TU WIEN Informatics

Zero-Shot Scatterplot Annotation

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Pascal Rupp
Matrikelnummer 12323804

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assoc. Prof. Dr.in techn. Manuela Waldner, MSc

Wien, 10. August 2026

Pascal Rupp	Manuela Waldner

TU WIEN Informatics

Zero-Shot Scatterplot Annotation

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Pascal Rupp
Registration Number 12323804

to the Faculty of Informatics

at the TU Wien

Advisor: Assoc. Prof. Dr.in techn. Manuela Waldner, MSc

Vienna, August 10, 2026

Pascal Rupp	Manuela Waldner

Erklärung zur Verfassung der Arbeit

Pascal Rupp

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 10. August 2026

Pascal Rupp

Danksagung

An erster Stelle möchte ich mich bei meiner Betreuerin, Assoc. Prof. in Dr. in techn. Manuela Waldner, MSc, für die Betreuung dieser Arbeit bedanken. Die regelmäßigen Besprechungen und das direkte, konstruktive Feedback haben die Arbeit über die gesamte Bearbeitungszeit hinweg auf Kurs gehalten.

Ebenso danke ich meiner Familie und meinen Freunden für ihre Unterstützung während des gesamten Studiums. Auch wenn ich in stressigen Phasen wenig Zeit für sie hatte, haben sie das immer verstanden — dafür bin ich sehr dankbar.

Acknowledgements

First and foremost, I would like to thank my supervisor, Assoc. Prof. Dr.techn. Manuela Waldner, MSc, for supervising this thesis. Our regular meetings and her direct, constructive feedback kept the work on track throughout.

I would also like to thank my family and friends for their support throughout my studies. Even during busy stretches when I had little time for them, they always understood — I'm grateful for that.

Kurzfassung

Streudiagramme von Bild-Embeddings gruppieren visuell ähnliche Bilder zu Clustern. Um zu verstehen, wofür ein Cluster steht, müssen die einzelnen Bilder inspiziert und ihre Gemeinsamkeiten interpretiert werden. In dieser Arbeit wird untersucht, ob sich ein vom Nutzer vorgegebenes Vokabular an erwarteten Konzepten mithilfe eines vortrainierten Vision-Language-Modells interaktiv und automatisch in ein bestehendes Streudiagramm platzieren lässt.

Der Ansatz nutzt den gemeinsamen hochdimensionalen Embedding-Raum von Contrastive Language-Image Pretraining (CLIP) für Bilder und Text. Ein Textlabel wird genauso codiert wie ein Bild und mithilfe des angepassten Projektionsmodells Uniform Manifold Approximation and Projection (UMAP) in das bestehende zweidimensionale Layout der Bild-Embeddings projiziert. Da beide CLIP-Encoder darauf trainiert wurden, denselben hochdimensionalen Raum zu teilen, liefert die Position des Labels im resultierenden Streudiagramm eine Näherungsantwort auf die Frage, wo Bilder zu finden sind, die zu diesem Wort passen – und das ohne Klassifikator-Training und ohne ein einziges gelabeltes Beispiel. Eine einfache Oberfläche zur Konzepterstellung ermöglicht es, ein eigenes Annotationsvokabular zu definieren, anzupassen und sofort im Streudiagramm zu platzieren.

Der Ansatz wird auf den Datensätzen Canadian Institute for Advanced Research-100 (CIFAR-100) und Fashion-Modified National Institute of Standards and Technology database (Fashion-MNIST) evaluiert. Über alle 100 CIFAR-100-Klassen hinweg liegen die projizierten Labels im Durchschnitt etwa so nah an ihrem Zielcluster wie ein typisches Bild dieser Klasse selbst. Die Genauigkeit schwankt jedoch deutlich: Visuell und semantisch klar abgegrenzte Klassen werden zuverlässig beschriftet, während Klassen, die in CLIPs eigenem Embedding-Raum überlappen – etwa mehrere personenbezogene Kategorien –, nicht zuverlässig erkannt werden. Das System generalisiert außerdem auf Vokabular, mit dem der Datensatz nie beschriftet wurde, und bleibt dabei im interaktiven Zeitbudget (rund 21 ms pro Label nach einmaligem Setup).

Die Ergebnisse zeigen: Nutzerdefinierten Text direkt in ein Bild-Streudiagramm zu projizieren ist ein schneller und praktikabler Weg, dessen Cluster ohne gelabelte Trainingsdaten interpretierbar zu machen – mit einer Zuverlässigkeit, die durch das zugrunde liegende Vision-Language-Modell begrenzt wird und nicht durch den Projektionsmechanismus selbst.

Abstract

Scatter plots of image embeddings group visually similar images into clusters. To understand what a cluster represents, the individual images must be examined and their commonalities interpreted. This paper investigates whether a user-specified vocabulary of expected concepts can be interactively and automatically placed into an existing scatter plot using a pre-trained vision-language model.

The approach utilizes the shared high-dimensional embedding space of Contrastive Language-Image Pretraining (CLIP) for images and text. A text label is encoded in the same way as an image and projected into the existing two-dimensional layout of the image embeddings using the adapted Uniform Manifold Approximation and Projection (UMAP) model. Since both CLIP encoders were trained to share the same high-dimensional space, the position of the label in the resulting scatter plot provides an approximate answer to the question of where to find images that match that word—without any classifier training and without a single labeled example. A simple concept creation interface allows users to define and customize their own annotation vocabulary and immediately place it on the scatter plot.

The approach is evaluated on the Canadian Institute for Advanced Research-100 (CIFAR-100) and Fashion-Modified National Institute of Standards and Technology database (Fashion-MNIST) datasets. Across all 100 CIFAR-100 classes, the projected labels are, on average, about as close to their target cluster as a typical image from that class itself. However, accuracy varies significantly: classes that are clearly distinct visually and semantically are labeled reliably, while classes that overlap in CLIP's own embedding space—such as several person-related categories—are not reliably recognized. The system also generalizes to vocabulary that was never used to label the dataset, while remaining within the interactive time budget (approximately 21 ms per label after a one-time setup).

