curves: one for each model and one named the complementary curve. This was created by taking the better of the two produced ranks for each text query. A larger gap between the complementary curve and the curve of the better performing model might indicate a better potential for combination. However, please note that if we use the method described before, we would not be able to achieve the performance showcased in the complementary curve because we don't know which model will perform better for each query.

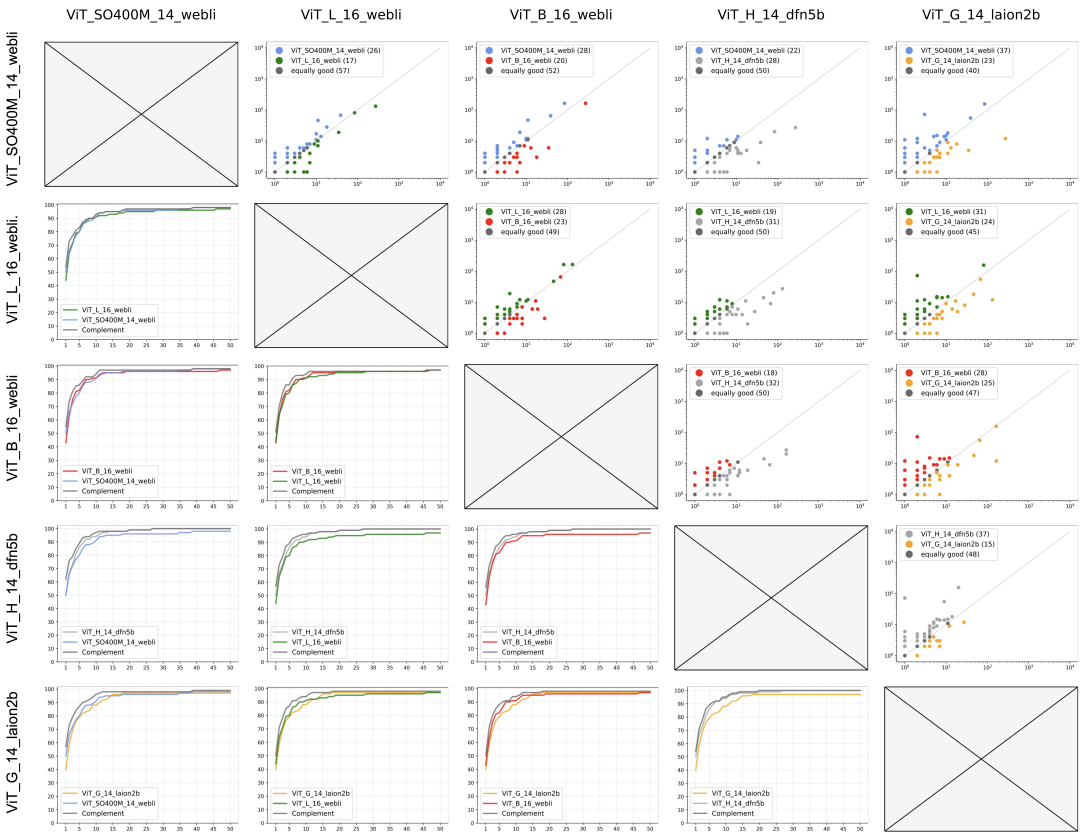

Figure 4.7 Private Photos Dataset with Short Text Labels

