Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

# DataComp Challenge

Distributed Systems Laboratory


Dustin Brunner

brunnedu@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

**Supervisors:**
Till Aczél, Benjamin Estermann
Prof. Dr. Roger Wattenhofer


December 22, 2023

# Acknowledgements

I would like to express my gratitude to my supervisors, Till Aczel and Benjamin Estermann from the Distributed Computing Group (DISCO) at ETH Zürich for providing their valuable guidance and support throughout my distributed systems laboratory. Their insights and expertise were crucial in shaping my research and approach to the Datacomp challenge.

I am also thankful to Professor Roger Wattenhofer for enabling this enriching semester project and providing an opportunity to engage in such a novel and exciting field of study.

My appreciation extends to the Computer Engineering and Networks Laboratory (TIK) at ETH Zürich for providing the Arton compute cluster, which offered the resources and computational facilities for conducting my experiments.

Additionally, I would like to express my gratitude to my fellow students and friends, whose support and discussion have been a source of motivation and insight during my work on this project.

# Abstract

This report outlines our participation in the small filtering track of the DataComp challenge, a competition aimed at curating a training dataset from a larger pool to optimize the performance of a CLIP model trained on this dataset across a wide array of benchmark tasks. Our initial strategy centered on developing a neural network, the Quality Comparison Model (**QCM**), to assess the usefulness of image-caption pairs for inclusion in the training set based on a comparative approach. Later, after observing that the model excelled at content alignment but didn't properly factor in the semantic similarity between the image and caption of a given sample, we shifted our focus to content alignment and relied on another technique, flipped CLIP similarity, for cross-modality filtering. One key finding of our experiments is that a simple neural network, such as our proposed Content Alignment Model (**CAM**), is sufficient for content alignment between two large-scale datasets based on precomputed ViT-B/32 CLIP or DINOv2 embeddings. Due to its simplicity, our final approach benefits from very high computational efficiency which is desirable when handling large-scale datasets. With this approach, we were able to compete at the top of the leaderboard for the small-scale DataComp filtering track. Our research provides valuable insights into effective content alignment using precomputed image and text embeddings, along with presenting novel approaches to ranking methods for generating a ranking from randomly sampled pairwise comparisons.

# Contents

Chapter 1

# Introduction

This distributed systems lab report details the discoveries and outcomes of our participation in the DataComp data filtering competition [1], a unique challenge focusing on the curation of an image-caption dataset for pre-training a CLIP model [2]. Unlike previous machine learning competitions that focused on improving model architectures, DataComp emphasizes the importance of the data curation process. In this way, DataComp encourages innovation in filtering techniques to further stretch the boundaries of performance and make pre-training more efficient.

Our decision to participate in the smallest available scale of the filtering track of the DataComp competition enabled us to explore various approaches within our resource limits. In this track, participants are provided with an image-caption dataset pool from Common Crawl. The challenge is to effectively filter this pool, with the goal of training a CLIP model that achieves optimal performance across a large number of downstream tasks.

The main purpose of this report is to document the different approaches comprehensively and to provide a clear pathway for subsequent efforts in this direction. The motivation behind this project lies in the recent breakthroughs in multimodal learning such as CLIP [2], DALL-E [3] and GPT-4 [4] that heavily rely on large-scale image-text datasets. It has been shown in multiple cases that filtering a large-scale dataset crawled from the web can have a significant positive impact on performance. This was for example the case for a number of LAION datasets such as LAION-400M [5] or LAION-5B [6].

As we progressed through the competition, our initial idea focused on creating a neural network capable of assessing the quality of individual samples. However, recognizing the inherent challenges of this idea, we shifted our approach to a comparative model that would assess the relative quality of a pair of image-caption samples. This pivot required us to experiment with a variety of methods for deriving a ranking based on randomly sampled pairwise comparisons. Our experiments demonstrated that although the neural network did not directly learn the notion of a "high-quality" sample, it excelled at detecting samples that were similar in content to the high-quality datasets used for training. This observation

prompted us to further refine our approach.

In the end, splitting the filtering into cross-modality filtering and content alignment proved to be the key. Cross-modality filtering would ensure that only samples with high semantic similarity between the image and the caption would be kept and content-alignment ensured that samples with similar contents to datasets used in downstream tasks would be preferred. For cross-modality filtering, we would use the newly proposed flipped CLIP similarity by the The Devil is in the Details [7] team also participating in the challenge. For content alignment we relied on a symmetric single linear layer model, eliminating the need for comparisons and complex ranking algorithms. This streamlined approach wasn't only able to get competitive results on the average performance over the full benchmark suite but also achieved the highest ImageNet validation accuracy among all participants on the small scale [1].

The structure of this report reflects our progression through the DataComp competition. Following this introduction, Section 2 delves into the related work, providing context and background for our methodologies. Section 3 outlines the methodologies we employed, detailing the evolution of our strategies. Section 4 presents our experiments, results, and analyses. Finally, Section 5 concludes with reflections on our findings and potential directions for future work in this exciting new domain of data curation research.

[1] This result is achieved when only training and evaluating once using a random seed of 0 which is standard practice on the public leaderboard.

Chapter 2

# Related Work

In previous machine learning competitions, the conventional focus was often on refining models and their training procedures while keeping training data and evaluation metrics fixed. However, the DataComp challenge [1] provides an interesting deviation from this norm by highlighting the significance of dataset creation in machine learning. By participating in DataComp, our research aligns with this innovative perspective, exploring various novel filtering techniques for the optimization of training datasets within a fixed model and evaluation framework.

In this section, we start by providing some important background information about the DataComp challenge. Afterward, we give an overview of the CLIP [2] model architecture developed by OpenAI, which is a multimodal embedding model and forms the cornerstone of the competition. We then review some popular dataset filtering techniques that have been applied in the past and outline novel methodologies employed by other teams in DataComp, highlighting their differences and similarities to the approaches we pursued. These insights not only provide context for our methods but also show the variety of possibilities for improvement in dataset curation.

## 2.1 DataComp

DataComp [1] is a competition centered around dataset experiments proposed by a large group of researchers affiliated with a wide range of institutions including the University of Washington, LAION, Google Research and Apple. In contrast to past benchmarks that focused on model development, DataComp focuses on the training data generation process for a fixed CLIP [2] model architecture. DataComp offers 2 tracks that participants can choose to follow, the first being the filtering track and the second being the BYOD (Bring Your Own Data) track. The goal of both tracks is to generate a training dataset for a CLIP model so that it then performs as well as possible on a large number of zero-shot downstream image classification and image retrieval tasks on datasets such as ImageNet [8],

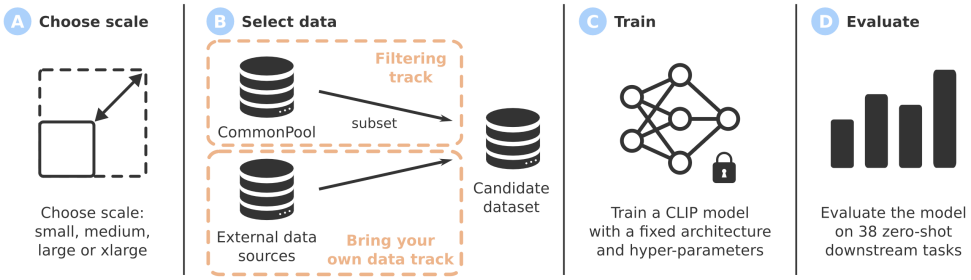


Figure 2.1: Overview of the DataComp workflow from the original paper [1] 1.) Selecting scale of original pool (**small-xlarge**). 2.) Filtering track: filtering original pool into a subset. 3.) Training a fixed CLIP model on filtered subset with fixed hyperparameters. 4.) Evaluating trained CLIP model on 38 zero-shot downstream tasks.

MS-COCO [9] or MNIST [10]. Figure 2.1 presents an overview of the workflow for the DataComp competition.

