AEye: A Visualization Tool for Image Datasets

Florian Grötschla Luca A. Lanzendörfer Marco Calzavara Roger Wattenhofer

ETH Zurich, Switzerland
{fgroetschla, lanzendoerfer, mcalzavara, wattenhofer}@ethz.ch

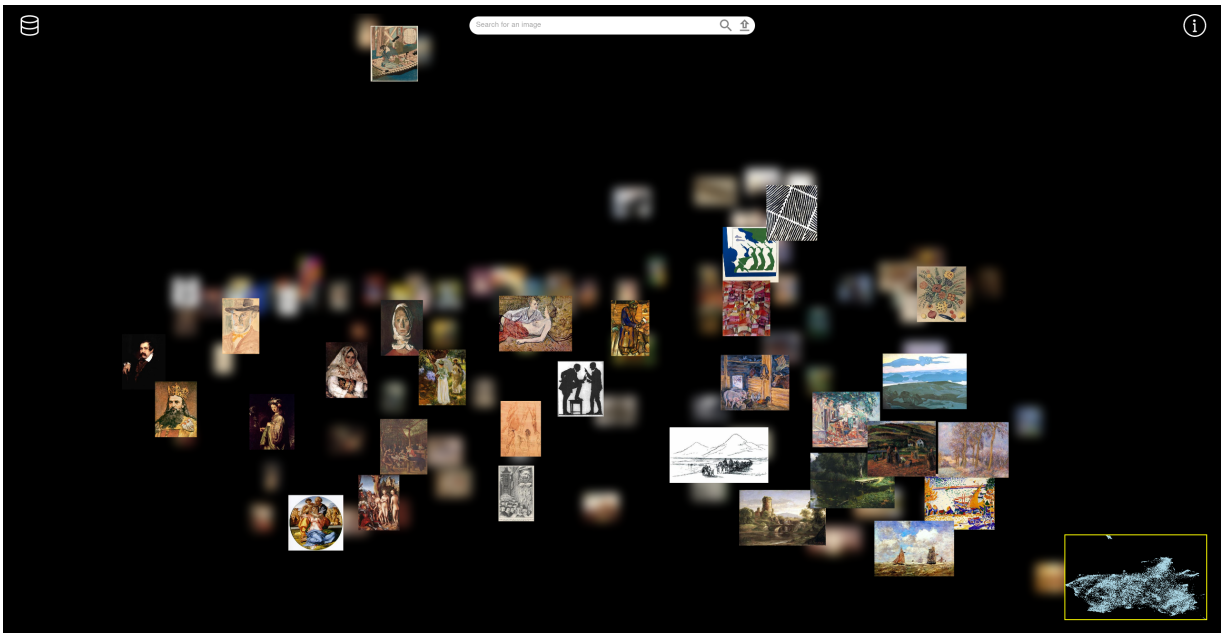

Figure 1: Overview of the AEye interface. Images are positioned according to their location in the CLIP embedding space and arranged in layers that the user can navigate by zooming. **Top left:** Dataset selector, **Top middle:** Search bar for semantic text and image search. **Top right:** Show information about the application. **Bottom right:** Minimap of the embedding space.

Abstract

Image datasets serve as the foundation for machine learning models in computer vision, significantly influencing model capabilities, performance, and biases alongside architectural considerations. Therefore, understanding the composition and distribution of these datasets has become increasingly crucial. To address the need for intuitive exploration of these datasets, we propose AEye, an extensible and scalable visualization tool tailored to image datasets. AEye utilizes a contrastively trained model to embed images into semantically meaningful high-dimensional representations, facilitating data clustering and organization. To visualize the high-dimensional representations, we project them onto a two-dimensional plane and arrange images in layers so users can seamlessly navigate and explore them interactively. AEye facilitates semantic search functionalities for both text and image queries, enabling users to search for content. We open-source the codebase for AEye, and provide a simple configuration to add datasets.