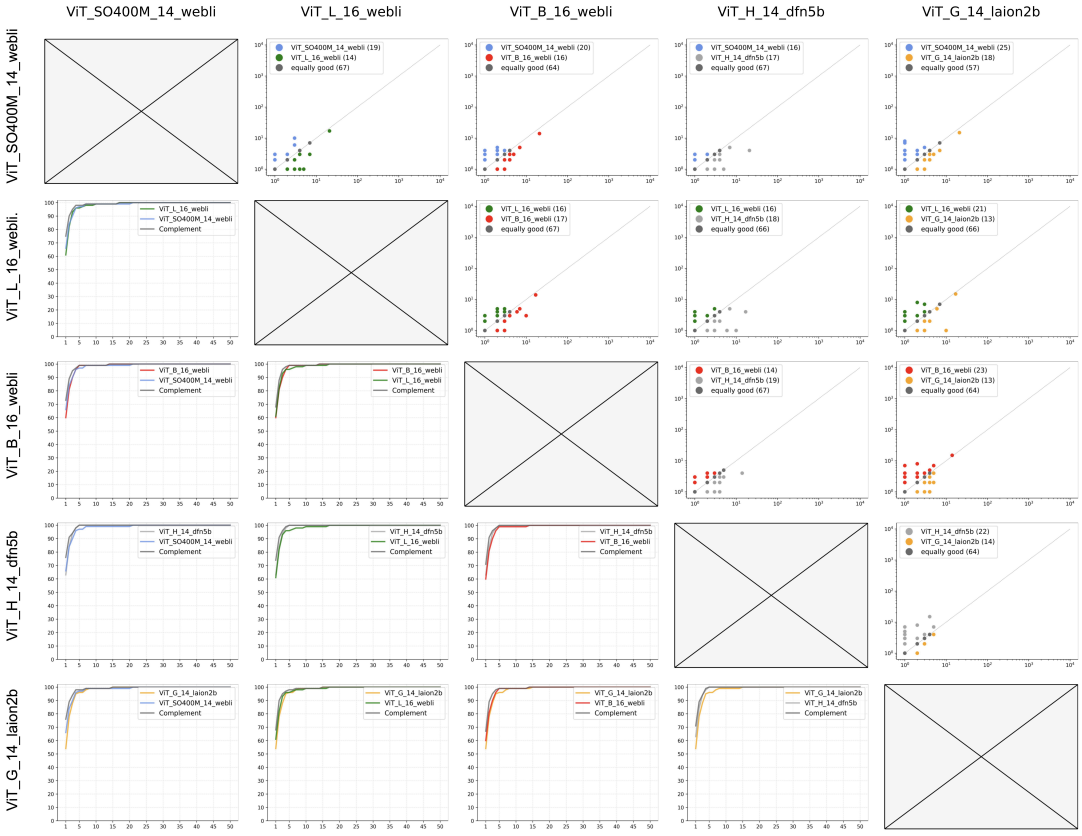

Figure 4.8 Private Photos Dataset with Long Text Labels

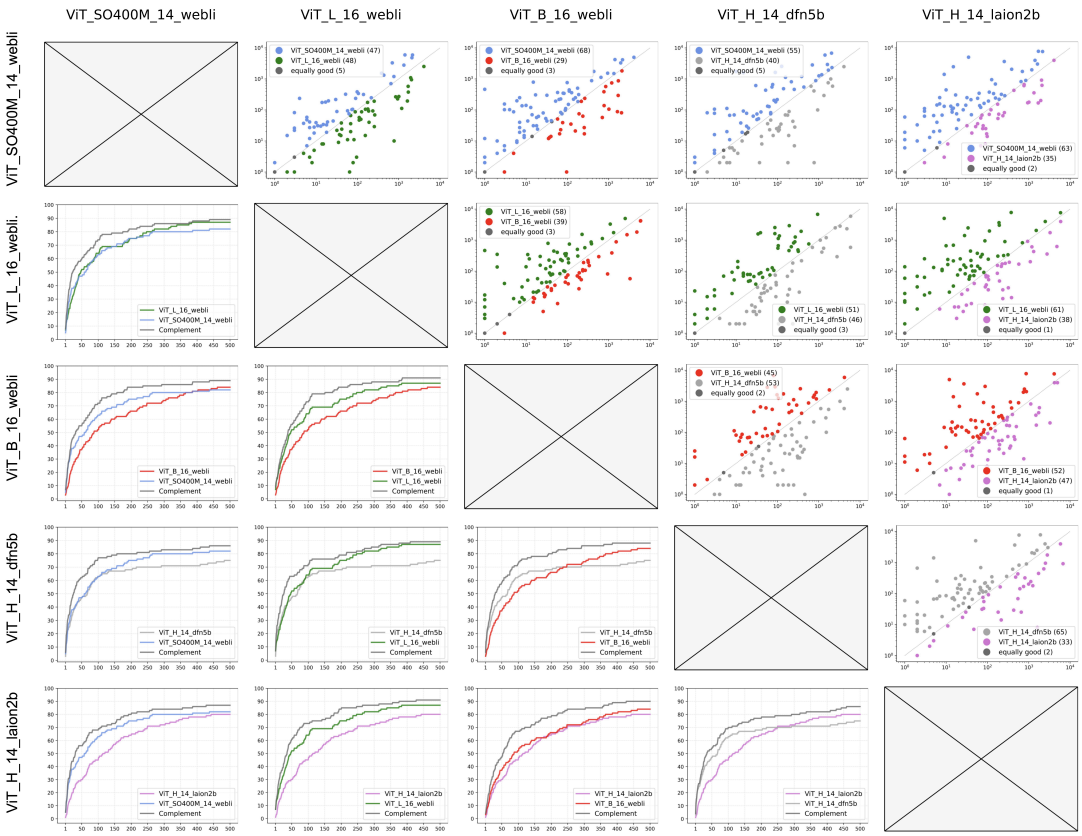

Figure 4.9 MVK Dataset with Short Text Labels

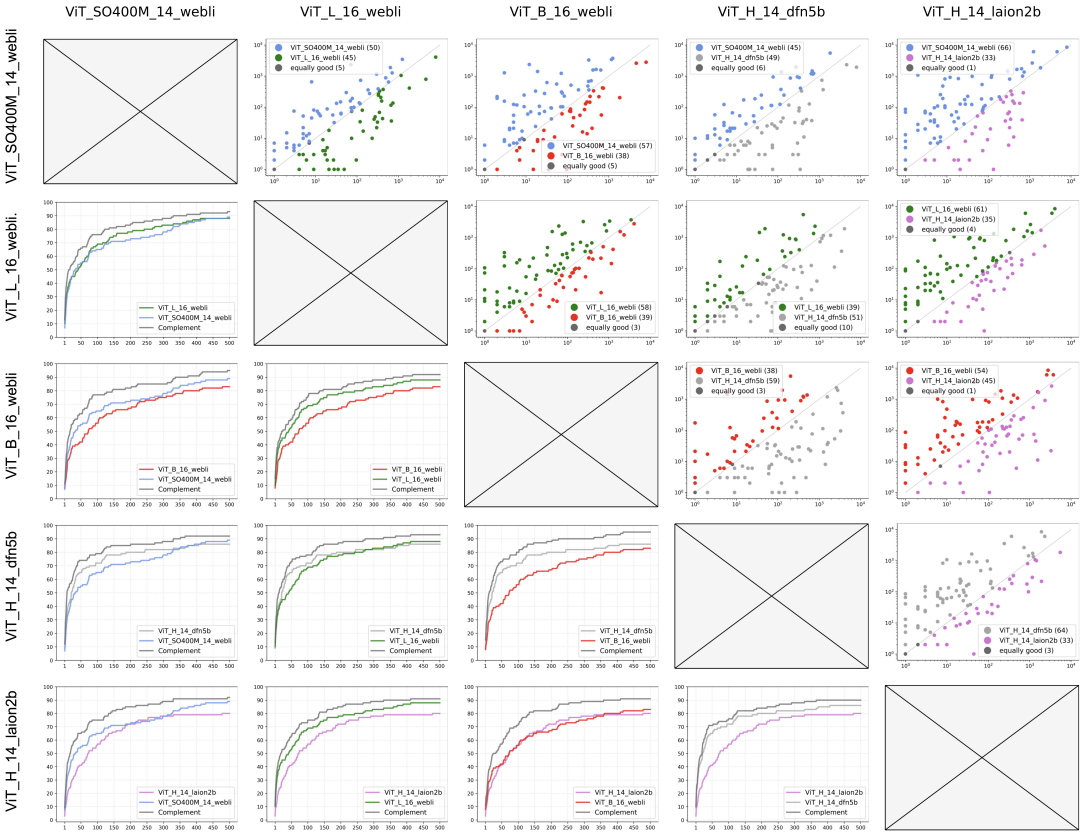

Figure 4.10 MVK Dataset with Long Text Labels

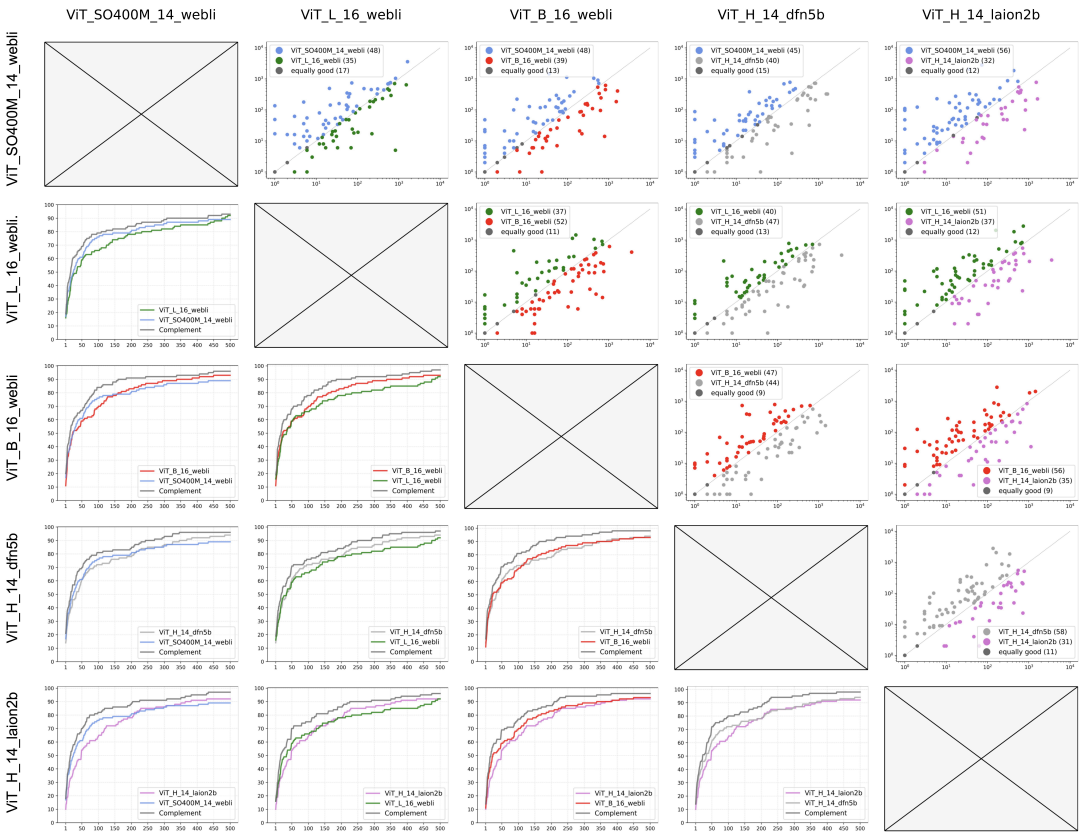

Figure 4.11 LSC Dataset with Short Text Labels

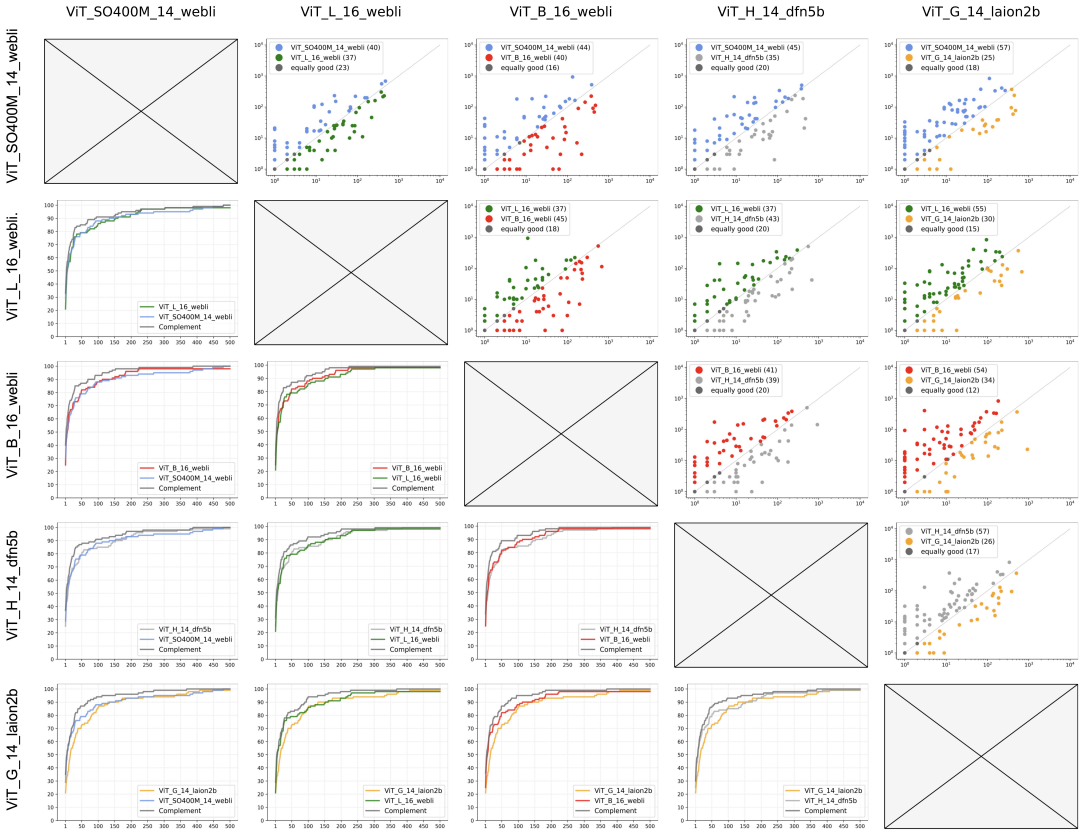

Figure 4.12 LSC Dataset with Long Text Labels

Upon reviewing the grid plots for the Private Photos Dataset, it appears that there is little potential for improving performance by combining two models. Since all the models performed well, there is no logical need to combine them. This is supported by the fact that most queries ended up on the diagonal for all scatter plots, and the curves are very similar to the complementary curve.

Conversely, the MVK dataset shows the best potential for improving the retrieval system's performance by combining two models. We observed a small number of queries ending up on the diagonal for scatter plots, and the complementary curve is better than the model curves. Combining ViT_H_14_dfn5b with all three webli-trained models seems like a promising combination for both short and long labels.

Furthermore, LSC also shows potential for improving the retrieval by combining two models. We can see that a relatively small number of queries ended up on the diagonal for scatter plots. This number increases for long text labels, but upon inspecting the cumulative graphs, the complementary curves still perform substantially better, even for long labels. The combination of ViT_H_14_dfn5b with webli-trained models, as well as with ViT_G_14_laion2b, seems like a promising combination for long text labels. For short text labels, combinations of models containing ViT_H_14_laion2b and ViT_H_14_dfn5b seem promising.