The results show that projecting user-defined text directly onto an image scatter plot is a fast and practical way to make its clusters interpretable without labeled training data—with reliability limited by the underlying vision-language model rather than by the projection mechanism.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**

2 Related Work **3**
2.1 Visualization of Unstructured Data 3
2.2 Scatterplot Annotation and Embedding Exploration 3
2.3 Semantic Labeling with Vision-Language Models 4

3 Technical Foundations **5**
3.1 Embedding Spaces and Similarity Metrics 5
3.2 Dimensionality Reduction 5
3.3 Vision-Language Models 6

4 Concept and Design **7**
4.1 Requirements 8
4.2 Preprocessing 9
4.3 Label Candidate Generation 10
4.4 Interaction Design 11

5 Implementation **15**
5.1 System Architecture 15
5.2 Frontend 16
5.3 Backend 17
5.4 JDE Sandbox Integration 18
5.5 Performance Optimizations 18

6 Evaluation **19**
6.1 Experimental Setup 19
6.2 Quantitative Accuracy 20

6.3 Generalization and Robustness 23
6.4 Computational Performance 26

7 Conclusions **27**
7.1 Discussion 27
7.2 Answering the Research Questions 28
7.3 Limitations 29
7.4 Future Work 30

Overview of Generative AI Tools Used **33**

List of Figures **35**

List of Tables **37**

List of Algorithms **39**

Bibliography **41**

CHAPTER 1

Introduction

Dimensionality reduction techniques such as UMAP (Uniform Manifold Approximation and Projection) [MHSG18] and t-SNE (t-distributed Stochastic Neighbor Embedding) [vdMH08] project high-dimensional data into 2D scatterplots, making similarity relationships in large collections explorable. Clusters often emerge naturally, but without semantic labels, what a cluster represents remains hard to interpret.

(a) Without labels

(b) With projected text labels

Figure 1.1: The UMAP scatterplot of image embeddings from CIFAR-100 is shown before and after projecting user-defined text labels into it. Without labels (a), the viewer has to inspect the images in each cluster to identify what each cluster of thumbnails represents. In (b), however, each cluster is labelled directly, including those for concepts that the dataset was never explicitly labelled with, such as “vehicle” or “a picture of a lion”.

Annotations can help by highlighting structures and adding textual explanations; systems like ChartAccent [RBL+17] show how this supports storytelling and guides attention in

data-driven visualizations. Most existing approaches, however, rely on manual input or labeled training data, limiting them for open-ended exploration of unseen categories.

Recent vision-language models such as Contrastive Language-Image Pre-training (CLIP) [RKH+21] learn joint embeddings of images and text from large-scale image-caption data, enabling zero-shot comparison between visual and linguistic concepts out of the box. This suggests that a user-provided vocabulary of expected concepts could be placed directly into such a scatterplot by projecting text labels into the same low-dimensional-space, with an interactive interface letting users define, explore, and refine that vocabulary.

This idea is harder to realize than it first appears. UMAP does not preserve distances from the original embedding space in a globally consistent way — it optimizes for local neighborhood structure, and the resulting layout depends on the specific set of points being projected [MHSG18]. A label semantically close to a cluster in CLIP's 768-dimensional space is therefore not guaranteed to land near it after projection — and the reverse can happen too: a label ending up close to a cluster it has no real relation to. Whether zero-shot text embeddings can be placed accurately and consistently enough into a shared projection to serve as usable annotations is, to our knowledge, an open question.

This thesis investigates how zero-shot vision-language models can be used to place a user-provided vocabulary of expected concepts directly into a similarity-preserving scatterplot of images, including an interactive interface for defining and refining that vocabulary. The work is guided by two research questions:

- **RQ1:** To what extent can zero-shot classification models place user-provided vocabulary meaningfully into similarity-preserving scatterplot visualizations?
- **RQ2:** How can labels be used as a direct interactive exploration method in scatterplots?

RQ1 is addressed quantitatively on the Canadian Institute for Advanced Research-100 database (CIFAR-100) [Kri09] by measuring the distance between placed labels and the ground-truth centroids of their corresponding image clusters, complemented by further quantitative analysis of out-of-vocabulary terms and label name sensitivity, as well as a qualitative discussion of individual placement examples. RQ2 is addressed by the interaction design itself (Section 4.4) and illustrated using the results from Chapter 6. The main contribution is a working prototype demonstrating zero-shot label placement as a viable approach for interactive image exploration, without labeled training data or a predefined category taxonomy.

CHAPTER 2

Related Work

2.1 Visualization of Unstructured Data

A common approach to exploring unstructured image collections is to project high-dimensional feature vectors into two dimensions using dimensionality reduction techniques such as UMAP [MHSG18] or t-SNE [vdMH08], producing a scatterplot in which visual similarity corresponds to spatial proximity — an approach used, for instance, by Embedding Atlas [RHLM25] for interactive exploration of large embedding collections (see Section 2.2 for details). The resulting layouts are effective for revealing cluster structure, but interpreting what a cluster represents remains a manual task. Zhao et al. [ZLC+19] evaluate four multi-dimensional visualization techniques — including Radviz, whose spring-based layout places items closer to the dimension anchors they resemble most — and show that no single technique dominates across all analytical tasks, motivating hybrid approaches that combine a 2D projection with additional semantic cues rather than relying on the projection alone.

2.2 Scatterplot Annotation and Embedding Exploration

Annotating scatterplots with meaningful labels is a recurring challenge in information visualization, and existing approaches span a spectrum from fully manual to fully automatic [RDQR24]. At the manual end, tools like ChartAccent [RBL+17] let users add textual and visual annotations to charts to support storytelling and guide viewer attention. Such tools offer precise control over what is highlighted and how, but every annotation has to be placed by hand, which does not scale to exploratory settings with large, unlabeled datasets.

At the automatic end, classical label-placement approaches treat annotation as an optimization problem, minimizing overlap while maximizing proximity to labeled regions [CMS95]. These methods assume a fixed, known set of categories or features to

label, and therefore cannot accommodate a vocabulary that is open-ended or not known in advance — a limitation that zero-shot vision-language models such as CLIP [RKH+21] only recently overcame, by comparing arbitrary text against visual content without a predefined category set. More recent tools instead combine automatic clustering or structure detection with data-driven label generation, extending this idea beyond static charts: Shi et al. [SOWC24] automatically generate and place graphical annotations on animated scatterplots, and Jiang et al. [JSL+25] generate multi-level annotations for time series visualizations. Closer to this thesis' setting, Embedding Atlas [RHLM25] uses density-based clustering and automated labeling to annotate large-scale embedding projections interactively, scaling to millions of points in the browser. Across all of these automatic approaches, labels are derived from the data itself rather than from user-defined concepts, which means the resulting vocabulary cannot be controlled or anticipated by the user.

The system presented in this thesis occupies a middle ground between these two ends: instead of requiring manual placement or generating labels automatically from the data, it lets users define their own vocabulary through a simple interface for creating concepts and projects these concepts directly into the embedding space — making label placement explicitly vocabulary-driven while remaining as effortless as an automatic approach. Table 2.1 summarizes this positioning.

Table 2.1: Positioning of this thesis relative to manual and automatic scatterplot annotation approaches.