Index Terms: Image embeddings, image visualization, contrastive learning, semantic search.

1 Introduction

In today's data-driven landscape, the role of data in shaping the performance of artificial intelligence (AI) applications cannot be overstated. The quality, quantity, and complexity of the data significantly affect the performance and reliability of machine learning models across various domains. As datasets continue to grow in size, researchers and practitioners face challenges in understanding and extracting meaningful insights, such as identifying patterns or outliers in the datasets. Traditional methods of data analysis often fall short when analyzing large-scale image datasets, highlighting the need for novel approaches to data exploration and visualization.

Effectively visualizing large-scale image datasets requires approaches that can distill visual information into semantically meaningful representations, enabling users to uncover patterns, trends, and anomalies within the data. In response to these challenges, we introduce AEye – a novel approach to visualizing image datasets. AEye leverages recent advancements in AI, specifically contrastive learning techniques, to embed semantic information into high-dimensional image representations. By projecting these representations onto a two-dimensional plane, AEye facilitates the visualization of image datasets in a manner that aligns with human perception and intuition.

We present the design and implementation of AEye and demonstrate its effectiveness in visualizing large image datasets. Through a series of demonstrative use cases, we illustrate how AEye enables researchers and practitioners to gain deeper insights into image data, uncover hidden patterns, and facilitate informed decision-making. By providing AI-guided visualization, AEye offers a practical solution for visualizing large-scale image datasets and lets researchers and laymen extract insights from their data.

AEye is available at `aeye.ethz.ch`.

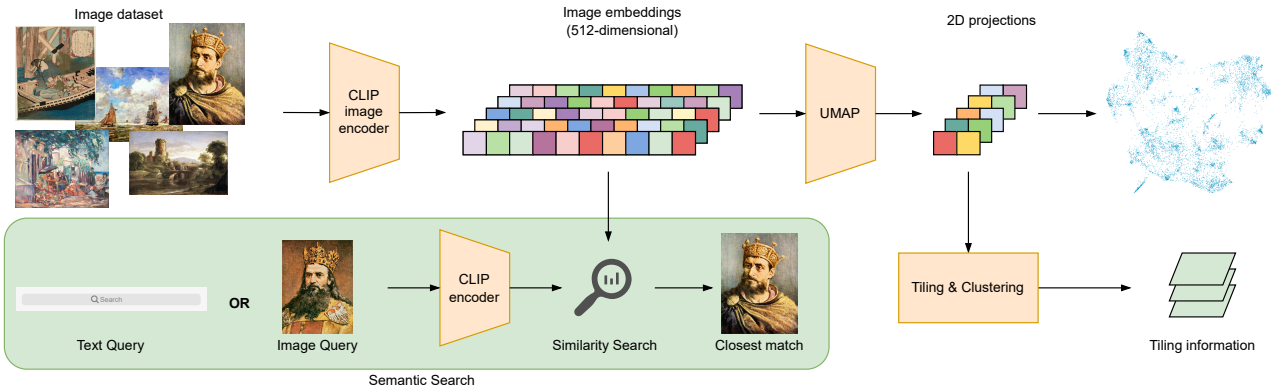

Figure 2: Architecture overview of AEye. Images are embedded with CLIP, stored in a vector database, and projected to a two-dimensional space with a UMAP projection (Section 3.1). The resulting positions are used for the visualization and a tiling and clustering module that computes representatives for each layer (Section 3.2). The semantic search takes text or image queries and uses CLIP to encode them. The vector database is used to find the nearest neighbor in the embedding space (Section 3.3).

2 Related Work

Large-scale data visualization is often facilitated by clustered and hierarchical representations [13, 12, 5]. Dimensionality reduction techniques such as Principal Component Analysis (PCA) [19], t-Distributed Stochastic Neighbor Embedding (t-SNE) [15] or Uniform Manifold Approximation and Projection (UMAP) [8] have previously been applied for data visualization [11, 1]. These techniques map high-dimensional data points to lower dimensions while clustering the data and making inherent patterns in the data more apparent. In our work, we choose UMAP for dimensionality reduction as it is more scalable than PCA and t-SNE while preserving local and global structures. Our work also touches upon the intuitive search for biases and imbalances in image datasets, which have previously been observed, and techniques were proposed to uncover and combat them [16, 4, 18].