4.3 Image-Image Similarity

In this experiment, we extended the results from the study conducted in the RESET paper using four different models. Instead of selecting the four best-performing models, we opted for models trained on different datasets. We chose 'clip' as a reference model, as the original paper already uses a version of CLIP. Additionally, we selected 'ViT_SO400M_14_webli', which performed well in previous experiments, 'ViT_H_14_laion2b', which was the best-performing model trained on the 'laion2b' dataset, and 'ViT_H_14_dfn5b', trained on the 'dfn5b' dataset which also performed very well in previous experiments. The original study measured the consistency of human and model judgments by determining how many of the triplets were consistent, divided by all triplets. A total of 30 models were tested, with the best model reaching a consistency ratio of over 0.76. The results of our extended experiment are plotted in a bar graph. The original feature extractors are represented by blue bars, while our four models are shown with red bars.

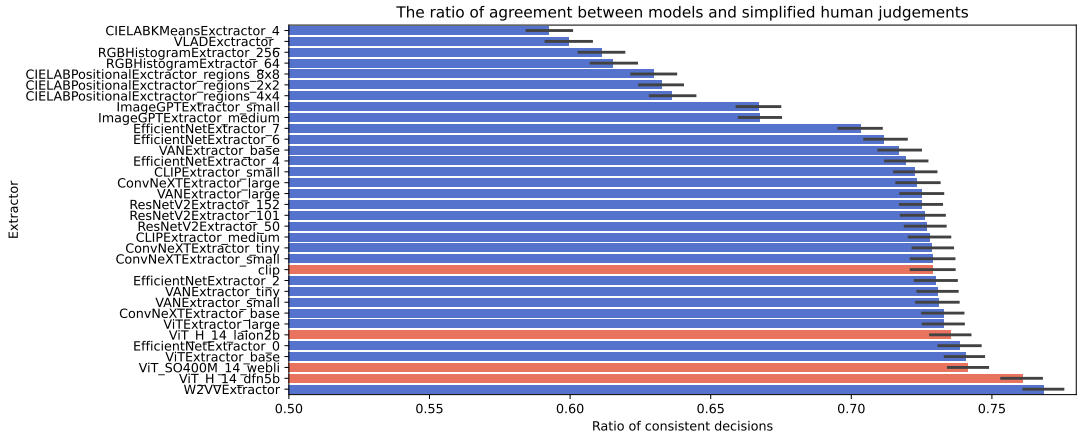

Figure 4.13 Plot showing the consistency of model predictions with human judgments.

We can see that all three OpenCLIPs performed very well, ranking in the top 6 among all 34 models. In particular, ViT_H_14_dfn5b achieved a very strong performance, coming very close to the best model W2VVExtractor. Our CLIP model shows very similar performance to CLIPExtractor_medium, although it's not the same version.

Conclusion

In this work, we conducted a comparative analysis of multiple pre-trained joint-embedding models on datasets with varying characteristics that posed different challenges for the models. We also investigated how the amount of information in the text query influences our models' performance. Additionally, we extended a RESET study to examine the models' consistency in understanding similarity with humans.

Our findings indicate that OpenCLIP models, particularly those trained on the webli dataset, performed the best. However, the OpenCLIP model trained on the dfn5b dataset also demonstrated strong performance. We also observed the potential for using a combination of models in retrieval systems, especially for the MVK dataset and the LSC dataset. Providing more information within text queries improved the performance of models on all tested datasets.

Furthermore, our selected OpenCLIP models used to expand the original study, showed good consistency with human judgments. They ranked in the top 6 out of 34 tested extractors, achieving consistency close to 75

In the future, this research can be extended by evaluating the models using a larger set of text queries generated by different individuals. Additionally, new models and more datasets can be incorporated.

Bibliography

1. Radford, Alec; Kim, Jong Wook; Hallacy, Chris; Ramesh, Aditya; Goh, Gabriel; Agarwal, Sandhini; Sastry, Girish; Askell, Amanda; Mishkin, Pamela; Clark, Jack; Krueger, Gretchen; Sutskever, Ilya. Learning Transferable Visual Models From Natural Language Supervision. 2021.

2. Truong, Quang-Trung; Vu, Tuan-Anh; Ha, Tan-Sang; Jakub, Lokoc; Tim, Yue Him Wong; Joneja, Ajay; Yeung, Sai-Kit. Marine Video Kit: A New Marine Video Dataset for Content-based Analysis and Retrieval. 2022. Available from arXiv: 2209.11518 [cs.CV].