As we're committed to the filtering track, the report will focus on it and we won't discuss the BYOD track any further. In the filtering track participants are provided with a pool of image-text samples from Common Crawl [11]. Depending on the chosen scale (**small**, **medium**, **large**, **xlarge**), the initial pool in the competition contains anywhere from 12.8 million to 12.8 billion samples. Table 2.1 gives an overview of the different scales available in DataComp, due to hardware limitations and quicker feedback loops we focused on the **small** scale.

| Scale | Model | Train compute (MACs) | Pool size and # samples seen |
|---|---|---|---|
| **small** | ViT-B/32 | $9.5 \times 10^{16}$ | 12.8M |
| **medium** | ViT-B/32 | $9.5 \times 10^{17}$ | 128M |
| **large** | ViT-B/16 | $2.6 \times 10^{19}$ | 1.28B |
| **xlarge** | ViT-L/14 | $1.1 \times 10^{21}$ | 12.8B |

Table 2.1: Different scales available in DataComp.

The Common Crawl samples all originate from the web and consist of an image and a corresponding alt-text which is an HTML attribute that is supposed to describe the contents of the image in case the image can't be rendered or for accessibility reasons such as for screen readers. DataComp applies only minimal preprocessing to the pools in the form of NSFW detection, evaluation set deduplication, and face blurring. The participants are then supposed to filter the pool to an arbitrary subset that may also contain the same sample multiple times, while it isn't allowed to modify the samples in any way. The goal is to find

the optimal filtered subset so that a CLIP model trained on this subset performs as well as possible on the downstream benchmark tasks.

In order to make it possible to compare different-sized subsets as fairly as possible the model architectures as well as all the training hyperparameters are fixed including the number of samples seen during training which is fixed to the pool size of the scale chosen as shown in Table 2.1.

DataComp also provides several baselines for the filtering track such as basic filtering by language, caption length, or image size, and the LAION-2B filtering which consists of English language filtering in conjunction with CLIP similarity filtering. For CLIP similarity filtering a different pre-trained CLIP model is utilized to determine the semantic similarity between the image and the corresponding caption of a sample. This cross-modality filtering technique is applied to ensure the caption describes the visual content of the image. Furthermore, DataComp also provides baselines for text-based filtering and image-based filtering. Text-based filtering involves retaining samples whose captions have some overlap with ImageNet class names. Image-based filtering clusters CLIP image embeddings from the entire datasets and retains only those samples that belong to the nearest neighbor cluster of an embedded ImageNet sample.

In order to compare the performance of different filtering methods DataComp utilizes a diverse set of 38 downstream image classification and retrieval benchmark tasks. The benchmarks range from a large part of the VTAB (Visual Task Adaptation Benchmark) [12] suite, over popular image classification tasks such as MNIST [10], CIFAR [13], or ImageNet [8] to image and text retrieval tasks on Flickr30k [14] or MSCOCO [9]. Additionally, they include the two fairness and bias benchmarks Dollar Street [15] and GeoDE [16]. Therefore different filtering approaches are rated based on the average performance over all downstream benchmark tasks.

### 2.1.1 Contrastive Language-Image Pre-Training (CLIP)

CLIP (Contrastive Language-Image Pre-Training) [2] is a multimodal embedding model developed by OpenAI consisting of an image and a text encoder that embeds images and text descriptions of images into a shared embedding space.

It is trained using a contrastive learning approach on a large set of image caption pairs typically obtained by web crawling. This means that for a given batch both encoders are optimized to maximize the cosine similarity between an image's embedding and the embedding of its corresponding caption, while also trying to minimize the cosine similarity between embeddings of image caption pairs within the batch that do not match. This novel learning approach allows the model to link visual and textual data significantly better compared to previous image classification models. Figure 2.2 provides an overview of the CLIP train and prediction workflow.

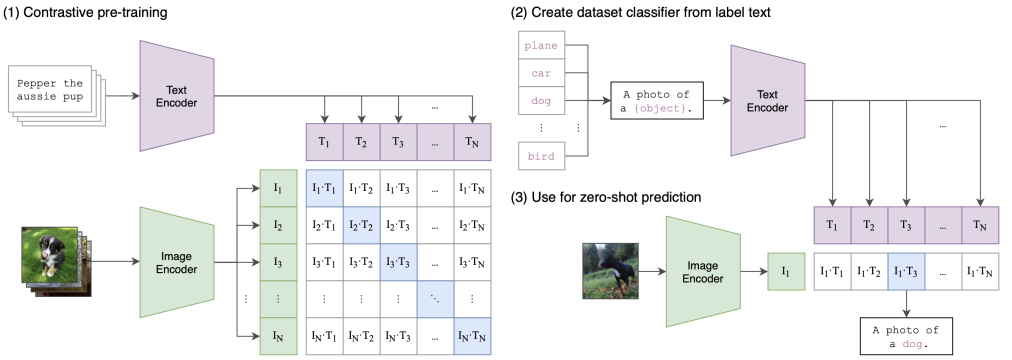


Figure 2.2: Summary of the approach used by CLIP from the original paper [2] 1.) Contrastive pre-training with maximization of cosine similarity between image and text embedding vectors coming from the same sample (diagonal) and minimization if they don't come from the same sample (off-diagonal). 2.) Transformation into a classifier by embedding all labels after inserting them into a medium phrase. 3.) Use for zero-shot prediction by embedding image and returning the label whose corresponding embedding has the highest similarity.

A trained CLIP model can then be used for zero-shot classification tasks by embedding the image to be classified and all labels it could be classified as and then selecting the label whose embedding shows the highest cosine similarity to the embedding of the image as the prediction. Usually, the labels are first transformed into image descriptions by the use of a so-called medium phrase like "a photo of a {LABEL}.".

This approach is unique in that it allows the model to learn a large variety of visual concepts during pre-training on a very large dataset and to then be applied to nearly arbitrary downstream classification tasks. That property also causes zero-shot classifications from CLIP to be more robust compared to classic discriminative models that are only trained to distinguish between a relatively small set of different classes and are therefore more likely to overfit on non-semantic information. In the CLIP paper, this property was demonstrated by the significantly better performance of a zero-shot CLIP classifier compared to a ResNet101 trained on ImageNet on a variety of distribution shifts of ImageNet such as ImageNetV2 [17], ObjectNet [18], or ImageNet Sketch [19].

Another task that CLIP excels at is the task of cross-modal retrieval where the objective is to retrieve the semantically most similar images to a textual description or the other way around the most similar textual descriptions to an image. This ability is very useful for searching through a large image database for example.

## 2.2 Filtering Approaches

In this section, we aim to outline several effective filtering strategies, including LAION-2B and DataComp-1B. We'll also discuss filtering methods from other teams in the DataComp competition that either inspired our approach or shared similarities with our ideas.

### 2.2.1 LAION-2B & DataComp-1B

Stable Diffusion is a state-of-the-art image-to-text generative model by LAION AI trained on LAION-2B [6] a massive 2.3 billion sample image-text dataset filtered from a comparable pool of samples to Common Crawl. For the creation of the LAION-2B dataset, first language filtering is applied to the original Common Crawl pool so that only samples with English texts are kept. Next, CLIP similarity filtering is applied to filter samples with high similarity between image and text based on a pre-trained CLIP model from OpenAI. More specifically, they used a ViT-B/32 model to embed both the image and the text of a given sample and then kept only samples whose embeddings had a cosine similarity above a predefined threshold. The logic behind this idea is that a model trained on samples with a higher semantic similarity between image and text would be able to better learn the relationship between both modalities.

