Image Embedding Explorer: Opening the Black Box of Deep Learning Models

Rixin Chen u7882614 Xi Ding u7751035 Jinqian Jiang u7774680 Xingchen Zhang u7670173

Group Contribution Statement

All team members contribute equally to this project.

Abstract

This paper presents the Image Embedding Explorer, an interactive tool designed to open the "black box" of deep learning models by visualizing and analyzing image embeddings across network layers. Focusing on CLIP and DINOv2 models, we examine how features evolve from layer to layer by projecting high-dimensional embeddings into lower-dimensional spaces using projection pursuit techniques such as PCA, t-SNE, UMAP, and Grand Tour. Our evaluation applies KNN accuracy on the labeled CIFAR-10 dataset and Silhouette Scores on the CIFAR-10 and CelebA datasets, revealing different performances of the CLIP and DINOv2. Through deeper analysis, we can understand the model's focus areas and weaknesses when extracting embeddings. Our findings also demonstrate the potential of non-linear and self-supervised approaches in capturing nuanced image features, advancing interpretability in deep learning. Through the Image Embedding Explorer, users gain an intuitive way to visualize and understand complex model embeddings, which enhances transparency in model decision-making.

1 Introduction

1.1 Background and Motivation

Deep learning has revolutionized the field of machine learning, achieving remarkable performance in computer vision tasks. Models like Contrastive Language–Image Pre-training [11] (CLIP) and Self-Distillation with No Labels [12] (DINOv2) have been demonstrated as powerful tools for learning rich visual representations from images. However, despite their success, deep learning models are often criticized for being "black boxes" due to their complex architectures and high-dimensional parameter spaces. Understanding the internal workings of these models remains a significant challenge, limiting trust, transparency, and further advancement in the field.

The interpretability of deep learning models is very important for improving model performance, understanding model decision-making processes, and evaluating effectiveness across different datasets. The details are as follows:

- **Model Improvement and Decision Understanding**: Interpreting models aids in identifying and enhancing them by revealing what features they focus on and how they make decisions. By examining the embeddings produced at different layers within a deep learning model, researchers can gain insights into the hierarchical structure of learned features. Layer-wise analysis reveals the incremental learning stages that ultimately contribute to the model's final predictions, offering a glimpse into the "black box" nature of deep models.
- **Performance Evaluation Across Datasets**: Interpretability enhances the ability to assess model performance across different datasets. Analyzing which parts of a dataset a model classifies well and where it struggles can clarify the strengths and weaknesses of the model. This understanding allows for targeted adjustments, such as data augmentation or architecture modifications, to address the model's weak points.

To improve the interpretability of models, it is necessary to explore techniques that allow us to visualize and analyze the internal representations and decisions made by deep learning architectures. To this end, we developed the **Image Embedding Explorer**, an interactive tool designed to help users gain insight into models through visualized embeddings. This tool provides a user-friendly interface for exploring embeddings across different model layers, allowing users to observe how feature abstraction evolves progressively. Additionally, the Explorer enables users to examine the relationships between images and their corresponding embeddings, uncover clustering patterns, and identify the features that the model leverages to distinguish between classes.

1.2 Contributions

The contributions of this paper can be summarized as follows:

1. **Layer-Wise Embedding Visualization**: Our tool supports embeddings from any layer of models like CLIP and DINO, enabling detailed analysis of hierarchical feature representations and their evolution within the network.
2. **Multiple Projection Pursuit options**: We offer techniques such as PCA [9], UMAP [10], t-SNE [15], and Grand Tour [1], allowing users to choose the most suitable approach for different data structures and analysis goals.
3. **Interactive 3D Exploration**: An interactive interface lets users navigate embeddings in 3D, with options to rotate, zoom, and examine individual data points, linking embeddings directly to the visual data.
4. **Improved Model Interpretability**: Our tool provides insights into model workings by revealing patterns and clusters in embeddings, supporting tasks like model evaluation and debugging.

2 Related Work

Image Embeddings Extraction. Advancements in deep learning have significantly enhanced models' ability to generate information-rich image embeddings. Early models, such as AlexNet [7] and VGG [14], laid the foundation by capturing visual features through convolutional layers, though they were limited in their versatility and required extensive labeled data.