To obtain high-dimensional image embeddings, we use Contrastive Language-Image Pretraining (CLIP) [9], which learns to understand images and text simultaneously by embedding them in a shared latent space. CLIP is trained on a diverse range of image-text pairs from the internet, enabling it to learn robust and generalized representations that capture the semantic content of images across a wide spectrum of concepts and categories. CLIP embeddings capture rich semantic information for images and text, enabling a wide range of tasks such as image classification, image retrieval, and text-to-image generation [10, 3] without task-specific supervision. We use the pretrained OpenAI CLIP model to embed all images. CLIP and other contrastive learning-based approaches have been used for image visualization before [2, 20], mostly as point cloud visualizations.

The visualization technique most closely aligned to ours is the Embedding Projector [14], which also visualizes embeddings generated by ML models with projected positions and offers a similar navigation technique consisting of zooming and panning. While it can also display images at the projected positions, it does not offer the layered visualization approach we provide.

AEye builds on these previous works and adds novel methods to show only a representative selection of images with a layered visualization style. Using contrastive learning methods, we can ensure that embeddings maintain semantic information and facilitate additional search features. Lastly, our approach scales to larger datasets in terms of the visualization itself, which always remains comprehensible with not too many images on screen, and computational resources, which directly benefit from the former.

3 AEye Application

AEye is a web-based application designed to facilitate the exploration and comprehension of large-scale image datasets. At its core, AEye leverages the CLIP (Contrastive Language-Image Pretraining) [9] embedding space to organize and visualize images in a two-dimensional plane. The positions of images within this embedding space are determined by their semantic similarity, allowing for intuitive navigation and exploration.

An overview of the AEye processing steps can be seen in Figure 2. In a data preprocessing stage, we compute CLIP embeddings for all images in the dataset, which are then stored in a vector database for fast nearest-neighbor lookups, which the semantic search relies on. The high-dimensional CLIP embeddings are then projected to two dimensions using the UMAP algorithm [8] to find spatial positions for all images. To accommodate the limited screen space and the large number of images in the dataset, AEye employs a layered visualization approach. Multiple layers are created that the user can navigate through. The last layer contains all images of the dataset at their projected positions, while the other layers only contain a selection of *representatives*. As the user zooms in, the view transitions from layer to layer while zooming in on a continuously smaller area of the layers, which lets us populate them more and more densely while limiting the number of images on screen at any time. The representatives are chosen as the centers of a clustering we achieve with a modified k-means clustering algorithm [7]. This ensures that each layer provides a condensed yet informative dataset view. The process is outlined in more detail in Section 3.2. Lastly, we can compute AI-generated captions with LLaVA [6] for all generated images in the preprocessing stage.

The visualization starts with a view of the first layer, which displays representative images from each cluster in the embedding space. As the user interacts with the visualization, they have the ability to zoom in on specific regions of interest, progressively revealing more detailed subsets of images from deeper layers. This interactive exploration enables users to uncover hidden patterns, clusters, and relationships within the dataset, empowering them to gain insights into the underlying structure of the data.