With the publication of the DataComp [1] paper, the authors also published the DataComp-1B dataset representing the best baseline filtering method on the DataComp benchmark. DataComp-1B was filtered from the largest filtering track pool **xlarge** with 12.8 billion samples by combining the two most promising filtering strategies and ended up including approximately 1.4 billion samples. More specifically it kept the intersection of the filtered subsets generated by CLIP similarity filtering with a ViT-L/14 CLIP model, similar to the LAION-2B filtering, and Image-based filtering. Image-based filtering has the goal of aligning the image content distribution to that of ImageNet. It does so by clustering the image embeddings extracted from a ViT-L/14 model of all images in the pool and clustering them into 100'000 groups using Faiss [20]. It then finds the nearest neighbor group for every ImageNet image and only keeps the samples belonging to these clusters. They either used ImageNet-21K [21] (14 million images) or ImageNet-1K [8] (1.2 million images) for this procedure. A CLIP ViT-L/14 model trained on DataComp-1B was able to achieve a new state-of-the-art zero-shot accuracy on ImageNet of 79.2% compared to the previous state-of-the-art of 78.5% achieved by a significantly larger ViT-g/14 CLIP model trained on LAION-2B which required an order of magnitude more train compute. Our approaches are similar but in contrast to this method, our approaches employ a custom Content Alignment Model (**CAM**), which serves a similar purpose but relies on a specialized neural network for content alignment, rather than a clustering approach.

### 2.2.2 Approaches by Other DataComp Teams

As we joined the DataComp Challenge a few months after its start, we could benefit from the approaches and ideas that had already been put forward by other competing teams. A notable strategy was from Data Filtering Networks [22], which aimed to enhance neural network-based filtering. They experimented with binary classification models to differentiate between high-quality dataset images (like ImageNet or CC12M) and low-quality ones (such as Common Crawl), using various encoder backbones. However, they found that these models underperformed compared to CLIP similarity filtering, which they attributed to the binary classification models' rigid assumptions on the content of high-quality images. They found that CLIP similarity filtering was more flexible in this regard. In the end, they found that it was crucial to train the CLIP model used for similarity filtering on very high-quality samples. With this insight, they were eventually successful by training a CLIP model on the HQITP-350M dataset, consisting of 357 million high-quality image-text pairs, which led to significant improvements over previous methods. Our approaches are similar to their binary classification model approach with the difference being that we employ a comparison task for training the models and that we ended up aligning to multiple downstream datasets instead of just one.

Another interesting approach was by T-MARS (Text Masking and Re-Scoring) [23], who addressed a flaw in standard CLIP similarity filtering. They noticed that the OpenAI pre-trained CLIP model often rated samples with written text on the images that overlapped with the caption to have an overly inflated semantic similarity between the image and text, essentially the model was performing Optical Character Recognition (OCR). T-MARS countered this by masking text on images and recalculating CLIP similarity, leading to a better alignment with the desired semantic similarity between images and captions.

Later, The Devil is in the Details [7] introduced a much simpler yet similarly effective method to reduce the model's tendency to perform OCR. They flipped images horizontally during processing, leveraging the fact that CLIP models were not trained with horizontal flip augmentation. This resulted in the CLIP similarity scores providing a more accurate reflection of semantic similarity. Another helpful contribution by The Devil is in the Details was the clear structuring of a filtering pipeline into single-modality filtering, cross-modality filtering, and distribution alignment. Single-modality filtering would for example be language filtering of the captions. Cross-modality filtering would ensure that the semantic content of the image and caption were similar and distribution alignment would ensure that the content of filtered samples aligns with the content of downstream benchmark tasks.

Chapter 3

# Methodology

Over the 4 months of working on the distributed systems laboratory and with the publication of other teams' approaches, we had to modify our approach multiple times. Our initial goal was to create a neural network that could determine the quality of individual samples for effective ranking and selection. Recognizing the inherent complexity in accurately assessing sample quality, we decided to use a comparative approach, comparing pairs of samples to determine the higher-quality one.

Because of our decision to use a comparative approach, we required methods for ranking the samples based on randomly selected pairwise comparisons. For this, we explored various ranking methods, including the Elo rating system, PageRank, and the HITS algorithm. During our experiments, we noticed that our initial Quality Comparison Model (**QCM**) was better at identifying samples with content similar to high-quality datasets, rather than learning the complete concept of what it means for a sample to be high quality. This insight led us to split our filtering approach into two separate parts: cross-modality filtering and content alignment with benchmark datasets. For the cross-modality filtering we decided to use the recently proposed flipped CLIP similarity and for content alignment our Content Alignment Model (**CAM**).

As the project progressed further, we discovered the effectiveness of a simple neural network in distinguishing between large datasets based only on their CLIP embeddings. This insight led us to simplify our model to a single symmetric linear layer, thus bypassing the need for ranking algorithms. This straightforward approach was successful, enabling us to achieve competitive results in the small filtering track of the DataComp challenge.

## 3.1 Quality Comparison Model (QCM)

To determine which of two image-text samples is of higher quality we built a fully modular comparison model that we called the Quality Comparison Model (**QCM**). This model operates by processing precomputed embeddings of the

input samples. Depending on the configuration, the model is fed either the embeddings of both modalities (image and text) or only the embeddings of a specified modality (image or text). If both modalities were input the model concatenates the respective embeddings of both modalities so that it ends up with a 512 (single-modality) or 1024-dimensional (dual-modality) representation of both input samples. These representations are then compressed by a shared linear compression layer to a lower dimensionality. At this point, the cosine similarity between the image and the text input embedding vectors, also called the CLIP similarity score, is optionally concatenated to the compressed representations. We employ this approach to enable our model to factor in the cross-modality semantic similarity of samples. Otherwise, it wouldn't be able to do so, as the architecture is too limited to learn the process of performing cosine similarity in the compression layer. Then both compressed representations are fed to a classification head whose output reflects a logit of the probability that the second input sample is of higher quality. As classification head either a single linear layer or a simple 2-layer MLP with a ReLU activation function for non-linearity is used. Figure 3.1 provides an overview of the whole **QCM** architecture.

### 3.1.1 Training & Testing

To train the **QCM** we require a high-quality image-text dataset, for which we usually use the COCO Captions 2014 [9] train split if not otherwise stated, and a low-quality image-text dataset, for which we usually use the small DataComp [1] pool originating from Common Crawl [11]. Then we iterate over all of the samples in the low-quality dataset and for each of them randomly sample an image-text pair from the high-quality dataset. The precomputed embeddings of the image and/or text, depending on the modalities used, of these two samples are then passed to the **QCM** in random order. If the first sample is the high-quality sample the label is set to 0 and otherwise it is set to 1. As objective function we use binary cross-entropy and we optimize the model using an Adam optimizer with the default learning rate of 0.001 and without weight decay. We usually train for 10 epochs and the validation accuracy is always very high ($>$0.95) with the train loss being very low.

During inference, we randomly select pairs of image-text samples from the small DataComp pool and compare them using the trained **QCM**. To ensure consistency, the same preprocessing steps applied during training are replicated at test time. The total number of comparisons is always a multiple of the original dataset size. For example, with a comparison factor of 10 and an original dataset size of $N$, we conduct $10N$ comparisons. In this case, each sample would be compared 20 times on average (i.e., $2 \times 10$).

To guarantee that every sample is compared at least once, thereby providing a basis for quality assessment, we employ a permutation-based sampling strategy.

Specifically, we generate a number of random permutations of the entire dataset equal to the comparison factor. These permutations are then concatenated resulting in a sequence and all consecutive pairs of samples within this sequence are compared with each other. This method ensures that all samples are compared equally as often.