In recent years, general-purpose models like CLIP [11] and DINOv2 [12] have gained popularity for their robust and flexible feature extraction capabilities. CLIP employs a contrastive learning approach with dual encoders for images and text, enabling it to perform tasks like zero-shot classification and cross-modal matching through joint representations. DINOv2, a self-supervised model utilizing a teacher-student architecture, is particularly effective for transfer learning in settings with minimal annotated data. Both models provide information-rich embeddings at multiple layers, allowing more detailed analysis on feature evolution across the model's hierarchy and gain deeper insights into the learning process.

Visualization. Image embeddings are challenging to interpret due to their high dimensionality. Over the years, various visualization methods have emerged to address the challenges of representing high-dimensional data in an interpretable manner. Among the foundational techniques is Principal Component Analysis (PCA) [9], a linear projection method that identifies principal components to capture maximum variance in data. PCA is effective for an initial overview but often lacks the flexibility to reveal intricate patterns in complex datasets, leading to the need for more advanced methods.

Recently popular non-linear methods like t-SNE [15] and UMAP [10] often produce visualizations that are good at preserving neighborhood relationships and capturing cluster structures. These methods use optimization techniques to enhance cluster separation in low-dimensional space according to a specified metric, resulting in visualizations with clearer and more distinct groupings. They are especially useful for demonstrating a model's ability to distinguish between classes and for identifying potentially confusable groups, which can help uncover weaknesses in the model. However, the static nature resulting from their optimization process limits their potential for interactive exploration.

Conversely, dynamic methods like the Grand Tour [1] and Manual Tour [3], part of the 'tours' family, enable continuous data exploration through rotating projections, uncovering underlying structures from multiple perspectives. By distributing information across various dimensions over time rather than collapsing it into a single static projection, these methods offer a more direct and unfiltered representation of complex patterns. Among dynamic projection methods, the Grand Tour is frequently implemented for its simplicity and flexibility, making it easily adaptable to other tour methods. However, our experiments indicate that, despite modifications, dynamic tours generally do not outperform PCA in terms of cluster clarity due to their lack of optimization. In our Image Embeddings Explorer, we integrate four different visualization methods into one tool, providing users with a versatile, hybrid approach for comprehensive data exploration.

Metrics. We evaluate our tool using two metrics: the accuracy of KNN classifiers on labeled datasets and a custom metric, the Silhouette Score of clusters obtained by K-Means on unlabeled datasets. For the KNN classifier accuracy, commonly used in related works on dimensionality reduction for labeled datasets [10] , we first apply a k-Nearest Neighbors (KNN) algorithm to classify each data point in lower-dimensional space. The accuracy is then calculated by comparing the predicted label of each data point (based on the majority label of its nearest neighbors) to its true label, as follows:

$$\text{KNN Accuracy} = \frac{\text{Correctly predicted data points}}{\text{Total data points}}$$

Additionally, by using the same set of high-dimensional data to compute KNN accuracy as a benchmark, we can evaluate how effectively different visualization methods preserve the structural integrity of high-dimensional embeddings in their low-dimensional representations.

Previous studies have not evaluated dimensionality reduction methods on unlabeled datasets. In our approach, we adapt the Silhouette Score [13], an well-known metrics, for use with unlabeled datasets by first applying K-Means clustering to the data, followed by calculating the Silhouette Score. This adapted approach assesses clustering quality by measuring how well each data point aligns with its assigned cluster relative to other clusters:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the average distance between point i and other points in the same cluster, and $b(i)$ is the average distance between point i and points in the nearest neighboring cluster.

3 Methodology

3.1 Preliminary

This section provides an overview of the embeddings and projection pursuit methods we use.