	No training data	Open vocabulary	Automatic placement
ChartAccent [RBL+17]	✓	✓	–
Embedding Atlas [RHLM25]	✓	–	✓
This thesis	✓	✓	✓

The distinguishing factor is the combination, not any single column: ChartAccent supports open vocabulary but requires manual placement, while Embedding Atlas automates placement but restricts the vocabulary to what it derives from the data. This thesis combines both: an open vocabulary placed automatically.

2.3 Semantic Labeling with Vision-Language Models

The emergence of large vision-language models has opened new possibilities for semantic labeling of visual data without prior training. CLIP's zero-shot transfer capability has been applied to image classification, retrieval, and clustering, but its use for interactive scatterplot annotation remains largely unexplored.

This is exactly the gap this thesis addresses: rather than using CLIP for classification or retrieval, it uses CLIP's joint space (Section 3.3) to place user-defined text directly into a scatterplot of images, as described in Chapter 4.

CHAPTER 3

Technical Foundations

3.1 Embedding Spaces and Similarity Metrics

An embedding space maps data into vectors in $\mathbb{R}^d$, such that semantic relationships are encoded as geometric proximity. In this thesis, images and text count as high-dimensional data not in their raw form (a pixel grid or a character sequence), but through the representation produced by a vision or language encoder: passing an image through CLIP's vision encoder, for instance, yields a single vector in $\mathbb{R}^{768}$ (Section 3.3). It is this encoded representation, not the raw input, that is referred to as an embedding throughout this thesis. Comparing two embeddings is therefore reduced to computing the distance or similarity between vectors.

A common measure is cosine similarity:

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|}$$

This measure depends only on direction and is invariant to vector magnitude. When embeddings are L2-normalized, cosine similarity reduces to a dot product. In this system, all embeddings are normalized after extraction, so similarity computation becomes a single matrix operation.

3.2 Dimensionality Reduction

High-dimensional embedding spaces, such as those produced by CLIP, are not directly interpretable. Dimensionality reduction methods address this issue by projecting vectors into two dimensions while preserving structural relationships.

UMAP [MHSG18] constructs a neighborhood graph in the original space and optimizes a low-dimensional layout that preserves this structure. It keeps local neighborhoods intact while still scaling efficiently to large datasets.

UMAP supports arbitrary distance metrics. Since embeddings in this system are compared using cosine similarity, UMAP is configured to use cosine distance as well, ensuring consistency across the pipeline. Once fitted, UMAP can embed new points into an existing layout without retraining, which enables the interactive placement of user-defined text labels within an existing image scatterplot (Chapter 4).

3.3 Vision-Language Models

Vision-language models learn a shared representation space for images and text. This enables cross-modal tasks, such as retrieval and zero-shot classification.

CLIP [RKH+21] consists of separate image and text encoders that are trained jointly on large datasets of image-caption pairs using a contrastive objective. Both encoders map inputs into the same embedding space where matching pairs are pulled together and non-matching pairs are pushed apart.

This shared space enables direct comparison between image and text embeddings without task-specific training. In this system, CLIP provides the embedding space in which images and user-defined text labels are represented. This allows labels to be projected into the same space as the image scatterplot using cosine similarity (Section 3.1) and UMAP-based projection (Chapter 4).

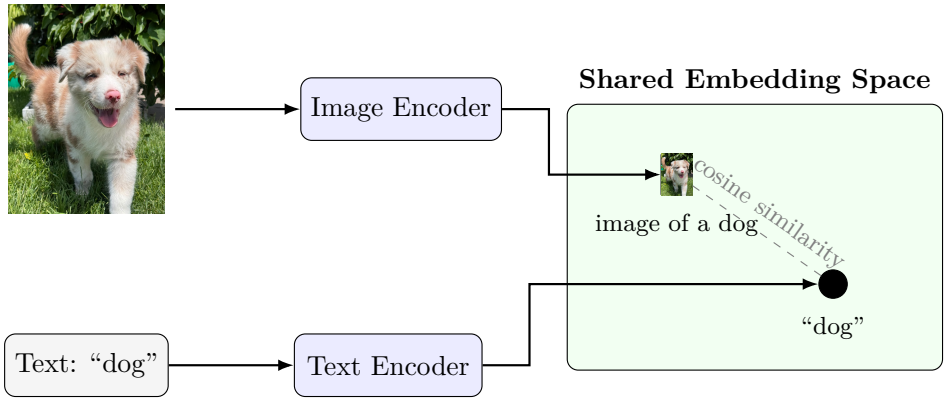


Figure 3.1: CLIP maps images and text into a shared embedding space, enabling direct comparison via vector similarity.

Together, these components form the basis of the system. CLIP defines the joint space; cosine similarity provides a unified comparison metric; and UMAP projects the resulting structure into two dimensions for visualization.

CHAPTER 4

Concept and Design

The proposed system addresses the problem of placing a user-provided vocabulary of expected concepts meaningfully into a similarity-preserving scatterplot visualization of an unstructured image collection, without prior training. Rather than clustering the images and then searching for the best label per cluster, the system takes the opposite direction: the user supplies the labels they expect to find, and the system shows where in the scatterplot images matching those labels are likely to be located. The central idea is to leverage a pre-trained vision-language model — specifically CLIP [RKH+21] — to embed both images and user-defined text labels into a shared semantic space, and to project this unified representation into two dimensions for interactive exploration.

The pipeline consists of three conceptual stages, illustrated in Figure 4.1: (1) image features are extracted offline and stored as precomputed representations; (2) a dimensionality reduction is fitted on the image embeddings once per session, producing a stable 2D layout; (3) user-provided text labels are encoded and projected into the same 2D space at interaction time, enabling direct geometric comparison between image clusters and semantic concepts.

A key design principle is the use of a single joint space for both modalities: image and text embeddings are obtained from the same vision-language model and therefore reside in the same space by construction. This alignment is what makes zero-shot annotation possible — no classifier is trained, and no labeled data is required.

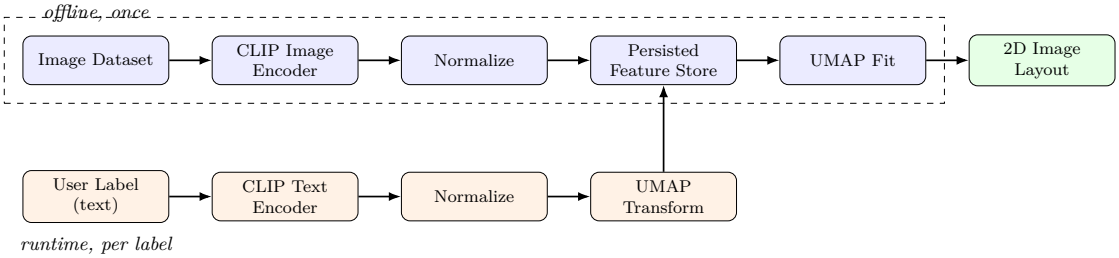


Figure 4.1: System overview: image features are extracted, normalized, and persisted offline; a dimensionality reduction is fitted once on this store to produce a stable 2D layout; user-provided text labels are then encoded, normalized, and projected into that same 2D space at runtime.