3. Gurrin, Cathal; Zhou, Liting; Healy, Graham; Bailer, Werner; Dang Nguyen, Duc-Tien; Hodges, Steve; Jónsson, Björn Þór; Lokoč, Jakub; Rossetto, Luca; Tran, Minh-Triet; Schöffmann, Klaus. Introduction to the Seventh Annual Lifelog Search Challenge, LSC'24. In: *Proceedings of the 2024 International Conference on Multimedia Retrieval*. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 1334–1335. ICMR '24. isbn 9798400706196. Available from doi: 10.1145/3652583.3658891.

4. Veselý, Patrik; Peška, Ladislav. RESET: Relational Similarity Extension for V3C1 Video Dataset. In: Rudinac, Stevan; Hanjalic, Alan; Liem, Cynthia; Worring, Marcel; Jónsson, Björn Þór; Liu, Bei; Yamakata, Yoko (eds.). *MultiMedia Modeling*. Cham: Springer Nature Switzerland, 2024, pp. 1–14. isbn 978-3-031-56435-2.

5. Rossetto, Luca; Gasser, Ralph; Lokoc, Jakub; Bailer, Werner; Schoeffmann, Klaus; Münzer, Bernd; Soucek, Tomás; Nguyen, Phuong Anh; Bolettieri, Paolo; Leibetseder, Andreas; Vrochidis, Stefanos. Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019. *IEEE Trans. Multim.* 2021, vol. 23, pp. 243–256. Available from doi: 10.1109/TMM.2020.2980944.

6. Lokoc, Jakub; Veselý, Patrik; Mejzlík, Frantisek; Kovalcík, Gregor; Soucek, Tomás; Rossetto, Luca; Schoeffmann, Klaus; Bailer, Werner; Gurrin, Cathal; Sauter, Loris; Song, Jaeyub; Vrochidis, Stefanos; Wu, Jiaxin; Jónsson, Björn Þór. Is the Reign of Interactive Search Eternal? Findings from the Video Browser Showdown 2020. *ACM Trans. Multim. Comput. Commun. Appl.* 2021, vol. 17, no. 3, 91:1–91:26. Available from doi: 10.1145/3445031.

7. Heller, Silvan; Gsteiger, Viktor; Bailer, Werner; Gurrin, Cathal; Jónsson, Björn Þór; Lokoc, Jakub; Leibetseder, Andreas; Mejzlík, Frantisek; Peska, Ladislav; Rossetto, Luca; Schall, Konstantin; Schoeffmann, Klaus; Schuldt, Heiko; Spiess, Florian; Tran, Ly-Duyen; Vadicamo, Lucia; Veselý, Patrik; Vrochidis, Stefanos; Wu, Jiaxin. Interactive video retrieval evaluation at a distance: comparing sixteen interactive video search systems in a remote setting at the 10th Video Browser Showdown. *Int. J. Multim. Inf. Retr.* 2022, vol. 11, no. 1, pp. 1–18. Available from doi: 10.1007/S13735-021-00225-2.

8. Schall, Konstantin; Bailer, Werner; Barthel, Kai Uwe; Carrara, Fabio; Lokoc, Jakub; Peska, Ladislav; Schoeffmann, Klaus; Vadicamo, Lucia; Vairo, Claudio. Interactive multimodal video search: an extended post-evaluation for the VBS 2022 competition. *Int. J. Multim. Inf. Retr.* 2024, vol. 13, no. 2, p. 15. Available from doi: 10.1007/S13735-024-00325-9.

9. Lokoc, Jakub; Andreadis, Stelios; Bailer, Werner; Duane, Aaron; Gurrin, Cathal; Ma, Zhixin; Messina, Nicola; Nguyen, Thao-Nhu; Peska, Ladislav; Rossetto, Luca; Sauter, Loris; Schall, Konstantin; Schoeffmann, Klaus; Khan, Omar Shahbaz; Spiess, Florian; Vadicamo, Lucia; Vrochidis, Stefanos. Interactive video retrieval in the age of effective joint embedding deep models: lessons from the 11th VBS. *Multim. Syst.* 2023, vol. 29, no. 6, pp. 3481–3504. Available from doi: 10.1007/S00530-023-01143-5.

10. Vadicamo, Lucia; Arnold, Rahel; Bailer, Werner; Carrara, Fabio; Gurrin, Cathal; Hezel, Nico; Li, Xinghan; Lokoc, Jakub; Lubos, Sebastian; Ma, Zhixin; Messina, Nicola; Nguyen, Thao-Nhu; Peska, Ladislav; Rossetto, Luca; Sauter, Loris; Schöffmann, Klaus; Spiess, Florian; Tran, Minh-Triet; Vrochidis, Stefanos. Evaluating Performance and Trends in Interactive Video Retrieval: Insights From the 12th VBS Competition. *IEEE Access*. 2024, vol. 12, pp. 79342–79366. Available from doi: 10.1109/ACCESS.2024.3405638.