3.1.1 Image Embeddings

In deep learning, an image embedding is a vector representation, $\mathbf{z} \in \mathbb{R}^d$, where d is the embedding dimensionality that captures high-level features of an image. For an input image I, a model f_θ parameterized by θ (weights and biases of the model) transforms I through multiple layers, yielding an embedding:

$$\mathbf{z} = f_\theta(I) \in \mathbb{R}^d.$$

Models like CLIP and DINO are designed to optimize the embeddings such that semantically similar images lie closer in this high-dimensional space, forming clusters of similar representations. The embeddings are typically extracted from specific model layers, allowing researchers to analyze feature evolution through the network's hierarchy. The resulting embeddings, however, are often high-dimensional ($d \gg 3$), making direct visualization infeasible without further dimensionality reduction.

3.1.2 Projection pursuit and visualization

Mapping high-dimensional embeddings into 3D-dimensional spaces requires projection pursuit techniques that preserve essential structures, such as clusters or relationships, within the data. Let $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\} \subset \mathbb{R}^d$ be a set of n embeddings obtained from a dataset. Dimensionality reduction can be formulated as finding a function $g : \mathbb{R}^d \to \mathbb{R}^k$ (where $k \ll d$) that maps each high-dimensional embedding $\mathbf{z}_i$ to a lower-dimensional representation $\mathbf{y}_i = g(\mathbf{z}_i) \in \mathbb{R}^k$, ideally preserving the neighborhood relationships in the high-dimensional space.

Several common methods exist for g, each with different mathematical properties:

Principal Component Analysis (PCA): PCA is a linear transformation that projects the data onto a k-dimensional subspace by maximizing variance. The projection matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ is determined by the top k eigenvectors of the covariance matrix $\mathbf{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})^T$. The embeddings are then reduced as follows:

$$\mathbf{Y} = \mathbf{Z} \cdot \mathbf{W},$$

where $\mathbf{Y} \in \mathbb{R}^{n \times k}$ represents the reduced embeddings. While efficient, PCA may fail to capture non-linear structures present in complex data like images.

t-SNE and UMAP: t-SNE and UMAP are non-linear methods better suited for complex data. t-SNE minimizes the Kullback–Leibler divergence between the probability distributions of point-pairs in the high-dimensional and low-dimensional spaces, aiming to preserve local structure. UMAP, on the other hand, constructs a fuzzy topological representation in high-dimensional space and attempts to optimize a lower-dimensional equivalent.

Grand Tour: It enables dynamic exploration by continuously rotating data, creating interpolations between random linear projections. To enable dynamic exploration, we apply continuous rotations to the data and interpolate between different random projections. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ represent the data matrix, where n is the number of data points and d is the dimensionality, $\mathbf{R}_t \in \mathbb{R}^{d \times d}$ represent a rotation matrix at time t, and $\mathbf{P}_1, \mathbf{P}_2 \in \mathbb{R}^{d \times k}$ represent two random linear projection matrices, where $k < d$. To achieve continuous rotation and interpolation, we rotate the data using $\mathbf{R}_t$ and interpolate between projections $\mathbf{P}_1$ and $\mathbf{P}_2$ using an interpolation parameter $\alpha \in [0, 1]$:

$$\mathbf{X}_{t,\alpha} = \mathbf{X}\mathbf{R}_t\left((1 - \alpha)\mathbf{P}_1 + \alpha\mathbf{P}_2\right)$$

Projection Pursuit Tour: We also explored an innovative approach to implement a projection pursuit tour by using KNN accuracy as the guiding criterion. We trained KNN classifiers on low-dimensional data projected from randomly selected angles, identifying the projection angle with the highest KNN accuracy and rotating around this direction to explore nearby projections. While this approach aimed to identify directions with improved clustering, it ultimately did not yield significant visual improvements. Thus, we excluded it from the final tool, opting instead to retain only the Grand Tour to provide users with maximum flexibility for exploratory visualization.