4.1 Requirements

The design of the system is guided by the following requirements.

Zero-shot annotation. Without any task-specific training, the system must assign semantic labels to image clusters on the fly. To accommodate unknown or evolving user vocabularies, label candidates are provided interactively at runtime and compared directly within the shared embedding space.

Evolving vocabulary. The vocabulary is not known in advance: users start with no labels at all and add, remove, or rename concepts as their understanding of the dataset develops. The system must support this without any retraining or reconfiguration step.

Semantic consistency. Image and text embeddings must reside in a shared, geometrically coherent space so that proximity in the 2D projection reflects genuine semantic relatedness.

Real-time interaction. Text label projection must be fast enough to support interactive exploration. Computationally expensive steps — image encoding and projection fitting — are therefore performed offline or during session initialization. At runtime, label mapping is limited to text encoding and projection transform. The projection is fitted once and reused, so the image layout remains stable while the user edits vocabulary (see Section 6.4).

Scalability. The system should handle datasets of practical size (CIFAR-100 and the Fashion-Modified National Institute of Standards and Technology database (Fashion-MNIST) [XRV17]) without prohibitive runtime cost.

Interpretability. The relationship between user concepts and image clusters must be visually legible. Label anchors are placed directly in scatterplot coordinates, visually distinct from image points.

4.2 Preprocessing

Preprocessing is split into an offline phase for image embeddings and an online phase for text embeddings, reflecting the asymmetry between the two modalities: image datasets are static and known in advance, while user-defined text labels are dynamic and unpredictable at design time. Both paths are shown together in Figure 4.1. Algorithms 4.1–4.4 provide the corresponding pseudocode, and Table 4.1 defines the notation.

Table 4.1: Notation used in Algorithms 4.1–4.4.

Symbol	Meaning
N	Number of images in the dataset
D	Embedding dimensionality
$F \in \mathbb{R}^{N \times D}$	Matrix of normalized image embeddings
$M \leq N$	Sample size used to fit the UMAP projection
$\mathcal{U}$	Fitted UMAP projection model
$P \in \mathbb{R}^{M \times 2}$	2D layout of the sampled image embeddings
$L = \{l_1, l_2, \dots\}$	Set of user-provided text labels
$\lvert L \rvert$	Number of user-provided text labels in the current interaction
$T \in \mathbb{R}^{\lvert L \rvert \times D}$	Matrix of normalized text label embeddings
C_{enc}	Cost of one CLIP image-encoding pass
C_{text}	Cost of one CLIP text-encoding pass
k	Top-k support size per label

Image Embeddings

Image features are extracted offline, before any interactive session, using the vision encoder of the chosen vision-language model, and normalized to unit length to enable direct similarity comparison in the same embedding space. Images are processed in batches for efficiency. The resulting feature vectors, and filenames are persisted to disk for reuse across sessions, and loaded once at session initialization for fast lookup during projection fitting and focus score computation. This offline step is summarized in Algorithm 4.1; its output is the feature matrix F used in the projection fitting step below.

Algorithm 4.1: Offline Image Feature Extraction

Input: Dataset path, output directory
Output: Feature matrix $F \in \mathbb{R}^{N \times D}$, label vector, filename index

```
model ← LoadVisionEncoder();
foreach batch B of images from dataset do
    f ← model.Encode(B);
    f ← f / ‖f‖₂ ;                          // normalize to unit norm
    Append f to F;
end
Save F, labels, filenames to disk;
```

Projection Fitting

At the start of a session, the stored image embeddings are used to fit a 2D UMAP projection once, before the user interacts with the system. Fitting is performed on a sample F_s of size $M \leq N$ rather than the full feature matrix to keep session start-up fast. Here, F_s is the subset of M image embeddings used for fitting. Algorithm 4.2 describes this step. The fitted model $\mathcal{U}$ is kept and reused to place new text labels into the same 2D space without refitting, satisfying the stable-layout requirement from Section 4.1. Here, $\mathcal{U}$ denotes the fitted UMAP transform itself, while P denotes the resulting 2D coordinates of the sampled image embeddings.

Algorithm 4.2: Session Initialization — Projection Fitting

Input: Feature matrix $F \in \mathbb{R}^{N \times D}$, sample size $M \leq N$
Output: Fitted projection model $\mathcal{U}$, 2D layout $P \in \mathbb{R}^{M \times 2}$
1 $F_s \leftarrow \textsc{Sample}(F, M)$;
2 $\mathcal{U}, P \leftarrow \textsc{FitProjection}(F_s,\ \text{metric} = \text{cosine},\ \text{dims} = 2)$;
3 Store $\mathcal{U}$ in session;

Text Embeddings

Text embeddings are generated on-the-fly at interaction time, since user-provided label candidates are not known in advance. When the user triggers label projection, the current vocabulary is deduplicated and encoded using the same encoder and normalization as the image features, ensuring both modalities remain directly comparable without any further alignment step.

4.3 Label Candidate Generation

Label candidates are entered through the existing concept-creation interface of the JDE Sandbox framework (Chapter 5). For this thesis, the interface is treated as a lightweight vocabulary input mechanism: users provide textual concept terms, and these terms are projected into the scatterplot as label candidates. The placement pipeline therefore depends only on the text content and not on any explicit graph structure.

Vocabulary Extraction

The interface is traversed to collect all textual terms from concept text blocks. Connections between blocks are ignored for label generation; each text block contributes terms independently. Terms are normalized (whitespace trimmed and collapsed) and deduplicated case-insensitively — i.e., merged so that each distinct term appears only once regardless of capitalization — so that variants such as *shoe* and *Shoe* are treated as a single candidate while the original capitalization is preserved in the output.

Projection into 2D Space

The deduplicated vocabulary is encoded and projected into 2D using the already-fitted projection model from Section 4.2. Because this model was fitted on image embeddings from the same shared space, the resulting 2D coordinates for text labels are geometrically coherent with the image scatterplot layout: concepts semantically similar to a cluster of images appear in proximity to that cluster. This runtime operation is specified in Algorithm 4.3 and measured in Section 6.4.