3.1 CLIP Embeddings and UMap Projection

While CLIP embeddings effectively preserve the semantic meaning of images by encoding rich semantic information learned during pretraining [10], their high dimensionality, typically 512 dimensions, poses challenges for direct visualization. The sheer number of dimensions hinders intuitive interpretation and exploration of the

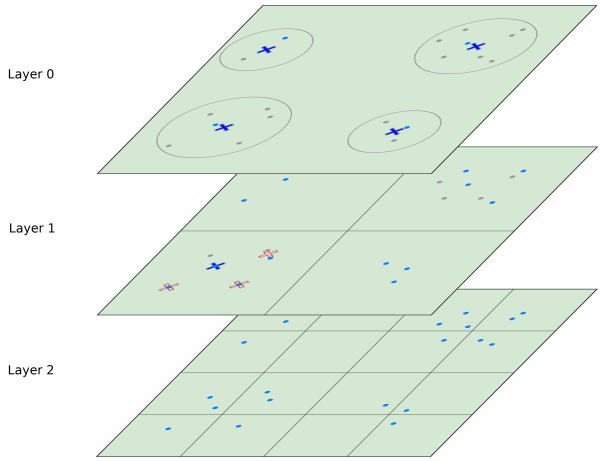

Figure 3: Visualization of the tiling hierarchy. Representatives for level 0 (blue dots) are obtained by clustering all points with k-means. Only the points closest to the centroids (crosses) are retained. In the next level, a k-means clustering is computed on every sub-tile, with the restriction of fixed centroids for the positions of representatives from the previous layers. Again, the closest points to the centroids are retained. This process finishes when all points can be kept for one level. Pseudocode can be found in Algorithm 1.

embedding space. Therefore, we employ dimensionality reduction techniques to project them onto a two-dimensional plane to facilitate the visualization based on CLIP embeddings. This transformation enables us to represent the complex semantic relationships encoded in the embeddings in a more compact format. Among various dimensionality reduction techniques, we select UMAP due to its ability to preserve the data's local and global structure. Unlike traditional methods like PCA, which primarily focus on preserving variance, UMAP aims to capture the underlying manifold structure of the data, ensuring that nearby points in the high-dimensional space remain close together in the low-dimensional projection. An example of the projected embedding spaces can be seen on the minimap on the bottom right of Figure 1, the right side of Figure 2, as well as the bottom right of Figure 4. Projections like these are a common technique for the visualization of high-dimensional data [15, 14] as the resulting point clouds usually exhibit a nicely clustered view of the embedding space. We use the projected embeddings as image positions throughout the application. As we cannot always show all images on screen, we further develop methods to select which images to show on the current layer through a clustering-based approach.

3.2 Choosing Representative Images

Large-scale image datasets contain an overwhelming number of images, necessitating a strategic approach to presenting a subset of images to users. We adopt a hierarchical strategy comprising multiple layers to address this challenge. In the initial layer, users encounter a limited set of representative images. By zooming in, users can traverse through these layers, progressively revealing additional images until the final layer displays all images. This hierarchical approach effectively manages the number of images on screen, resulting in a comprehensible visualization. The selection of representative images is guided by several criteria: they should provide a coarse-grained overview of the embedding space, maintain sufficient spacing between each other to avoid too many overlapping images, and reflect the characteristics of images within their respective area. Moreover, continuity in representation across layers is ensured by maintaining representatives from previous layers for all following ones. By doing so, images presented to the user in a previous layer do not disappear when zooming in but instead stay in place. We achieve these objectives through a clustering-based approach and tiling of the projected embedding space.

Algorithm 1 Computation of Representatives

Require: Image Dataset I, threshold k
  $E(i) \leftarrow \text{CLIP}(i) \quad \forall\, i \in I$
  $\text{pos} : i \mapsto \mathbb{R}^2 \leftarrow \text{UMAP}(E)$
  $d \leftarrow$ number of layers such that tiles on the last layer contain at most k images.
  for $l \leftarrow 0$ to $d-1$ **do**
    for tile t in T_l **do** ▷ $T_l \sim$ all tiles in layer l
      $R_{\text{prev}} \leftarrow \{r \in R_{t'} \mid t' \in T_{l'}, l' < l \text{ and } r \in A(t)\}$ ▷ $A(t) \sim$ area covered by tile t
      $\text{centers}_{\text{fixed}} \leftarrow \{\text{pos}(r) \mid r \in R_{\text{prev}}\}$
      $I_t \leftarrow \{i \in I \mid \text{pos}(i) \in A(t)\}$ ▷ images in current tile
      $\text{centers} \leftarrow \text{k-mean}(I_t, \text{centers}_{\text{fixed}}, k)$ ▷ k-mean with fixed centers
      $R_t \leftarrow \{\arg\min_{i \in I_t} (\text{dist}(\text{pos}(i), c)) \mid c \in \text{centers}\}$
    end for
  end for