3.2 Overall Framework

As illustrated in Figure 1, the tool's workflow consists of three main stages:

1. **Embedding Extraction:** The tool allows users to input embeddings generated by models at different layers of processing. For this project, we utilize the CLIP and DINO models, focusing on embeddings generated from purely image-based datasets (CIFAR-10 [6] and CelebA [8]). Users can choose embeddings either from the final output layer or from intermediate layers, facilitating a deeper exploration into the model's representational structure.
2. **Projection Pursuit:** Once embeddings are obtained, we apply projection pursuit techniques to map them into a 3D space for easier visualization. Supported methods include PCA, t-SNE, UMAP, and Grand Tour. Each method provides unique insights into the structure and separability of the embeddings, helping to "open the black box" of how models like CLIP and DINO encode image data.
3. **Interactive Exploration:** In the 3D visualization space, each point represents an individual image from the dataset. Users can interact with these points, and upon selection, the tool displays the original image corresponding to the embedding. This interactive functionality enables users to explore and understand the relationships between images as learned by the model, bridging the gap between abstract embedding vectors and their visual representations.

This framework provides a systematic approach for inspecting how different models and their layers represent visual data, facilitating a more intuitive understanding of deep learning embeddings.

4 Experiments

4.1 Experiment setups

4.1.1 User Interface

The front end of our tool is developed using Dash and Plotly [5], providing a responsive web-based platform for exploring high-dimensional embeddings. As shown in Figure 2, users can select the dataset (e.g., CIFAR-10 or CelebA) and the embedding model (e.g., CLIP, DINOv2). A layer slider allows navigation across different network blocks (e.g., from Block 1 to Final Layer), enabling users to examine how embeddings evolve through the network layers.

The main visualization area features a 3D UMAP projection as the default view, along with options to switch between PCA, t-SNE, and Grand Tour projections. These plots are dynamically updated based on user selections, allowing for real-time comparisons of different dimensionality reduction techniques.

Our embedding files, organized by dataset, model and layer, are dynamically loaded based on user selections. The back end processes the selected embeddings using PCA, t-SNE, UMAP, or Grand Tour. Each method is implemented to balance efficiency with quality of projection, leveraging pre-computed embeddings for faster load times.

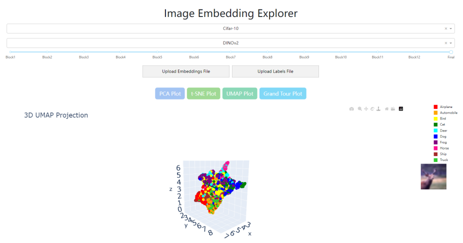

Figure 2: User Interface [1]

4.1.2 Data Processing Pipeline

Images are processed through pre-trained Vision Transformer models for both CLIP (ViT-B/32) and DINOv2 (ViT-S/14). Before being fed into the models, images undergo a series of preprocessing steps to ensure compatibility with the model input size, as both models require images of a specific dimension, typically 224x224 pixels. For datasets with varying image sizes, images are first resized and center-cropped to maintain consistency. Specifically, each image is resized to 256x256 pixels, then center-cropped to 224x224 pixels. This approach preserves the central part of the image while adjusting it to the required input dimensions.

In each model, hooks are registered on every Transformer block within the visual encoder to capture intermediate [CLS] token embeddings, allowing us to observe the evolution of features across layers. The final layer embedding, which represents the complete image encoding, is also captured. All embeddings are organized by layer and dataset and saved as .npy files, providing a hierarchical view of feature abstraction across network depths for both CLIP and DINOv2.

[1] For a larger picture of the user interface, please refer to the appendix A1.