Algorithm 4.3: Runtime — Text Label Projection

Input: Text labels $L = \{l_1, l_2, \ldots, l_{|L|}\}$, fitted projection model $\mathcal{U}$
Output: 2D positions $\{(x_i, y_i)\}$ for each label

$L' \leftarrow \textsc{Deduplicate}(\textsc{Normalize}(L))$;
$T \leftarrow \text{model.}\textsc{EncodeText}(L')$;
$T \leftarrow T \,/\, \|T\|_2$; `// normalize, same space as image features`
$P_T \leftarrow \mathcal{U}.\textsc{Transform}(T)$;
return $\{l'_i \mapsto (x_i, y_i) \mid (x_i, y_i) \in P_T\}$;

Visual Representation

Projected label candidates are rendered as anchor points directly in scatterplot coordinates, visually distinct from image points by size, color, and a text label, and rendered at the highest z-order to ensure visibility regardless of local point density.

4.4 Interaction Design

The system supports an iterative annotation workflow in which the user alternates between inspecting the image scatterplot and adjusting their concept vocabulary, closing the loop each time labels are re-projected (Figure 4.2).

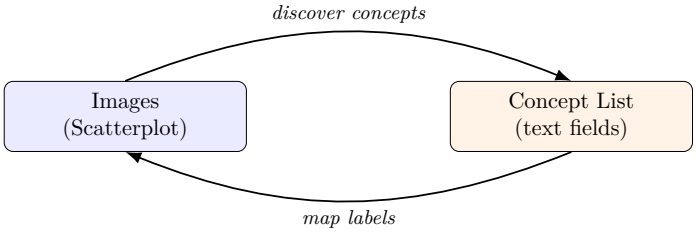


Figure 4.2: The core interaction loop: inspecting the images in the scatterplot informs which concepts the user adds to the concept list (*discover concepts*); projecting that list back into the scatterplot then shows where those concepts are located among the images (*map labels*), which in turn informs further discovery.

In practice, this loop proceeds as follows:

1. **Adjust vocabulary.** The user adds, removes, or renames concepts in the concept list — for instance because a cluster in the scatterplot looked unfamiliar, or because a previously placed label landed somewhere unexpected.

2. **Map labels.** Triggering label projection extracts the current vocabulary, encodes it, and projects it into the 2D scatterplot space.

3. **Discover concepts.** The user examines the placement of concept anchors relative to image clusters and uses scatterplot interactions to explore the data in detail, which often suggests the next concept to add or adjust, restarting the loop.

Scatterplot Interactions

The scatterplot supports zooming, panning, and lasso selection to isolate dense local regions. This allows users to inspect selected images despite clutter, compare neighborhoods, and refine label terms. When Focus is active, hovering over a concept anchor highlights its top-k support items (the most similar image points).

The top-k support set, and the label assignment used by the spatial regrouping below, are computed by Algorithm 4.4 directly in CLIP's high-dimensional embedding space (independent of the 2D projection). Here, support$[i]$ denotes the set of the k most similar images for label i.

Algorithm 4.4: Runtime — Focus Score Computation

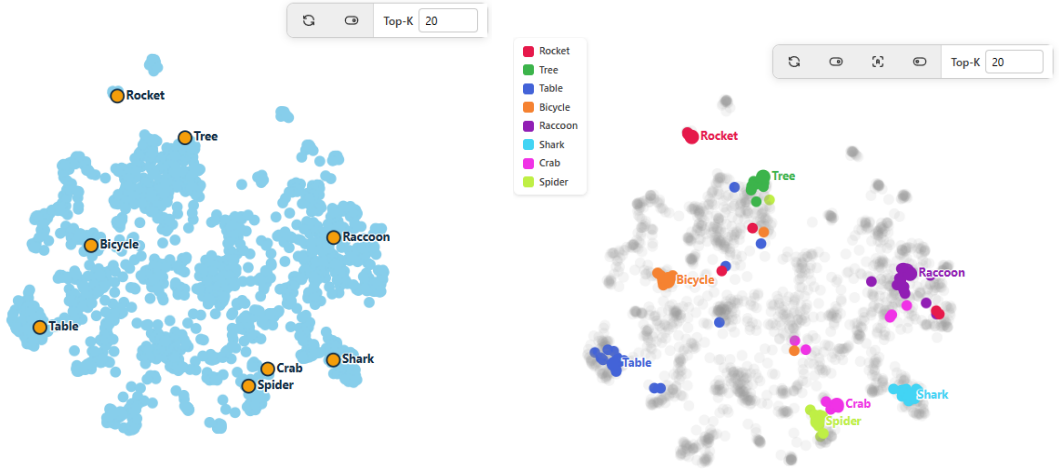
Input: Text embeddings $T \in \mathbb{R}^{|L| \times D}$, image embeddings $F \in \mathbb{R}^{N \times D}$, parameter k (top-k support size)
Output: Per-label top-k support sets, label assignments per image

$S \leftarrow T \cdot F^{\top}$; `// similarity matrix, shape` $|L| \times N$
$k \leftarrow \max(1,\ \min(k,\ N))$; `// effective` k`, bounded by sample size`
foreach *label* $i \in \{1, \ldots, |L|\}$ **do**
  support$[i] \leftarrow$ TopK$(S[i, :],\ k)$;
end
foreach *image* $j \in \{1, \ldots, N\}$ **do**
  assignment$[j] \leftarrow \arg\max_i\ S[i, j]$;
end
return support sets, assignments;

Label-Driven Spatial Regrouping

To reduce clutter, the system can displace image points toward a concept anchor according to image-label similarity. Highly associated points move closer to the anchor, while unrelated points remain near their original position and are rendered with slightly reduced opacity. This creates local white space and a clearer concept-centered structure.

The underlying UMAP layout remains unchanged; only rendered positions are shifted, and the effect can be toggled or reversed. Unlike classical Focus+Context methods [Fur86, CKB08], this does not distort the projection itself.



(a) Regrouping disabled — only the concept anchor is visible.

(b) Regrouping enabled — similar images are pulled toward the concept anchor.

Figure 4.3: Label-driven spatial regrouping: (a) without regrouping, only the anchor marks the concept; (b) with regrouping enabled, associated images move toward the anchor.

Computational Cost

Table 4.2 summarizes the computational cost of the four algorithmic phases described above — offline feature extraction (Algorithm 4.1), projection fitting (Algorithm 4.2), text label projection (Algorithm 4.3), and focus score computation (Algorithm 4.4). Chapter 5 maps these phases to the concrete framework implementation.

Phase	Complexity	Typical cost
1. Offline feature extraction	$\mathcal{O}(N \cdot C_{\text{enc}})$	minutes (once per dataset)
2. Projection fitting	$\mathcal{O}(M \log M)$ [MHSG18]	seconds (once per session)
3. Text label projection	$\mathcal{O}(\lvert L\rvert \cdot C_{\text{text}} + \lvert L\rvert \cdot D)$	200–500 ms
4. Focus score computation	$\mathcal{O}(\lvert L\rvert \cdot N)$	$<$50 ms

Table 4.2: Computational complexity and typical runtime cost per phase.