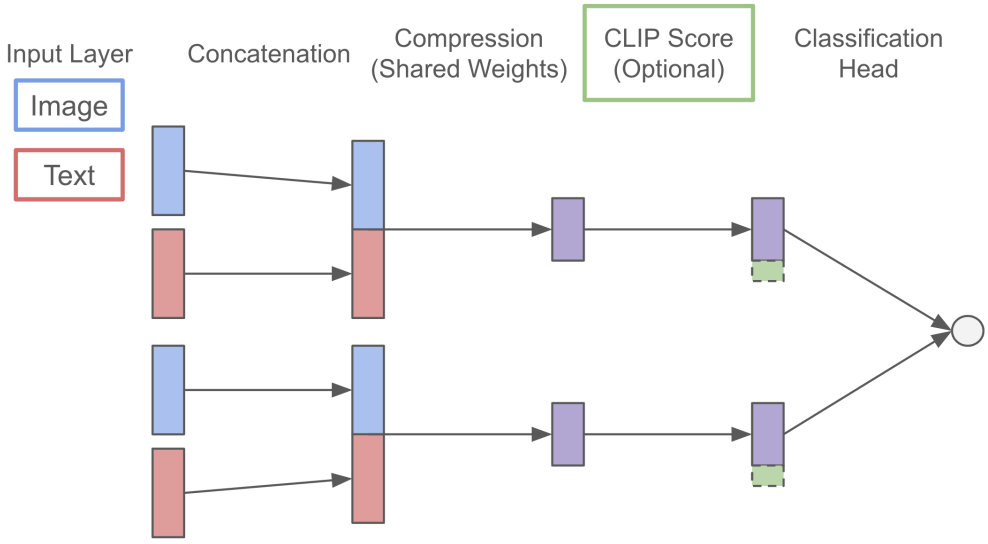


Figure 3.1: Overview of the **QCM** architecture. 1.) If the model is used in a dual-modality configuration, the image and text input embeddings are concatenated. 2.) The model compresses the representations of both samples using a linear compression layer, which employs shared weights for both inputs. 3.) Optionally, the inner product between the image and text input embeddings, also known as CLIP similarity score, is concatenated to the compressed representations. 4.) The representations for both samples are fed together to a classification head.

### 3.1.2 Ranking from Pairwise Comparisons

To retrieve a ranking from randomly sampled pairwise comparisons of dataset items we require a ranking algorithm. There are several such algorithms available that have been developed with various use cases in mind, such as the Elo rating system [24] that was developed for ranking chess players based on their game outcomes, the PageRank [25] algorithm famous for revolutionizing the way relevant web search results were displayed to the user developed by Google, or HITS (Hyperlink-Induced Topic Search) [26], which is designed to identify the most authoritative and hub pages on the internet by analyzing the pattern and quality of links between websites. In the following sections, we will introduce the ranking methods we experimented with.

### Elo Rating System

The Elo rating system [24] named after its inventor Arpad Elo is a method for calculating the relative skill levels of players in a zero-sum game such as chess. In the beginning, it assigns each player a numerical rating, a common choice is 1500, which then gets updated continuously based on the outcomes of games. More specifically, after every game the winning player gets some points from the losing player while the number of points transferred depends on the rating difference between the two players. If the winning player was previously rated significantly lower than the losing player the ratings will be corrected stronger compared to the case where the winning player was already rated above the losing one. In this way, the Elo system makes it possible to make statements about the expected outcome of a game based solely on the ratings of the players competing as described by Theorem 3.1. The Elo update rule as defined in Algorithm 1 also has a hyperparameter $K$ scaling the size of the updates but because in our case we're only interested in the order of the ratings and not in their absolute values we leave it at the default value of 32. In order to generate a ranking from pairwise comparisons using the Elo algorithm we iterate over all comparisons once in random order and perform the Elo updates sequentially.

**Theorem 3.1** (Elo Expected Outcome)**.** *For two players with Elo ratings $R_A$ and $R_B$, the expected probability of player A winning is given by:*

$$P(A \text{ wins}) = \frac{1}{1 + 10^{(R_B - R_A)/400}} \tag{3.1}$$

**Elo with Convergence.** One issue with the Elo algorithm is that the final ratings depend on the order in which we iterate through the comparisons which is completely arbitrary. Therefore, we modified the classic Elo algorithm to make the resulting ranking from the Elo algorithm less dependent on the order in which we iterate over the comparisons and update the ratings. Instead of iterating over all comparisons and updating the ratings just once, we repeatedly iterate over the whole set of comparisons until the change in the ranking between two consecutive passes over all comparisons is sufficiently small. To measure the magnitude of the change in the ranking between two consecutive passes we use the Kendall $\tau$ coefficient which is a rank correlation coefficient used for measuring the similarity between two rankings of items. Therefore, we repeatedly update all ratings based on all comparisons until the Kendall $\tau$ coefficient is below a predefined threshold of 0.001. The full Elo with convergence algorithm is detailed as a pseudo algorithm in Algorithm 2.

### PageRank

PageRank [25] is an algorithm developed by Google to determine the relevance of different linked web pages from the internet. It is named after one of the co-

**Algorithm 1** Elo Update

**procedure** EloUpdate($loserRating, winnerRating, K = 32$)
    $winnerOdds \leftarrow 10^{(loserRating - winnerRating)/400}$
    $expectedWinProb \leftarrow 1/(1 + winnerOdds)$
    $loserNewRating \leftarrow loserRating - K \cdot (1 - expectedWinProb)$
    $winnerNewRating \leftarrow winnerRating + K \cdot (1 - expectedWinProb)$
    **return** $loserNewRating, winnerNewRating$
**end procedure**

**Algorithm 2** Elo with Convergence

$comparisons \leftarrow$ vector of sample index tuples $(winner, loser)$
$Ratings \leftarrow$ vector of 1500 for each data sample
$convergenceThreshold \leftarrow 0.001$
$converged \leftarrow$ False
**while not** $converged$ **do**
    $prevRatings \leftarrow Ratings$
    **for** $(winner, loser)$ **in** $comparisons$ **do**
        $winnerRating, loserRating \leftarrow Ratings[winner], Ratings[loser]$
        $Ratings[loser], Ratings[winner] \leftarrow$ EloUpdate$(loserRating, winnerRating)$
    **end for**
    $kendallTau \leftarrow KendallTauCoef(prevRatings, Ratings)$
    **if** $kendallTau < convergenceThreshold$ **then**
        $converged \leftarrow$ True
    **end if**
**end while**

founders of Google, Larry Page, as well as the term "web page". It works on the principle that a web page's relevance or importance is related to the number of hyperlinks on other web pages pointing to it and the quality of those incoming links. First, each web page is assigned an initial rank. Then each web page repeatedly distributes a portion of its rank to all other web pages it links to. The page ranks can be seen as a vector and the distribution process can be seen as a matrix-vector multiplication between the transition matrix determined by the links and the page rank vector. Additionally, PageRank uses a damping factor to simulate the probability of a user clicking on a link on a given page. A popular default value for the damping factor is 0.85 which we also end up using. The PageRank algorithm repeats the distribution process until the page ranks converge and don't change by more than a predefined threshold. Because this process can also be seen as potentiating the transition matrix, the power iteration method is a popular way to calculate PageRank rankings. To be able to apply PageRank to our use case, we treat each data sample in our original pool as a node of a graph. Whenever two samples are compared with each other we add an edge from the lower-quality sample to the higher-quality sample based on the outcome. After building a comparison graph in this manner we can use the PageRank algorithm to generate a ranking over all samples in the data pool.

#### HITS (Hyperlink-Induced Topic Search)

The HITS algorithm is a link analysis algorithm developed by Jon Kleinberg that rates web pages. Its core idea is to determine two types of web pages. One type is 'hubs', which are pages that link to many other pages, and the other type is 'authorities', which are pages that are linked to by many hubs. Therefore it assigns every page two scores, a hub score, and an authority score. Initially, every page starts with hub and authority scores of 1. The algorithm then updates these scores iteratively. A page's authority score is updated based on the sum of the hub scores of the pages pointing to it, and its hub score is updated based on the sum of the authority scores of the pages it points to. After each iteration, the scores are normalized to prevent them from inflating excessively. The HITS algorithm repeats this process until the scores converge and do not change significantly anymore between consecutive iterations. To apply HITS to our problem, we build a comparison graph as described in the previous section about PageRank and use the resulting authority scores as the sample qualities.

## 3.2 Content Alignment Model (CAM)

In order to effectively align the train set distribution to our evaluation sets distributions we modified the original **QCM** (Quality Comparison Model) slightly by making its architecture completely symmetric. The modified model is called **CAM**, standing for Content Alignment Model, because of its new purpose compared to the **QCM**. The main difference between the two model architectures is that the **CAM** doesn't have a classification head and instead, the symmetric compression layer serves directly as a content-alignment layer that outputs a scalar value indicating the input sample's similarity to the dataset we wish to align to. This makes the **CAM** architecture exceptionally straightforward as the compression layer is linear and therefore the outputted alignment score of a sample is simply the inner product between the model's weight vector and the sample's input embedding. For training, still the same comparison approach is utilized as for the **QCM** detailed in Section 3.1.1 but for inference, each sample only has to be passed once through the compression layer to determine its alignment score.