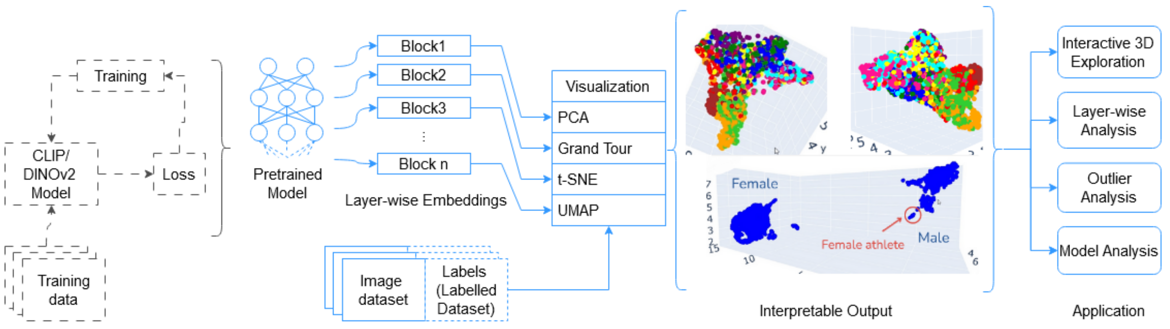

Figure 1: Workflow

4.1.3 Metrics for Evaluation

As noted earlier, we evaluated the quality of embeddings using two primary metrics: Silhouette Score from K-Means clustering on the unlabeled CelebA dataset and KNN accuracy on the labeled CIFAR-10 dataset. These metrics were computed for each projection method (PCA, t-SNE, UMAP) across multiple network layers (blocks) of both the CLIP and DINOv2 models on the CIFAR-10 and CelebA datasets. We excluded the Grand Tour method because it's designed for interactive exploration rather than providing the fixed, reproducible projections needed for quantitative embedding evaluation with our selected metrics. The high-dimensional embeddings (High-Dim) serve as a baseline for comparing the effectiveness of projection methods.

4.2 Results

We report both final and layer-wise results to examine each layer's contribution to embedding quality and model performance. Both the CLIP and DINOv2 pretrained models used in this study have 12 Transformer encoder blocks, from which we extracted embeddings, including the final output.

To obtain optimal metrics, we fine-tune the number of clusters K to report the highest Silhouette Score for each projection method across embedding models and datasets, and similarly adjust the number of nearest neighbors K to report the highest KNN accuracy.

4.2.1 Results of K-Means Silhouette Score on the CelebA Dataset

Table 2 and Figure 3 4 shows the results of K-Means Silhouette Score on the CelebA Dataset. The Silhouette Scores on high-dimensional embeddings from both CLIP and DINOv2 are lower than those on low-dimensional embeddings due to the curse of dimensionality [2]. As the number of dimensions increases, Euclidean distance becomes less effective at measuring similarity between data points. After projection, these scores increase significantly, indicating that the projections successfully alleviate this issue and reveal the underlying cluster structure. A higher Silhouette Score indicates better-defined clusters, as discussed in the following sections.

4.2.2 Results of KNN accuracy on the CIFAR-10 Dataset

Table 1 and Figure 5 shows the results of K-Means Silhouette Score on the CelebA Dataset. While Euclidean distance is used in the KNN algorithm, the KNN accuracy on high-dimensional embeddings remains consistently higher than on their 3D projections. This is because KNN focuses on local structure, which tends to be preserved even in high-dimensional space [4].

After projection onto 3D space, the KNN accuracy drops, reflecting a loss of information in local relationships. A projected KNN accuracy closer to that in the original high-dimensional space indicates better class separation within the data.

	High-Dim	PCA	t-SNE	UMAP
DINOv2	0.7153	0.3392	0.5625	<u>0.6022</u>
CLIP	0.6566	0.3044	0.5400	<u>0.5429</u>

Table 1: KNN classifier accuracy on the CIFAR-10 dataset for both high-dimensional and projected embeddings of the final output.

	High-Dim	PCA	t-SNE	UMAP
DINOv2	0.0951	0.3394	0.3330	<u>0.3939</u>
CLIP	0.0489	0.394	0.3729	<u>0.4706</u>

Table 2: Silhouette Scores of K-Means clusters on the CelebA dataset for both high-dimensional and projected embeddings of the final output.

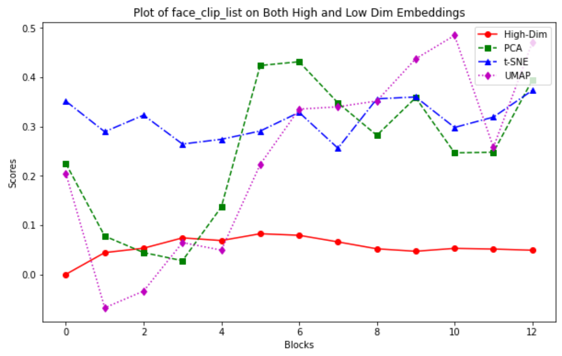

Figure 3: Silhouette Score of K-Means clusters on both high-dimensional and projected embeddings across various blocks of the CLIP model on the CelebA dataset.