11. Rossetto, Luca; Gasser, Ralph; Heller, Silvan; Parian-Scherb, Mahnaz; Sauter, Loris; Spiess, Florian; Schuldt, Heiko; Peska, Ladislav; Soucek, Tomás; Kratochvíl, Miroslav; Mejzlík, Frantisek; Veselý, Patrik; Lokoc, Jakub. On the User-Centric Comparative Remote Evaluation of Interactive Video Search Systems. *IEEE Multim.* 2021, vol. 28, no. 4, pp. 18–28. Available from doi: 10.1109/MMUL.2021.3066779.

12. Peska, Ladislav; Vomlelová, Marta; Veselý, Patrik; Skrhak, Vít; Lokoc, Jakub. Evaluating a Bayesian-like relevance feedback model with text-to-image search initialization. *Multim. Tools Appl.* 2023, vol. 82, no. 15, pp. 22305–22341. Available from doi: 10.1007/S11042-022-14046-W.

13. Tran, Ly-Duyen; Nguyen, Manh-Duy; Dang-Nguyen, Duc-Tien; Heller, Silvan; Spiess, Florian; Lokoc, Jakub; Peska, Ladislav; Nguyen, Thao-Nhu; Khan, Omar Shahbaz; Duane, Aaron; Jónsson, Björn Þór; Rossetto, Luca; Yen, An-Zi; Alateeq, Ahmed; Alam, Naushad; Tran, Minh-Triet; Healy, Graham; Schoeffmann, Klaus; Gurrin, Cathal. Comparing Interactive Retrieval Approaches at the Lifelog Search Challenge 2021. *IEEE Access*. 2023, vol. 11, pp. 30982–30995. Available from doi: 10.1109/ACCESS.2023.3248284.

14. Zezula, P.; Amato, G.; Dohnal, V.; Batko, M. *Similarity Search: The Metric Space Approach*. Springer US, 2006. Advances in Database Systems. isbn 9780387291512. Available also from: https://books.google.al/books?id=KTkWXsiPXR4C.

15. Andreadis, S. et al. VERGE in VBS 2022. In: Jonsson, B. Por et al. (eds.). *MultiMedia Modeling*. Cham: Springer, 2020, vol. 13142, pp. 778–783. Lecture Notes in Computer Science.

16. Lokoč, J.; Mejzlík, F.; Souček, T.; Dokoupil, P.; Peška, L. Video search with context-aware ranker and relevance feedback. In: Jonsson, B. Por et al. (eds.). *MultiMedia Modeling*. Cham: Springer, 2022, vol. 13142, pp. 505–510. Lecture Notes in Computer Science.

17. Jung, K.; Barthel, K.U.; Hezel, N.; Schall, K. PicArrange - visually sort, search, and explore private images on a mac computer. In: Jonsson, B. Por et al. (eds.). *MultiMedia Modeling*. Cham: Springer, 2022, vol. 13142, pp. 452–457. Lecture Notes in Computer Science.

18. Veselý, Patrik; Peška, Ladislav. Less Is More: Similarity Models for Content-Based Video Retrieval. In: Dang-Nguyen, Duc-Tien; Gurrin, Cathal; Larson, Martha; Smeaton, Alan F.; Rudinac, Stevan; Dao, Minh-Son; Trattner, Christoph; Chen, Phoebe (eds.). *MultiMedia Modeling*. Cham: Springer Nature Switzerland, 2023, pp. 54–65. isbn 978-3-031-27818-1.

19. Chen, Ting; Kornblith, Simon; Norouzi, Mohammad; Hinton, Geoffrey. A Simple Framework for Contrastive Learning of Visual Representations. 2020. Available from arXiv: 2002.05709.

20. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian. Deep Residual Learning for Image Recognition. 2015. Available from arXiv: 1512.03385 [cs.CV].

21. Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain; Uszkoreit, Jakob; Houlsby, Neil. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2021. Available from arXiv: 2010.11929 [cs.CV].

22. Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia. Attention Is All You Need. 2017. Available from arXiv: 1706.03762 [cs.CL].

23. Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. Available from arXiv: 1810.04805 [cs.CL].

24. Chen, Xi; Wang, Xiao; Changpinyo, Soravit; Piergiovanni, AJ; Padlewski, Piotr; Salz, Daniel; Goodman, Sebastian; Grycner, Adam; Mustafa, Basil; Beyer, Lucas; Kolesnikov, Alexander; Puigcerver, Joan; Ding, Nan; Rong, Keran; Akbari, Hassan; Mishra, Gaurav; Xue, Linting; Thapliyal, Ashish; Bradbury, James; Kuo, Weicheng; Seyedhosseini, Mojtaba; Jia, Chao; Ayan, Burcu Karagol; Riquelme, Carlos; Steiner, Andreas; Angelova, Anelia; Zhai, Xiaohua; Houlsby, Neil; Soricut, Radu. PaLI: A Jointly-Scaled Multilingual Language-Image Model. 2023. Available from arXiv: 2209.06794.