Because the purpose of the **CAM** is to align the distribution of samples we include in the train set to the distribution of samples in the downstream task datasets there is now an additional requirement for a cross-modality method to ensure that samples whose image and corresponding caption contain similar semantic information are preferred. For this, we employ the recently developed technique by the The Devil is in the Details [7] team competing in DataComp called flipped CLIP similarity. More specifically, we use a pre-trained ViT-B/32 CLIP model from OpenAI to embed the horizontally flipped image and the text of each sample in the small DataComp pool and calculate their inner product, which gives us the cosine similarity for each sample. Flipped CLIP similarity is an improvement over the already widely-used classic CLIP similarity because it reduces its tendency to assign samples for which part of the caption is included as written text somewhere in the image an overly inflated semantic similarity. This is because OpenAI didn't apply any horizontal flipping as data augmentation when training CLIP models.

To factor in both the content alignment score as well as the cross-modality flipped CLIP similarity score the ranks of all samples with respect to both scores are averaged to determine a sample's overall rank.

Chapter 4

# Experiments

This section of the report outlines the experiments we conducted as part of the distributed systems laboratory. It aims to evaluate the effectiveness of the various filtering approaches we pursued throughout our participation in the DataComp challenge. The primary focus is on the different configurations of the **CAM** and **QCM** models as well as on the variety of ranking methods including Elo, HITS, and PageRank.

## 4.1 Data & Tools

### 4.1.1 Data

The main datasets used during the project are from the DataComp competition. Because of the massive datasets used in the competition, the participants have to download them directly from the Common Crawl web scrapes which specify for every sample the image URL as well as the Alt text that is to be used as the caption. With a success rate of approximately 90%, we were able to download 11.6M of the 12.8M samples in the small pool. For evaluation DataComp uses a wide range of 38 benchmark tasks whose corresponding datasets are detailed in Table 15 of the original DataComp [1] paper.

Additionally, for content alignment and training the quality models, we used a number of train sets or other unrelated partitions to the test sets of some of the evaluation datasets. Table 4.1 provides an overview of these additional datasets.

### 4.1.2 Tools

The project's codebase was developed entirely in Python [27]. We leveraged the PyTorch [28] framework for constructing our neural network architectures, complemented by PyTorch Lightning [29] for streamlined and scalable coding. For handling large-scale datasets, notably the 450GB DataComp small pool, we employed WebDataset, a Python library designed for efficient data loading from

Table 4.1: Overview of datasets used in addition to DataComp datasets.

| Dataset | Partition | Size | Number of Classes |
|---|---|---|---|
| MSCOCO | train | 82783 | N/A |
| ImageNet-1K | train | 1281167 | 1000 |
| CIFAR-10 | train | 50000 | 10 |
| Food-101 | train | 75750 | 101 |
| SVHN | train | 65931 | 10 |
| MNIST | train | 60000 | 10 |
| Oxford 102 Flower | train | 1020 | 102 |
| Oxford-IIIT Pet | train | 2944 | 37 |
| Caltech-101 | train | 2754 | 101 |
| PatchCamelyon | train | 262144 | 2 |
| EuroSAT | VTAB train | 16200 | 10 |
| iWildCam | train | 129809 | 182 |

web storage and local file systems. Ranking algorithms like HITS and PageRank were implemented using NetworkX [30], a comprehensive network and graph analysis library in Python.

Pre-trained CLIP models from OpenAI were integrated via the OpenCLIP [31] library. Additionally, the HuggingFace platform served as a valuable resource for accessing other pre-trained models and downloading various datasets. For visualizations and analysis, we utilized Matplotlib [32] and Plotly [33], two widely-adopted plotting libraries, while Pandas [34] was our choice for data manipulation and analysis, particularly beneficial for working with smaller datasets.

## 4.2 Ranking Methods

To evaluate the various ranking methods based on their ability to retrieve a ranking from randomly sampled pairwise comparisons of data samples from a large pool a simulation environment was utilized. More specifically, we assume that the ground truth qualities are sampled from a standard normal distribution with mean 0 and standard deviation 1 and that they are known beforehand. Then we randomly compare different samples with each other and based on the outcomes of these comparisons the methods have to recover the original ranking. The resulting ranking is then compared with the ranking based on the ground truth qualities using multiple metrics such as Spearman's or Kendall's rank correlation coefficient, sensitivity at K (how many of the top K% samples were correctly identified as such), or a custom Ranking Distance metric we developed whose calculation is detailed in Figure 4.1.

```python
def ranking_distance(qualities: np.ndarray, predicted_qualities: np.ndarray, k:
↪  float = 0.2):
    if isinstance(k, float):
        # if k is fraction, convert to integer
        k = round(k * len(qualities))

    # 1.
    # indices of the k highest value items for both groundtruth and prediction
    top_k_groundtruth = qualities.argsort()[-k:]
    top_k_prediction = predicted_qualities.argsort()[-k:]
    ranking_qualities = qualities.argsort().argsort()

    # 2.
    # sum distances from top-k for all items wrongly predicted to belong to the
    ↪  k highest value items
    distances = []
    for pred in top_k_prediction:
        if pred not in top_k_groundtruth:
            distances.append(len(qualities) - ranking_qualities[pred] - k)
    distances = np.array(distances)

    # 3.
    # calculate maximal possible distance
    n = len(qualities)
    max_n_out = min(n - k, k)  # max number of items to sum over
    ub_dist = n - k  # max possible distance for single item
    lb_dist = ub_dist - max_n_out + 1  # min possible distance for single item
    max_dist = max_n_out * (ub_dist + lb_dist) / 2

    dist = distances.sum() / max_dist

    return dist
```

Figure 4.1: Python function for calculating the Ranking Distance metric. 1.) Identify all the samples that were mistakenly predicted to be among the top-k (highest-ranking) samples. 2.) For all these samples calculate how far away their actual rank is from being in the top-k group and sum the distances over all of them. 3.) Normalize the resulting sum so that it falls in the range between 0 (best case) and 1 (worst case).

The Ranking Distance metric is an extension of the sensitivity at K metric in that it not only considers how many of the top K% items were correctly identified but also sums for all the incorrectly identified items outside of the specified threshold their distance from the threshold. To make the ranking distance easier to interpret it is normalized so that its values are between 0, which would be the case if the sensitivity was 100%, and 1, which represents the absolute worst case of predicting the bottom K% items to be the top K% items.

Each simulation run was defined by the following variables: the number of items $N$, the comparison factor $\alpha$ which together with the number of items would determine the number of comparisons performed $\alpha N$, the standard deviation of random Gaussian noise added to the ground truth qualities when comparing two items $\sigma$, and most importantly the ranking method used.