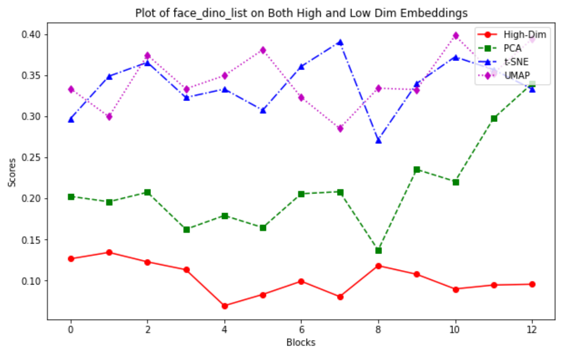

Figure 4: Silhouette Score of K-Means clusters on both high-dimensional and projected embeddings across various blocks of the DINOv2 model on the human face dataset.

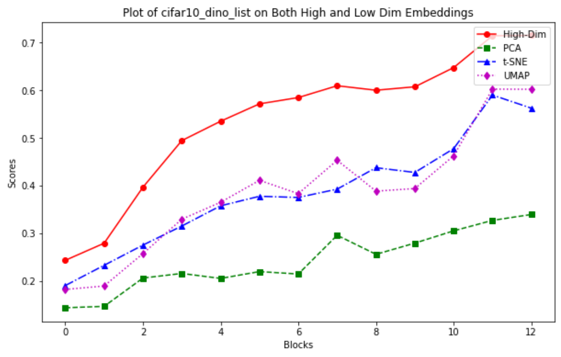

Figure 5: KNN accuracy on both high-dimensional and projected embeddings across various blocks of the DINOv2 model on the CIFAR-10 dataset.

5 Discussion

5.1 Findings

5.1.1 Model Evaluation

We provided identical image inputs to both models, enabling a direct comparison of their ability to generate meaningful embeddings from visual information alone.

Comparison on metrics. As seen in Table 12, DINOv2 demonstrated superior performance over CLIP in high-dimensional KNN accuracy, achieving a score of 0.7153 compared to CLIP's 0.6566. It indicates that DINOv2 is particularly effective in capturing local neighborhood structures within its high-dimensional embeddings. Given that DINOv2 was designed with self-supervised learning, it has a natural advantage in generating robust visual feature representations, which may contribute to its higher performance when only image data are provided. CLIP, on the other hand, typically leverages its contrastive language-image pre-training to build associations between modalities, yet when limited to visual input alone, its effectiveness in distinguishing fine-grained visual features appears constrained compared to DINOv2.

Silhouette scores for K-Means clustering further underscore the differences between CLIP and DINOv2 in handling label-free image data. In high-dimensional space, DINOv2 achieved a silhouette score of 0.0951, notably higher than CLIP's 0.0489, indicating that DINOv2's embeddings form more cohesive clusters even without labeled supervision. This aligns with DINOv2's architectural emphasis on self-distillation, allowing it to capture intrinsic data patterns and form coherent groupings based on visual features alone. Conversely, CLIP's contrastive learning approach, optimized for multimodal input, results in less-defined clustering with visual-only data, highlighting a potential limitation in its adaptability to single-modality scenarios.

Layer-wise Comparison. In Figure 3, we observe that the high-dimensional Silhouette Score peaks around block 5, indicating that CLIP's mid-layers capture well-separated facial features, likely due to its multi-modal training objective that aligns visual and textual features. However, this clustering quality declines in deeper layers, suggesting that CLIP's later stages prioritize semantic abstraction over precise cluster separability, possibly focusing more on aligning images with general concepts than distinguishing individual identities. Among the projections, t-SNE and UMAP outperform PCA, with t-SNE performing best in the mid-layers, highlighting that non-linear projections better preserve complex facial distinctions in lower dimensions for CLIP embeddings.

In contrast, Figure 3 and Figure 4 illustrate that DINOv2 maintains or improves Silhouette Score and KNN accuracy across layers, aligning with its self-supervised goal of hierarchical, coherent feature representation. Unlike CLIP, DINOv2's later layers retain strong clustering and classification performance, suggesting its training encourages intrinsic data structures, progressively refining class-specific details. UMAP scores improve significantly in DINOv2's deeper layers, particularly on CIFAR-10, where KNN accuracy peaks around block 10. This highlights DINOv2's capacity to capture well-defined class structures that remain informative for clustering and classification, even in low-dimensional projections.