CHAPTER 5

Implementation

This chapter describes how the concept presented in Chapter 4 was realized within an existing interactive visualization framework, referred to as the JDE Sandbox [Wol24], developed at Technical University Vienna (TU Wien). Where relevant, this chapter distinguishes between components that were already part of the framework and components implemented or substantially extended specifically for this thesis.

5.1 System Architecture

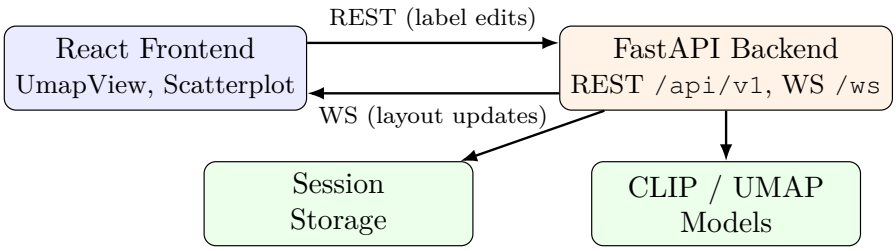


Figure 5.1: System architecture: client-server communication between the React frontend and the FastAPI backend.

The system follows a client-server architecture, as shown in Figure 5.1. The FastAPI backend exposes a REST API under `/api/v1` and a separate WebSocket endpoint under `/ws` for event-based updates. REST routes are organized by resource (sessions, datasets, data, concept graph, image grid, models). The front end communicates with the back end through a generated OpenAPI client and authenticates requests via a session token. The WebSocket connection includes automatic reconnection.

The backend is organized around the concept of a session. Each session holds unique data, including identifiers, configurations, timestamps, and more, which is saved in JSON periodically and with locking to avoid conflicts. This architecture, including its

persistence and access control, was part of the framework and wasn't modified for this thesis.

5.2 Frontend

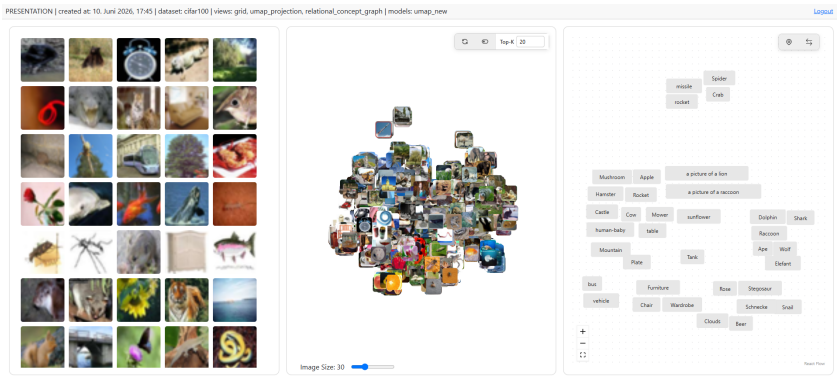


Figure 5.2: The JDE Sandbox layout, composed of a DataView, FrameView, and KnowledgeView. This general layout is part of the existing framework and was not modified for this thesis.

The frontend is React. The existing framework provided the page structure: a Sandbox view with DataView, FrameView, and KnowledgeView, and the Concept Map interfaces (Tldraw- and ReactFlow-based).

This thesis extended the UMAP view (`UmapView.tsx`) to support the interaction loop in Section 4.4: by triggering label projection, requesting and displaying focus scores for individual labels, rendering top-k support highlighting, and rendering the ground-truth centroid overlay. The scatterplot component was also extended to support visual encodings resulting from this thesis: lasso selection and the label-driven spatial regrouping that pulls high-similarity image points toward their concept anchor (Section 4.4). These interactions communicate with the backend through the generated API client and an event bus that connects the Concept Map view to the UMAP view, allowing edits in the Concept Map to trigger a new label projection without a full page reload.

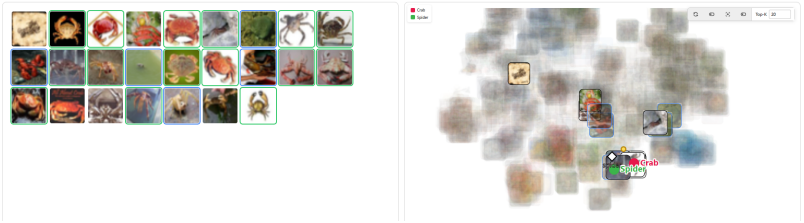


Figure 5.3: The UMAP scatterplot view of the concepts *crab* and *spider* with both anchors placed at the bottom right, shows the ground-truth cluster membership via colored image borders and active top-k support highlighting control (`Top-K` control, top right).

5.3 Backend

The backend is implemented in Python using FastAPI. The router structure, which includes separate endpoints for sessions, datasets, data, the concept graph, and the image grid, was already part of the framework. Models are registered as long-lived singleton instances in a shared registry, and the framework's update flow dispatches requests to the appropriate model and stores the results in the session.

The core implementation of this thesis involves the model registered for this purpose, `umap_projection_new.py`, together with the router endpoints that expose it: *map-concepts*, *focus-scores*, and *ground-truth*. This model applies UMAP to CLIP image embeddings using cosine distance and projects user-defined text labels into the resulting layout via `transform` (concept described in Section 4.3 and Algorithm 4.3). CLIP is loaded lazily on first use and cached on the model instance to avoid repeated initialization. Text encoding for label projection and focus score computation reuses this cached instance.

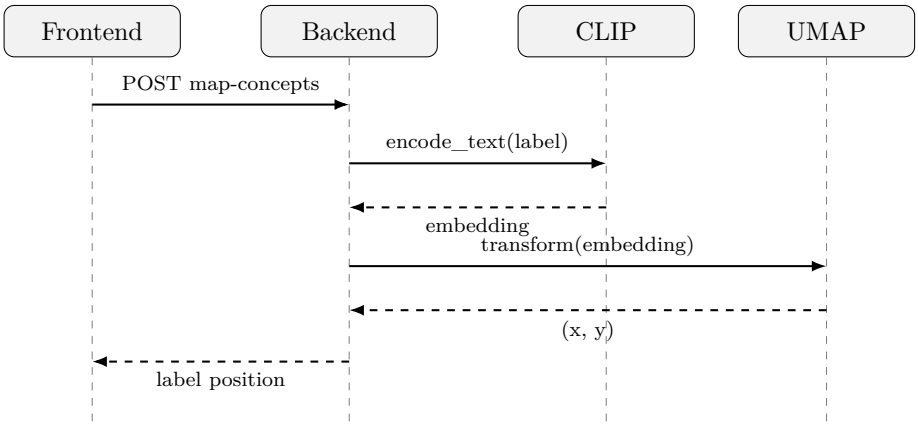


Figure 5.4: Sequence of a single label projection triggered from the Concept Map: the frontend sends the edited vocabulary to the backend, which encodes it via CLIP and projects it via the already-fitted UMAP model before returning updated positions.