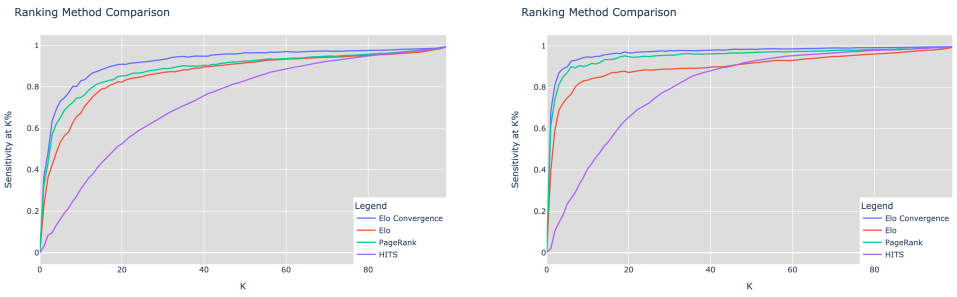


Figure 4.2: Comparison of sensitivity at varying thresholds of ranking methods with $N = 10000$ and $\sigma = 0$. In the chart on the left, a comparison factor of $\alpha = 10$ was used and on the right, a comparison factor of $\alpha = 50$ was used. The x-axis represents the percentage of the top-ranked items to be identified and the y-axis represents what fraction of them were correctly identified. Elo with Convergence has the highest sensitivity for identifying the top-ranked items at all thresholds.

| Ranking Method | Sensitivity at 20% ↑ | Ranking Distance at 20% ↓ | Kendall Coefficient ↑ | Spearman Coefficient ↑ |
|---|---|---|---|---|
| Elo Convergence | **0.918500** | **0.002905** | **0.911003** | **0.990010** |
| PageRank | 0.852500 | 0.009513 | 0.824707 | 0.960080 |
| Elo | 0.822500 | 0.015969 | 0.799190 | 0.951008 |
| HITS | 0.524000 | 0.101073 | 0.625254 | 0.819519 |

Table 4.2: Comparison of ranking methods with $N = 10000$, $\sigma = 0$ and $\alpha = 10$ using multiple ranking metrics. Elo with Convergence achieves the best results over all metrics and it can be observed that the metrics are highly correlated with each other.

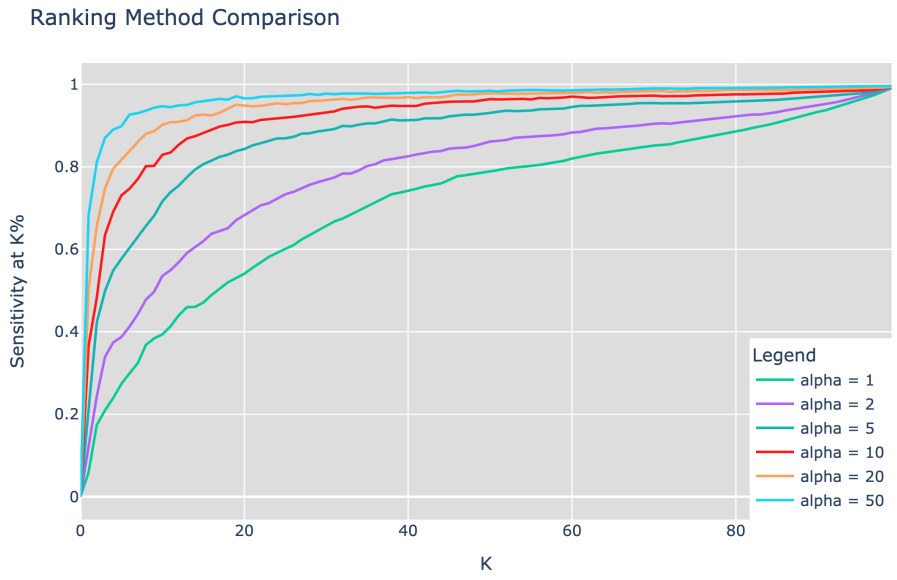


Figure 4.3: Comparison of sensitivities at varying thresholds when using different comparison factors ranging from 1 to 50. These experiments were conducted with Elo with Convergence, $N = 10000$, $\sigma = 0$. As expected, increasing the comparison factor has a positive impact on the sensitivities at all thresholds. It can also be seen that the sensitivity gain diminishes the larger the comparison factor becomes.

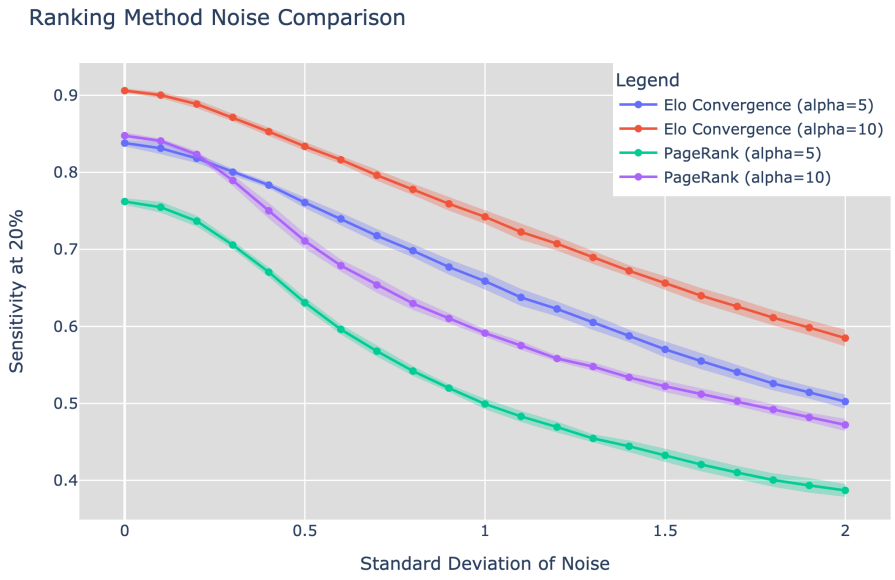


Figure 4.4: Comparison of sensitivities at the 20% threshold when comparisons are noisy at different noise levels between 0 and 2 with $N = 10000$. The ground truth qualities are sampled from a standard normal distribution with a standard deviation of 1, so at a noise level of 2, the noise is twice as large as the signal. The experiments were repeated with different random seeds and the error bars indicate the standard deviation over different varying seeds. Elo with Convergence and comparison factor $\alpha = 5$ performs comparably to PageRank with comparison factor $\alpha = 10$ indicating that Elo with Convergence is significantly more robust to noise in the comparisons.

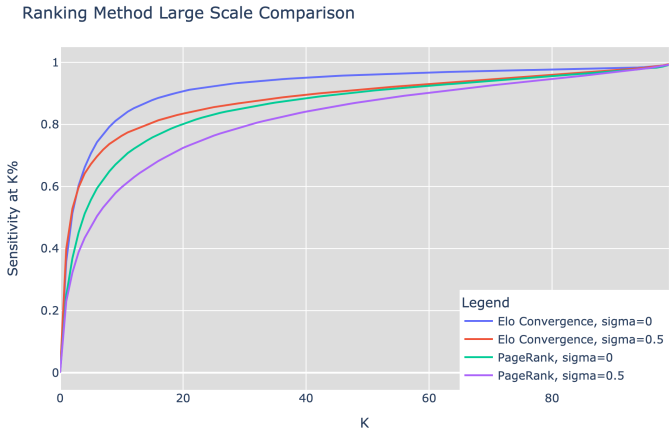


Figure 4.5: Comparison of sensitivities at varying thresholds for a large scale dataset with $N = 1e6$. The methods used are Elo with Convergence and PageRank either with noisy comparisons $\sigma = 0.5$ or with deterministic comparisons $\sigma = 0$. The comparison factor is $\alpha = 10$. The results show that a linear scaling of the number of comparisons performed is sufficient to maintain a high sensitivity for larger data sets. This means that insights gained from experiments performed on a smaller scale of $N = 10000$ should also generalize well to a larger scale.

## 4.3 Filtering Methods

### General Filtering Process

The general filtering process for the small DataComp pool using our proposed methods involves the following key steps:

1. **Score Assignment**: Each sample in the pool is assigned a score reflecting its potential utility in the filtered subset. This scoring is achieved using either a **QCM** or **CAM**, supplemented by other methods like CLIP similarity scores.

2. **Ranking**: Utilizing these scores, we establish a ranking over all samples in the small DataComp pool.

3. **Filtering**: The final filtered subset is created by retaining only the top-ranked samples. This selection is based on a predefined threshold, which determines the proportion of top-ranked samples to include in the subset. Ablation studies from the DataComp paper and empirical tests by other teams have generally shown that a threshold ranging from 15% to 30% optimizes the benchmark performance. Because of this, we used thresholds in this range for our experiments. It is generally observed that with more

effective filtering methods, a lower threshold is preferable, as these methods are better at excluding less useful samples.