5.1.2 Clustering Analysis

Distribution patterns. Our analysis shows that both CLIP and DINOv2 models generally perform well in separating images of different labels and categories within their generated image embeddings. For instance, in the CIFAR-10 dataset, not only were images with distinct labels separated into their own clusters, but conceptually similar categories—such as ship, automobile, and truck, all representing forms of transportation—were placed adjacent to each other in the visualization. And the animal classes clustered separately in another region.

A similar pattern emerged in the unlabeled face dataset, where different face categories naturally formed clusters. Notably, male and female images created two large and distinct clusters, showing the model's ability to capture significant categorical differences.

These findings suggest that the models capture broad patterns well. However, in the CIFAR-10 dataset, we observed some degree of overlap between images of different labels. One interpretation of this overlap is that certain classes in CIFAR-10 are inherently more similar to each other—such as "horse" and "deer," or "automobile" and "truck." This similarity, compounded by the lower resolution and limited detail of CIFAR-10 images, potentially increases ambiguity and makes these classes harder to distinguish, resulting in greater overlap in their representations.

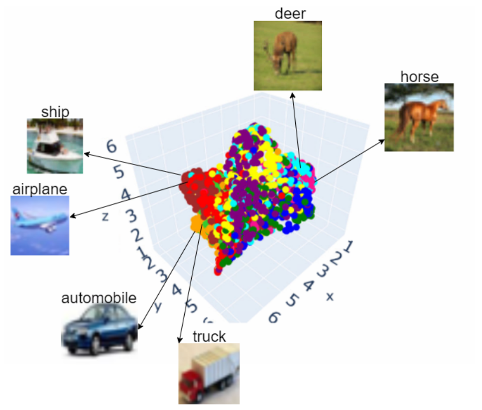

Figure 6: CIFAR10 dataset categorized clusters

Biases detection. In our experiments with unlabeled datasets, we observed that although images of male and female faces were generally well separated into two distinct clusters, there was a small subcluster within the male cluster consisting of images of female athletes. This indicates that the model may lack the capability to accurately recognize the characteristics of female athletes, instead categorizing them broadly as male due to their athletic attire.

This observation suggests a potential limitation in the model's ability to distinguish certain visual features, possibly associating sportswear more strongly with male characteristics. Given that this issue was present in both CLIP and DINO models, we infer that the phenomenon may be related to the training dataset rather than the architecture of the models themselves, suggesting an inherent bias in the pre-trained model used. To further explore this hypothesis, additional experiments could be conducted to confirm the influence of dataset bias on the model's performance.

5.1.3 Visualization Method Evaluation

In our experiments, we applied various dimension reduction techniques to evaluate their effectiveness in visualizing high-dimensional embeddings. The Figure 3, Figure 4, and Figure 5 illustrate the performance of these methods across different layers of CLIP and DINOv2, as measured by K-Means Silhouette Score and KNN Accuracy.

Non-linear Methods (UMAP and t-SNE): As shown in Figures3 and 4, UMAP and t-SNE consistently achieved high Silhouette Scores for projected embeddings, outperforming the linear method PCA. This indicates that non-linear methods are more effective in capturing and preserving cluster structures in the low-dimensional space, particularly in complex datasets like CelebA, where facial features vary widely. Furthermore, Figure5 shows that UMAP and t-SNE achieved higher KNN accuracy than PCA and Grand Tour, underscoring their ability to maintain local neighborhood structures in the projection.