Figure 3 shows an overview of the proposed approach. Each layer consists of a regular grid of tiles, with the side length halving from one layer to the next. Within each tile, a fixed predefined number k of images serves as representatives, approximately corresponding to the number of images visible on screen at any time. Selecting representatives is done by traversing layers from top to bottom and applying a k-means clustering algorithm with k centers to each tile. Representatives are then chosen as the images closest to the cluster centers to ensure that they reflect the underlying structure of the embedding space. To maintain consistency in representatives across layers, we modify the k-mean algorithm by retaining the positions of representatives from previous layers as fixed centroids throughout the algorithm's execution. A detailed algorithm description in pseudocode is presented in Algorithm 1. When transitioning from layer to layer, the viewport's size is scaled proportionally with the size of the tiles, meaning that the viewport approximately covers the same number of tiles in every layer. As we limit the number of representative images per tile by k, the number of images on screen at the same time does not get too large.

This approach yields meaningful representatives, maintains scalability, and improves performance in the web application. While the computation of the k-mean algorithm poses a significant computational burden, especially in the first layer, subsequent layers benefit from its application to smaller subsets, albeit more numerous. These computations can be efficiently parallelized, and the number of required layers grows insignificantly with larger dataset sizes. Our demonstration webpage accommodates image datasets exceeding 100k images, where the preprocessing for the tiling and clustering took about as long as the generation of CLIP embeddings. This was on commodity hardware and took only few hours, even on the biggest datasets. As the number of layers is expected to scale logarithmically with the number of images, there will never be a need to transition through too many zoom layers. Additionally, the generated tiling facilitates efficient data loading for the front end. As users navigate through layers, each tile consistently occupies a proportional screen space when in focus. These tiles serve as subdivisions, simplifying the selective loading of necessary data for visualization. The front end can request data for specific tiles, optimizing resource utilization and enhancing user experience.

3.3 Semantic Search and AI Captions

Beyond exploring visualized data through spatial navigation, AEye features semantic search functionality for text queries and images. Semantic search leverages the rich semantic representations encoded by the CLIP model for text and images, enabling users to retrieve relevant content easily, for example, with natural language.

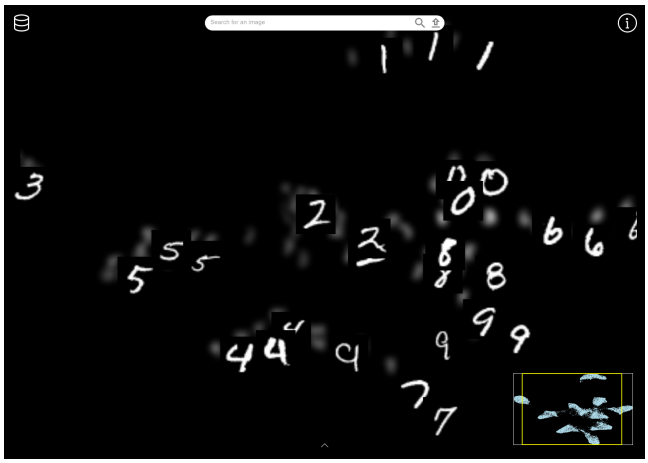

Figure 4: AEye view of the MNIST dataset. We observe that numbers are clearly separated by the projected CLIP embeddings, resulting in a meaningful clustering of the dataset. Similarly, the CelebA-HQ dataset shows a clear distinction between men and women.