### 4.3.1 Baselines

In order to provide some references for the DataComp evaluation results we present an overview of the evaluation results of popular filtering methods and general baselines. It's important to note that because we weren't able to download the complete small pool due to some missing URLs the DataComp percentages of top-ranked samples kept are in regards to the full small pool size of 12.8M while the percentages of our methods are in regards to 11.6M samples from the small pool we were able to successfully download. Table 4.3 provides an overview of some important baselines for the DataComp benchmark.

| Filtering Method | Dataset Size | ImageNet | 38 Dataset Average |
|---|---|---|---|
| No filtering | 12.8M | 0.025 [2] | 0.132 [2] |
| LAION-2B filtering | 1.3M | 0.031 [2] | 0.133 [2] |
| CLIP score (L/14 30%) | 3.8M | 0.051 [2] | **0.173** [2] |
| Image-based ∩ CLIP score (L/14 30%) | 1.4M | 0.039 [2] | 0.144 [2] |
| Flipped CLIP Score (B/32 25%) | 2.9M | **0.059** | 0.172 |

Table 4.3: Evaluation of Various Baseline Methods on DataComp's Small Scale Filtering Track. All experiments except for the Flipped CLIP Score experiment are taken from the DataComp paper. Filtering the top 30% of samples based on ViT-L/14 CLIP score provides the best performance over the full DataComp benchmark while filtering the top 25% according to flipped ViT-B/32 CLIP score results in the highest ImageNet-1K validation accuracy. It's important to consider that DataComp experiments were only conducted once with random seed 0, and with an observed variance of 0.006 across different seeds, it is difficult to attribute the observed advantages to specific causes.

### 4.3.2 Quality Comparison Model (QCM)

#### QCM Filtering Process

The filtering process using the **QCM** involves the following steps:

1. **Model Training**: The **QCM** is trained on a comparison task as outlined in Section 3.1.1. The training is focused on distinguishing between samples

[2]Note: The results presented are based on experiments only run once with a random seed of 0. The variance of the 38 dataset average between runs with different random seeds is up to 0.006. For results without this remark, the values are averaged over 3 runs with different random seeds.

from a lower-quality dataset, the small DataComp pool, and samples from a higher-quality dataset, the COCO Captions train split, based on their ViT-B/32 CLIP embeddings.

2. **Comparison Inference**: Randomly selected data samples from the small DataComp pool are compared against each other using the trained **QCM** that evaluates which sample in each pair is of higher quality. The sampling method for selecting data samples to compare is detailed in Section 3.1.1. Unless stated otherwise, a default comparison factor of 50 is used which based on our ranking method experiments proved to be sufficient for yielding a precise ranking.

3. **Ranking**: The outcomes of these pairwise comparisons are then used to generate a ranking of all samples. For the ranking, we utilize the Elo with Convergence method which performed best in our ranking method comparisons. In this way, the comparison results are converted into a coherent ranking that reflects the relative quality of each sample.

4. **Filtering**: Based on the established ranking, the top-ranked samples are filtered to generate the resulting subset. This subset selection is based on a pre-defined threshold, ensuring that only the most relevant samples, as determined by their rank, are retained for further use.

This process enables the effective utilization of the **QCM** for filtering the DataComp pool by systematically comparing, ranking, and selecting the most qualitatively relevant samples.

Table 4.4 provides an overview of the results using the dual-modality **QCM** in multiple configurations. Table 4.5 displays the results of experiments comparing combinations of various single-modality **QCM**s with CLIP similarity scores and two different reduction methods for merging the separate scores.

| Filtering Method | Dataset Size | ImageNet | 38 Dataset Average |
|---|---|---|---|
| QCM ($c = 8, d = 0.4$) 30% | 3.5M | **0.033** [2] | **0.149** [2] |
| QCM ($c = 8, d = 0.4$, clip=True) 30% | 3.5M | 0.032 [2] | 0.139 [2] |
| QCM ($c = 16$) 30% | 3.5M | 0.027 [2] | 0.140 [2] |
| QCM ($c = 2$) 30% | 3.5M | **0.033** [2] | 0.140 [2] |
| QCM ($c = 32, d = 0.4$) 30% | 3.5M | 0.032 [2] | 0.143 [2] |

Table 4.4: Evaluation of Various Dual-Modality **QCM** Model Configurations for Filtering on DataComp's Small Scale Filtering Track. The compression dimension $c$, the dropout probability $d$, and whether CLIP similarity was concatenated to the compressed sample representations after the compression layer "clip" is indicated in the parenthesis. The percentage represents the fraction of the top-ranked samples from the original pool kept. The experiments show that a compression dimension of $c = 8$ as well as using a dropout probability of $d = 0.4$ on the input layer without concatenating the CLIP score yields the best results.

| Filtering Method | Dataset Size | ImageNet | 38 Dataset Average |
|---|---|---|---|
| Image + Text + Flipped CLIP MR 30% | 3.5M | **0.054** [2] | 0.166 [2] |
| Image + Flipped CLIP MR 30% | 3.5M | 0.051 [2] | **0.167** [2] |
| Text + Flipped CLIP MR 30% | 3.5M | 0.049 [2] | 0.163 [2] |
| Image + Text + Flipped CLIP GS 30% | 3.5M | 0.042 [2] | 0.151 [2] |
| Image + Flipped CLIP GS 30% | 3.5M | 0.038 [2] | 0.149 [2] |
| Text + Flipped CLIP GS 30% | 3.5M | 0.039 [2] | 0.153 [2] |
| Image + Text + CLIP MR 30% | 3.5M | **0.054** [2] | **0.167** [2] |
| Image + Text + CLIP GS 30% | 3.5M | 0.042 [2] | 0.153 [2] |

Table 4.5: Evaluation of Various Methods for Reducing Single-Modality **QCM** Scores and CLIP Scores for Ranking and subsequent Filtering on DataComp's Small Scale Filtering Track. The modalities and CLIP scores included are indicated by the start of the filtering method name. The methods are named to reflect the included scores (single-modality **QCM** scores, CLIP scores), with 'MR' denoting Mean Ranking (average rank across included scores) and 'GS' denoting Geometric Score (geometric average across included scores). For instance, 'Text + Flipped CLIP MR' implies a single-modality **QCM** was trained on the captions (Text) and the final ranking is an average of the rank across this **QCM**'s quality score and the flipped ViT-B/32 CLIP score. The experiments show that the filtering based on the images and the flipped CLIP score achieves comparable performance to filtering based on both modalities and the flipped CLIP score.

### 4.3.3 Content Alignment Model (CAM)

#### CAM Filtering Process

To align our filtered data with the datasets for downstream benchmark tasks, we downloaded additional, non-overlapping train splits from as many of these datasets as possible. These splits, while separate from the test splits used in evaluations, shared a similar data distribution, making them suitable for training our alignment model. Given the diversity across the downstream task datasets, we categorized them into three groups – S, M, and L – based on the amount of data they should contribute to the resulting filtered subset.

We devised the following categorization based on the datasets' sizes and the specificity of their content:

**S** : PatchCamelyon, EuroSAT, CIFAR-10, SVHN, MNIST.

**M** : iWIldCam, Oxford 102 Flower, Food-101, Oxford-IIIT Pet, Caltech-101.

**L** : MSCOCO, ImageNet-1K.

The filtering process utilizing the **CAM** comprises the following steps:

1. **Model Training**: For each alignment dataset, we train a distinct **CAM**. This model focuses solely on differentiating samples from the small DataComp pool against those from an alignment dataset. For training, we employ the same comparison task outlined in Section 3.1.1 as for training the **QCM**. To distinguish samples from different datasets, the **CAM** relies exclusively on the 768-dimensional image embeddings from a DINOv2 model and doesn't utilize the captions.

2. **Alignment Score Inference**: After training, each sample from the small DataComp pool is passed through the **CAM**'s compression layer once to obtain its alignment score.

3. **Combination with Cross-Modality Score**: The alignment score of each sample is then combined with the flipped ViT-B/32 CLIP similarity score of the samples by averaging their respective ranks.