Grand Tour and PCA: Although this method less effective in cluster separation compared to optimized static methods, provided a flexible, continuous view of the data. Grand Tour does not rely on optimization to create well-defined clusters, resulting in lower KNN accuracy (Figure 5) compared to UMAP and t-SNE. However, Grand Tour's dynamic projection offers an advantage for interactive data exploration, allowing users to observe the data from multiple perspectives. This aligns with related research suggesting that Grand Tour is suitable for unfiltered visualization but may fall short in clustering clarity due to its reliance on random, non-optimized projections. While PCA provided a quick, linear approach to visualizing embeddings, it struggled to capture the complex structures of datasets such as CelebA, as evidenced by lower Silhouette Scores and KNN accuracy in Figure 3, Figure 4. Its performance confirms the need for non-linear or dynamic methods when dealing with high-dimensional data that exhibits non-linear patterns.

6 Conclusion

In conclusion, the Image Embedding Explorer is a powerful tool for visualizing embeddings from deep learning models like CLIP and DINOv2, enhancing interpretability through diverse dimensionality reduction and projection pursuit methods. Its interactive features offer insights into model behavior, promoting transparency and deeper understanding. Future work could refine projection methods by exploring alternative projection pursuit tours beyond the KNN accuracy-guided approach.

References

[1] Daniel Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

[2] Richard E. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[3] Dianne Cook and Andreas Buja. Manual controls for high-dimensional data projections. *Journal of computational and Graphical Statistics*, 6(4):464–480, 1997.

[4] T. M. Cover and P. E. Hart. Nearest neighbor (nn) norms: Nn pattern classification techniques. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[5] Plotly Technologies Inc. Collaborative data science, 2015.

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009. Accessed: 2024-09-01.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[9] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[10] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[11] OpenAI. Clip: Contrastive language-image pretraining. `https://github.com/openai/CLIP`, 2021. Accessed: 2024-09-01.

[12] Meta AI Research. Dinov2: Learning robust visual features without supervision. `https://github.com/facebookresearch/dinov2`, 2024. Accessed: 2024-09-01.

[13] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

A Appendix

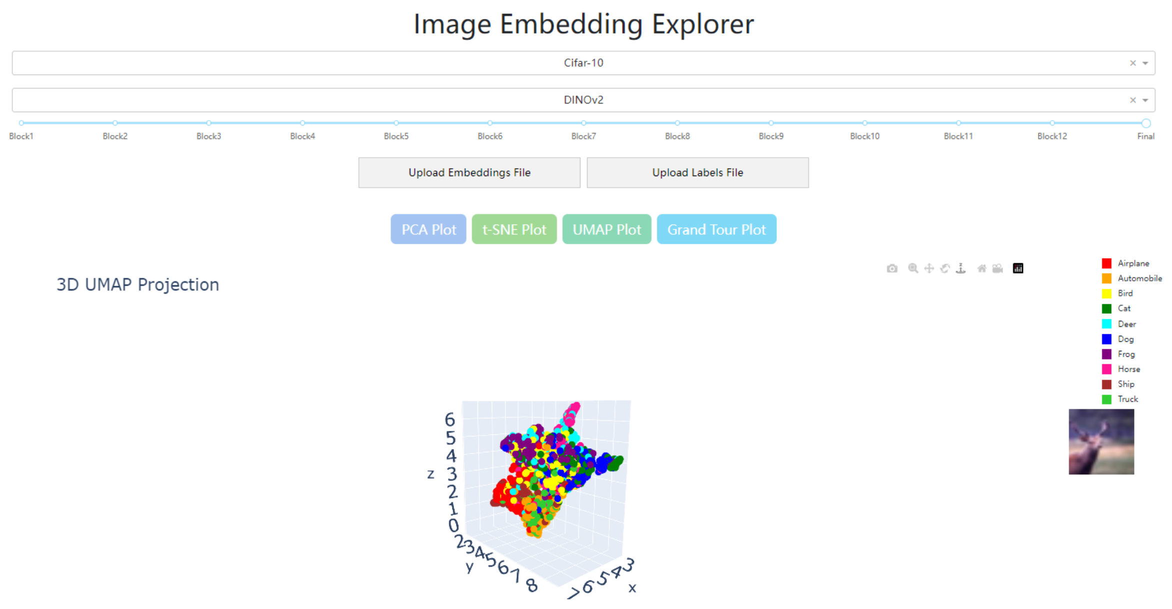
Image Embedding Explorer

Cifar-10

DINOv2

Figure A1: User Interface