Offline Feature Extraction Pipeline

This thesis introduces an offline CLIP feature extraction pipeline with one generic extractor and two dataset-specific scripts. The files are `clip_feature_extractor.py`, `extract_cifar100_clip_features.py`, and `extract_fmnist_clip_features.py`. The pipeline precomputes and persists image embeddings in advance rather than encoding images on demand (Algorithm 4.1).

The dataset modules `cifar100_dataset.py` and `fmnist_dataset.py` were updated to load these precomputed features (indexed by filename) instead of raw images. Shared constants, including the 768-dimensional CLIP embedding size, were consolidated into a single configuration module so that the image and text pipelines use the same embedding space.

5.4 JDE Sandbox Integration

The JDE Sandbox is an interactive environment where users explore a selected dataset within a session. It is configuration-driven; a profile combines a dataset, one or more views, and one or more models. The sandbox then assembles the corresponding UI and data pipeline. Two new profiles, *cifar100-umap* and *fmnist-umap*, were added to integrate the work of this thesis into this mechanism. The frontend and backend run as two separate processes during development (Vite for the frontend and Uvicorn for the backend), and no containerized deployment exists in the project.

The thesis didn't modify the Sandbox's core mechanisms, but it did extend the framework in the following ways: a new model, new endpoints, new dataset integrations, and an extended view. The Concept Map interfaces and the Sandbox layout are used as is.

5.5 Performance Optimizations

Several optimizations keep CLIP encoding and UMAP projection within interactive latency bounds, given the real-time requirements established in Section 4.1.

On the back end, image embeddings are precomputed offline and loaded from disk rather than extracted per request. During precomputation, CLIP image encoding is batched rather than processed one image at a time. All embeddings are L2-normalized at extraction time, enabling similarity computations to be expressed as a single matrix multiplication (Section 3.1) instead of a more expensive distance calculation. UMAP recomputation is skipped when the underlying sample has not changed, thus avoiding redundant projection work on repeated requests. Generation of datasets that are not needed immediately runs as a non-blocking background task. Progress is reported incrementally over the WebSocket connection rather than blocking the request.

On the front end, updates to the concept graph are debounced before being synchronized with the back end to avoid redundant requests during rapid editing. The scatterplot requests thumbnail-sized images rather than full-resolution images and uses a backend-side thumbnail cache.

These optimizations are reflected in the latencies measured and reported in Section 6.4. (CLIP encoding $\sim$5.1 ms warm, UMAP transform $\sim$14.3 ms warm), which remain well within the bounds needed for interactive use.

CHAPTER 6

Evaluation

6.1 Experimental Setup

We evaluated the system on a 2,000-image subset of CIFAR-100 [Kri09], covering all 100 ground-truth classes with 20 images per class. CIFAR-100 was chosen over simpler benchmarks, such as Fashion-MNIST [XRV17], because its 100 semantically diverse classes provide a more challenging test for zero-shot label placement. Many classes are visually similar (e.g. *boy*, *baby*, *man*), and the label vocabulary spans a wide range of abstraction levels.

Sampling was stratified by class: within each run, 20 images were drawn uniformly at random without replacement from each class. This procedure guarantees class coverage while keeping the dataset size manageable with respect to visual readability (2,000 points in one scatterplot) and UMAP fitting latency (Section 6.4).

Image embeddings were precomputed offline using `openai/clip-vit-large-patch14` at 768 dimensions. A UMAP model was then fitted to the full embedding matrix using cosine distance, matching the runtime configuration of the interactive system. Text labels were projected into the same 2D space via `umap_model.transform()`.

Label placement accuracy is measured as the Euclidean distance between the projected text label and the centroid of its corresponding image cluster in 2D UMAP space. After projection, the 2D coordinates are Cartesian, so Euclidean distance directly corresponds to the visual separation a user perceives in the scatterplot. This makes it the most interpretable accuracy metric in this context.

Experiments were run on an NVIDIA GeForce RTX 3050 Laptop GPU with an Intel Core i7 (12 cores) and 15.7 GB RAM, using Python 3.11, PyTorch 2.6 (CUDA 12.4), and umap-learn 0.5.12.

Because both the images sampled for each class and the UMAP fit itself involve a random component, a single run could easily overstate or understate the system's true accuracy. Here, sampling refers to which images are drawn into a subsample, not their embeddings. The CLIP features are precomputed once (see Section 4.2) and are simply looked up for the selected images. To obtain a stable estimate, every quantitative result is averaged over 200 repetitions: 20 independent, stratified subsamples of the dataset (20 images per class), each fitted with 10 separate UMAP runs using different random seeds. All other UMAP parameters remain fixed. For each of these 200 runs, we recalculate the label-to-centroid distance and the intra-cluster spread for each class, and we report the resulting mean and standard deviation. All subsample and UMAP seeds are logged for reproducibility.

6.2 Quantitative Accuracy

For validation purposes only, the system can render an optional overlay of ground-truth class centroids — the mean 2D position of all images belonging to a given class — directly in the scatterplot. Only the per-image class label is official ground truth here, taken directly from CIFAR-100's [Kri09] original annotations (and, in Section 6.2, Fashion-MNIST's [XRV17]); neither CLIP nor the dataset provides a centroid coordinate directly. The centroid itself is computed after projection, by averaging the 2D UMAP positions of all images sharing that label. This overlay plays no role in the interaction concept described in Chapter 4; it exists solely to make it possible to directly compare projected label positions against the dataset's known class boundaries for the evaluation reported in this chapter.

For each of the 100 CIFAR-100 classes, we project the class name and measure the Euclidean distance between the projected label and that class's image-centroid in 2D UMAP space. As a reference scale, we also compute each class's *intra-cluster spread*: the mean Euclidean distance between the class's own images and its centroid, in the same unitless 2D UMAP coordinates. The two values are therefore directly comparable. Intuitively, intra-cluster spread is the class's own width: if a label distance is below that width, the label still lies within the class's natural extent; if it is several times larger, the label missed the cluster. Averaged across all 100 classes and all 200 runs, the mean intra-cluster spread is 0.65 ($\sigma = 0.37$). As a concrete scale example, an error of about 0.30 is small relative to this average spread, whereas an error above about 1.3 is already more than twice that spread.