25. Zhai, Xiaohua; Mustafa, Basil; Kolesnikov, Alexander; Beyer, Lucas. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*. 2023.

26. Alabdulmohsin, Ibrahim; Zhai, Xiaohua; Kolesnikov, Alexander; Beyer, Lucas. Getting ViT in Shape: Scaling Laws for Compute-Optimal Model Design. 2024. Available from arXiv: 2305.13035 [cs.CV].

27. Beyer, Lucas; Zhai, Xiaohua; Kolesnikov, Alexander. *Big Vision* [https://github.com/google-research/big_vision]. GitHub, 2022 [visited on 2024-07-01].

28. Team, Google Research. *ViT-SO400M-14-SigLIP-384* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/timm/ViT-SO400M-14-SigLIP-384.

29. Team, Google Research. *ViT-L-16-SigLIP-384* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/timm/ViT-L-16-SigLIP-384.

30. Team, Google Research. *ViT-B-16-SigLIP-512* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/timm/ViT-B-16-SigLIP-512.

31. Fang, Alex; Jose, Albin Madappally; Jain, Amit; Schmidt, Ludwig; Toshev, Alexander; Shankar, Vaishaal. Data Filtering Networks. *arXiv preprint arXiv:2309.17425*. 2023.

32. Apple. *AXLearn* [online]. 2024. [visited on 2024-07-01]. Available from: https://github.com/apple/axlearn.

33. Apple, University of Washington. *DFN5B-CLIP-ViT-H-14-378* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/apple/DFN5B-CLIP-ViT-H-14-378.

34. Schuhmann, Christoph; Beaumont, Romain; Vencu, Richard; Gordon, Cade; Wightman, Ross; Cherti, Mehdi; Coombes, Theo; Katta, Aarush; Mullis, Clayton; Wortsman, Mitchell; Schramowski, Patrick; Kundurthy, Srivatsa; Crowson, Katherine; Schmidt, Ludwig; Kaczmarczyk, Robert; Jitsev, Jenia. LAION-5B: An open large-scale dataset for training next generation image-text models. 2022. Available from arXiv: 2210.08402 [cs.CV].

35. Beaumont, Romain. *LAION* [online]. 2024. [visited on 2024-07-01]. Available from: https://laion.ai/blog/laion-5b/.

36. Cherti, Mehdi; Beaumont, Romain; Wightman, Ross; Wortsman, Mitchell; Ilharco, Gabriel; Gordon, Cade; Schuhmann, Christoph; Schmidt, Ludwig; Jitsev, Jenia. Reproducible scaling laws for contrastive language-image learning. 2022. Available from arXiv: 2212.07143 [cs.LG].

37. Jitsev, Jenia. *ViT-g-14_laion2b_s34b_b88k* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/laion/CLIP-ViT-g-14-laion2B-s34B-b88K.

38. Beaumont, Romain. *ViT-H-14_laion2b_s32b_b79k* [online]. 2024. [visited on 2024-07-01]. Available from: https://huggingface.co/laion/CLIP-ViT-H-14-laion2B-s32B-b79K.

39. Jia, Chao; Yang, Yinfei; Xia, Ye; Chen, Yi-Ting; Parekh, Zarana; Pham, Hieu; Le, Quoc V.; Sung, Yunhsuan; Li, Zhen; Duerig, Tom. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. 2021.

40. Tan, Mingxing; Le, Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 2019. Available from arXiv: 1905.11946 [cs.LG].

41. Li, Junnan; Li, Dongxu; Savarese, Silvio; Hoi, Steven. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. 2023. Available also from: https://arxiv.org/abs/2301.12597.

42. Radford, Alec; Narasimhan, Karthik; Salimans, Tim; Sutskever, Ilya. Improving language understanding by generative pre-training. 2018.

43. Li, Junnan; Li, Dongxu; Xiong, Caiming; Hoi, Steven. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. 2022. Available from arXiv: 2201.12086 [cs.CV].

44. Li, Dongxu; Li, Junnan; Le, Hung; Wang, Guangsen; Savarese, Silvio; Hoi, Steven C.H. LAVIS: A One-stop Library for Language-Vision Intelligence. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 31–41. Available also from: https://aclanthology.org/2023.acl-demo.3.

45. Randellini, Enrico. *Image and text features extraction with BLIP and BLIP-2: how to build a multimodal search engine* [online]. 2024. [visited on 2024-07-10]. Available from: https://medium.com/@enrico.randellini/image-and-text-features-extraction-with-blip-and-blip-2-how-to-build-a-multimodal-search-engine-a4ceabf51fbe.

46. Rossetto, Luca; Schuldt, Heiko; Awad, George; Butt, Asad A. V3C - a Research Video Collection. 2018. Available from arXiv: 1810.04401 [cs.MM].

List of Figures

List of Tables