4. **Union over Alignment Datasets**: For each dataset we align to, a fraction of the top-ranked samples, determined by its category (S, M, L), is filtered based on the combination of the corresponding **CAM** alignment score and the cross-modality score. These subsets are then combined using a set union operation. Due to the substantial overlap among the subsets filtered from different alignment datasets, the final filtered subset is significantly smaller than the cumulative total of all individual fractions.

Table 4.6 provides an overview of the results using the **CAM** in various configurations.

| Filtering Method | Dataset Size | ImageNet | 38 Dataset Average |
|---|---|---|---|
| S 1% - M 5% - L 15% | 2.7M | 0.060 | 0.173 |
| S 0.5% - M 5% - L 15% | 2.7M | **0.062** | 0.175 |
| S 1% - M 3% - L 15% | 2.4M | 0.060 | **0.176** |
| S 1% - M 3% - L 20% | 3M | 0.058 | 0.173 |
| M 5% - L 15% | 2.7M | 0.060 | 0.174 |

Table 4.6: Comparative Evaluation of **CAM** Configurations for Alignment to Multiple Evaluation Datasets in DataComp. The percentages indicate what proportion of the top-ranked samples in the pool according to **CAM**s trained for aligning to the datasets in the corresponding categories (S, M, L) were included in the filtered subset.

### 4.3.4 Average Score by Samples

Near the end of the project, we implemented an averaging strategy for each sample in the original pool based on all the 48 evaluations of different filtered subsets we have run up until this point. For this, we calculated the average of all evaluation scores for each sample across all of the experiments in which it was included in the filtered subset. We then ranked the samples based on this averaged metric, anticipating an ensemble effect. However, contrary to our expectations, this approach did not yield the desired results. The benchmark performance using this filtering technique was subpar as shown in Table 4.7.

| Filtering Method | Dataset Size | ImageNet | 38 Dataset Average |
|---|---|---|---|
| Average Score by Sample 15% | 1.7M | 0.015 [2] | 0.118 [2] |
| Average Score by Sample 20% | 2.3M | 0.019 [2] | 0.133 [2] |
| Average Score by Sample 25% | 2.9M | **0.024** [2] | **0.139** [2] |

Table 4.7: Comparative Evaluation of Average Scores by Samples Filtering in DataComp. The percentage indicates what proportion of the top-ranked samples in the pool according to the average evaluation score of all experiments that a specific data sample was included in was filtered from the pool.

Chapter 5

# Conclusion & Future Work

This report has documented our journey through the DataComp challenges small filtering track, highlighting the exploration of various novel data filtering techniques. Our initial objective to compete at the top of the leaderboard was largely met, though not through the development of groundbreaking filtering techniques. Instead, our project focused on refining and enhancing existing methods, leading to a better understanding of the complex notion of what makes an image-caption sample useful and therefore of high quality for training a CLIP model.

A significant finding from our research was the effectiveness of simple models like the Content Alignment Model (**CAM**) in distinguishing between samples from different large-scale datasets relying only upon embeddings from a CLIP or DINOv2 model. This demonstrates the informativeness of the embedding spaces spanned by these embedding models. Our experiments also showed that there isn't any need for a non-symmetric comparison model such as the **QCM** and instead, a symmetric model like the **CAM** is sufficient. Another interesting discovery was the superiority of Elo with convergence over other ranking methods such as PageRank or Elo without convergence for retrieving a ranking from randomly sampled comparisons. Considering that the number of possible comparisons between a set of items scales quadratically with the pool size it was also surprising that it was sufficient to linearly scale the number of randomly sampled comparisons to maintain the desired ranking precision.

We faced several challenges during the project, especially dealing with large datasets and the slow feedback loops due to the demanding training process of the DataComp CLIP model. Another challenge we only noticed towards the end of the project was the high variance in the evaluation benchmark metrics between experiments with different random seeds. This was especially a problem on the small-scale filtering track where the evaluation metrics were generally quite small in magnitude.

In terms of impact, our experiments primarily demonstrated the validity of employing a simple neural network model instead of a clustering method, such as the one used for the image-based filtering baseline, for content alignment to downstream tasks.

On a personal note, this project was a very insightful experience in working with large-scale datasets and multi-GPU distributed training. It also enhanced my comprehension of the CLIP model, its embedding space, and the meaning of CLIP similarity scores.

In moving forward from our current work, several promising avenues exist to enhance the effectiveness of our approach, particularly in aligning to multiple downstream evaluation datasets using the **CAM**. One immediate area for improvement lies in refining our cross-modality filtering technique. Building on the innovative ideas of other participants in the DataComp competition, such as the Data Filtering Networks [22], we see potential in training the CLIP model used for CLIP score filtering exclusively on very high-quality datasets. This approach has shown promise in improving the performance on the DataComp benchmark suite. Another way of improving the cross-modality filtering is to perform the similarity calculation with a larger CLIP architecture such as a ViT-L/14.

Because the **CAM** has shown excellent computational efficiency and robustness in our experiments, its direct enhancement possibilities may be limited. Even though our experiments with more complex architectures didn't result in improved performance, future work could involve further experimenting with alternative neural network architectures.

# Bibliography


[1] S. Y. Gadre, G. Ilharco, A. Fang, J. Hayase, G. Smyrnis, T. Nguyen, R. Marten, M. Wortsman, D. Ghosh, J. Zhang, E. Orgad, R. Entezari, G. Daras, S. Pratt, V. Ramanujan, Y. Bitton, K. Marathe, S. Mussmann, R. Vencu, M. Cherti, R. Krishna, P. W. Koh, O. Saukh, A. Ratner, S. Song, H. Hajishirzi, A. Farhadi, R. Beaumont, S. Oh, A. Dimakis, J. Jitsev, Y. Carmon, V. Shankar, and L. Schmidt, "Datacomp: In search of the next generation of multimodal datasets," 2023.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[3] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," 2021.

[4] OpenAI, "Gpt-4 technical report," 2023.

[5] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," 2021.

[6] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev, "Laion-5b: An open large-scale dataset for training next generation image-text models," 2022.

[7] H. Yu, Y. Tian, S. Kumar, L. Yang, and H. Wang, "The devil is in the details: A deep dive into the rabbit hole of data filtering," 2023.

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

[9] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[10] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[11] "Common crawl," https://commoncrawl.org/, 2023.

[12] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby, "A large-scale study of representation learning with the visual task adaptation benchmark," 2020.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[14] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," 2016.

[15] W. Gaviria Rojas, S. Diamos, K. Kini, D. Kanter, V. Janapa Reddi, and C. Coleman, "The dollar street dataset: Images representing the geographic and socioeconomic diversity of the world," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 12 979–12 990. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/5474d9d43c0519aa176276ff2c1ca528-Paper-Datasets_and_Benchmarks.pdf

[16] V. V. Ramaswamy, S. Y. Lin, D. Zhao, A. B. Adcock, L. van der Maaten, D. Ghadiyaram, and O. Russakovsky, "Geode: a geographically diverse evaluation dataset for object recognition," 2023.

[17] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" 2019.

[18] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 9448–9458.

[19] H. Wang, S. Ge, Z. Lipton, and E. P. Xing, "Learning robust global representations by penalizing local predictive power," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 506–10 518.

[20] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," 2017.

[21] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," 2021.

[22] A. Fang, A. M. Jose, A. Jain, L. Schmidt, A. Toshev, and V. Shankar, "Data filtering networks," 2023.

[23] P. Maini, S. Goyal, Z. C. Lipton, J. Z. Kolter, and A. Raghunathan, "T-mars: Improving visual representations by circumventing text feature learning," 2023.

[24] A. E. Elo, *The Rating of Chessplayers, Past and Present*. New York: Arco Pub., 1978. [Online]. Available: http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216

[25] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project, Tech. Rep., 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768

[26] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," in *Journal of the ACM (JACM)*, vol. 46, no. 5. ACM, 1999, pp. 604–632.

[27] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019. [Online]. Available: https://github.com/Lightning-AI/lightning

[30] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[31] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, "OpenCLIP," Jul. 2021.

[32] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[33] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly

[34] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134