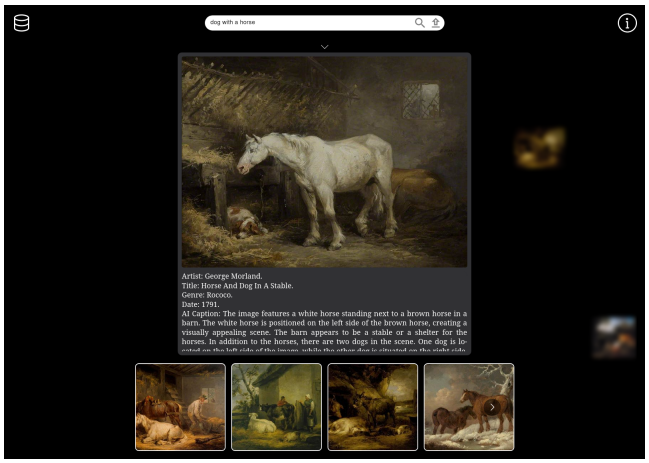

Figure 5: View of the application when searching for "a dog with a horse." The nearest neighbors in the embedding space are presented below the search result. In addition to metadata provided by the dataset, an AI-generated caption of the image is shown.

For text queries, the semantic search first embeds the user-provided text query using the CLIP text encoder. Similarly, the CLIP image encoder embeds the user-provided image for image queries. To search for the nearest neighbor (with regard to cosine similarity), we store all image embeddings that we generate in the preprocessing stage in a vector database, more specifically, milvus [17]. Vector databases are specifically designed to handle high-dimensional vector data and offer fast and scalable search capabilities. Therefore, when querying the database, we can use existing datastructures created in the preprocessing stage and reply quickly with the nearest neighbors sorted by similarity.

In addition to semantic search, AEye can also provide AI-generated image captions by running a captioning model on all images in the preprocessing phase. We use LLaVA [6] for this purpose, as it resulted in the most accurate and descriptive captions in our testing, but other models can easily be integrated and used. The captions provide valuable context and insights into the content of the images, enriching the user experience and providing feedback on how the AI model "sees" an image. Further, the captioning model is interchangeable, and the user can inspect the quality of captions for the used model.

3.4 Interface Design

The user is initially presented with a few of the top layer of the embedding space and a search bar for the semantic text and image search in the center. If the user decides to submit a search query, the view zooms in to the closest match in the embedding space and shows the view depicted in Figure 5, with an example for the MNIST dataset in Figure 4. For MNIST, we can clearly observe a semantic clustering of numbers in the projected embedding space, which lets us infer emerging patterns from the overview.

Users can further navigate the layered embedding space by zooming and moving the viewport around. Blurred previews are shown in the background to give a sense of the images in the next layer. Clicking on an image results in the same view as provided by the search, with more information provided by the dataset and an AI-generated caption of the image, as well as the closest neighbors in the CLIP embedding space. To stay oriented, a minimap with an overview of the whole embedding space is provided in the bottom right, as visible in Figure 1. Datasets can be selected on the top left, and an information button on the top right shows a small explanation for the application.

3.5 Case Study

To demonstrate a possible use case for AEye, we consider a hypothetical machine-learning practitioner working with the Common Objects in Context (COCO) 2017 dataset. The practitioner aims to improve their object detection model by first understanding the data. The COCO 2017 dataset contains 163,000 images, making it impractical to visualize all images at once or manually sift through them. This highlights the need for a comprehensive tool. The practitioner uses AEye to facilitate this process.

After the preprocessing, AEye produces its interactive visualization of the dataset on a 2D plane, clustering similar images together. This allows the practitioner to observe distinct clusters corresponding to different object categories, such as "person," "vehicle," and "animal," revealing the distribution of categories and identifying under- or overrepresented ones. Several outliers were also detected, which, upon further examination, revealed labeling errors and unusual object combinations that could impact model performance. The insights gained from AEye's visualization enable the practitioner to make informed decisions about the dataset. They identify underrepresented categories needing augmentation and corrected labeling anomalies, leading to a more balanced and accurate dataset for training their object detection model.

4 Conclusion

AEye offers a comprehensive solution for visualizing large-scale image datasets, leveraging contrastively trained embedding models for semantically rich representations. By incorporating hierarchical tiling, clustered subspaces, and semantic search features alongside AI-generated captions, AEye facilitates intuitive navigation and exploration of diverse image collections. The scalability and extensibility of AEye enable researchers and other users to explore various datasets, from machine learning to general art collections. With AEye, unlocking insights and uncovering patterns within large-scale image datasets becomes both accessible and insightful. A demonstration website with a selection of datasets, [1] as well as the source code, can be found online. [2]

[1] aeye.ethz.ch
[2] https://github.com/ETH-DISCO/aeye

REFERENCES

[1] S. Afzal, S. Ghani, M. M. Hittawe, S. F. Rashid, O. M. Knio, M. Hadwiger, and I. Hoteit. Visualization and visual analytics approaches for image and video datasets: A survey. *ACM Transactions on Interactive Intelligent Systems*, 13(1):1–41, 2023. doi: 10.1145/3576935 2

[2] N. Böhm, P. Berens, and D. Kobak. Unsupervised visualization of image datasets using contrastive learning. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[3] C. Che, Q. Lin, X. Zhao, J. Huang, and L. Yu. Enhancing multimodal understanding with clip-based image-to-text transformation. In *Proceedings of the 2023 6th International Conference on Big Data Technologies*, pp. 414–418, 2023. doi: 10.1145/3627377.36274 2

[4] S. Fabbrizzi, S. Papadopoulos, E. Ntoutsi, and I. Kompatsiaris. A survey on bias in visual datasets. *Computer Vision and Image Understanding*, 223:103552, 2022. doi: 10.1016/j.cviu.2022.103552 2

[5] P. Klemm, S. Oeltze-Jafra, K. Lawonn, K. Hegenscheid, H. Völzke, and B. Preim. Interactive visual analysis of image-centric cohort study data. *IEEE transactions on visualization and computer graphics*, 20(12):1673–1682, 2014. doi: 10.1109/TVCG.2014.2346591 2

[6] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024. 2, 4

[7] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489 2

[8] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 2

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021. 2

[10] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021. 2

[11] S. Raval, C. Wang, F. Viégas, and M. Wattenberg. Explain-and-test: An interactive machine learning framework for exploring text embeddings. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pp. 216–220, 2023. doi: 10.1109/VIS54172.2023.00052 2

[12] J. Schmidt, M. E. Gröller, and S. Bruckner. Vaico: Visual analysis for image comparison. *IEEE transactions on visualization and computer graphics*, 19(12):2090–2099, 2013. doi: 10.1109/TVCG.2013.213 2

[13] T. Schultz and G. L. Kindlmann. Open-box spectral clustering: applications to medical image analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2100–2108, 2013. doi: 10.1109/TVCG.2013.181 2

[14] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016. 2, 3

[15] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 2, 3

[16] A. Wang, A. Liu, R. Zhang, A. Kleiman, L. Kim, D. Zhao, I. Shirai, A. Narayanan, and O. Russakovsky. Revise: A tool for measuring and mitigating bias in visual datasets. *International Journal of Computer Vision*, 130(7):1790–1810, 2022. doi: 10.1007/s11263-022-01625-5 2

[17] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2614–2627, 2021. 4

[18] T. Wang, J. Zhao, M. Yatskar, K.-W. Chang, and V. Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. doi: 10.1109/ICCV.2019.00541 2

[19] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. doi: 10.1016/0169-7439(87)80084-9 2

[20] Y. Ye, R. Huang, and W. Zeng. Visatlas: An image-based exploration and query system for large visualization collections via neural image embedding. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3224–3240, 2024. doi: 10.1109/TVCG.2022.3229023 2