Across all 100 classes and all 200 runs, we measure a mean label-to-centroid distance of $\bar{d} = 0.61$ ($\sigma = 0.88$), indicating that, on average, labels land at or near the boundary of their target cluster. The standard deviation exceeding the mean already signals that this average hides substantial class-to-class variation rather than a tight, uniform accuracy — the per-class breakdown below shows why.

The best-performing classes tend to have visually distinct and compact clusters, and are placed there consistently: their low standard deviations show this is not a lucky

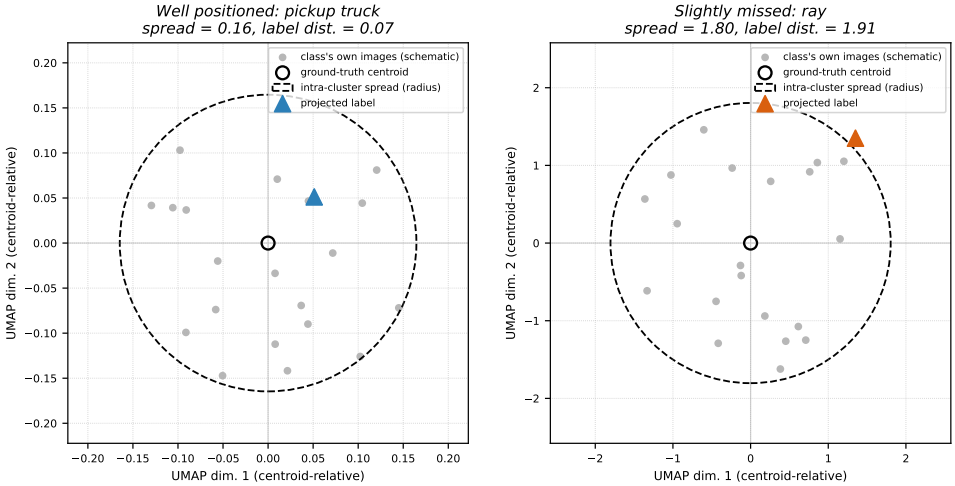


Figure 6.1: Schematic illustration of the intra-cluster spread metric for one well-positioned class (*pickup_truck*) and one class where the label lands just outside the cluster (*ray*). The circle radius (intra-cluster spread) and the label's distance from the ground-truth centroid are the real, measured values from Table 6.1; the individual gray points are a schematic illustration of how tightly a class's own images cluster around their centroid, not their actual 2D positions.

Table 6.1: Best and worst label placements by mean distance to ground-truth centroid, aggregated over the $N \times M = 200$ resampling runs (CIFAR-100).

	Class	Mean dist.	Std dist.	Intra-cluster spread
Best	pickup_truck	0.072	0.036	0.165
	keyboard	0.087	0.049	0.179
	couch	0.113	0.087	0.267
	snake	0.123	0.049	0.370
	rose	0.128	0.073	0.316
Worst	ray	1.910	0.287	1.804
	man	2.423	0.460	0.696
	boy	3.634	1.221	0.951
	baby	4.089	0.440	0.357
	sea	6.451	0.813	0.479

single draw. Poor placements fall into two patterns. The first pattern involves classes with high intra-cluster spread, such as *boy*, *man*, and *ray* (stingray), where images are visually heterogeneous. The second pattern involves classes where the label is pulled toward a semantically adjacent cluster, such as *baby* and *sea*, despite a tight cluster of their own — in this case, the label ends up in the wrong neighborhood entirely rather than merely scattered around the right one. This suggests that the hardest classes — the people-related and visually diffuse categories — are a robust property of CLIP's embedding space.

To put $\bar{d}$ in perspective, the average distance between two *different* classes' centroids in the same 2D space is 3.90 — roughly six times larger. So while $\bar{d} = 0.61$ is far from zero, it remains small relative to how far apart unrelated classes typically sit: on average, a placed label ends up only about a sixth of the way toward a neighboring class, rather than drifting into unrelated territory.

Figure 6.2 visualizes this directly: Each triangle marks where a text label was projected, each circle marks the true centroid of that class, and the connecting line is the error reported in Table 6.1. For the best cases, the triangle and circle almost overlap; for the worst cases, the line stretches across a visible portion of the layout.

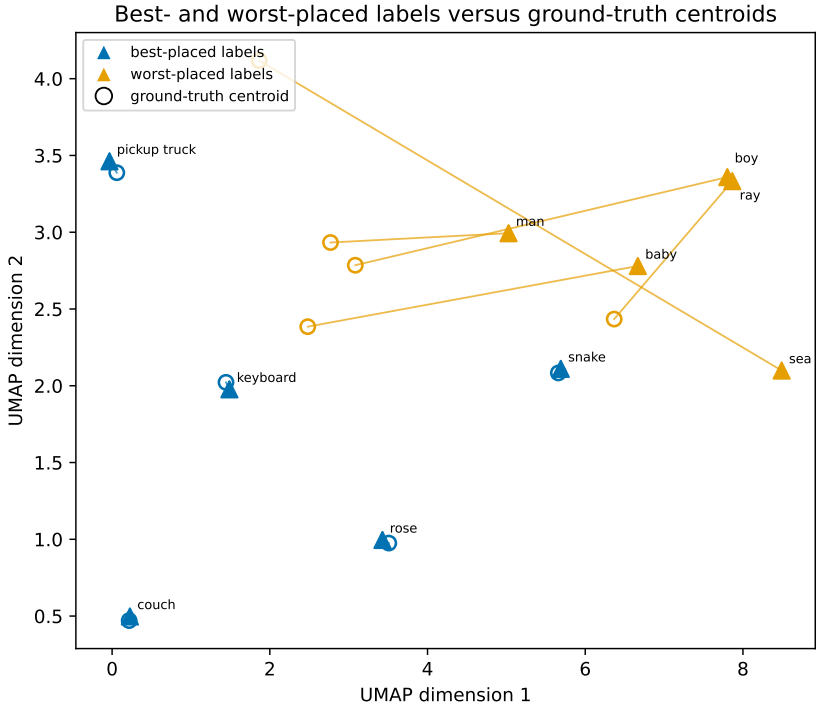


Figure 6.2: Projected label positions (triangles) versus ground-truth class centroids (circles) for the five best- and five worst-placed classes from Table 6.1, shown in blue and orange respectively. Connecting lines show the label-to-centroid error.

A Second Dataset: Fashion-MNIST

The same pipeline — identical CLIP variant, cosine-distance UMAP configuration, and label projection mechanism, without any dataset-specific tuning — also applies directly to Fashion-MNIST (10 classes) [XRV17], a domain that differs from CIFAR-100 in almost every visual respect: grayscale clothing items rather than natural color photographs. Figure 6.3 shows a handful of Fashion-MNIST labels projected into their scatterplot the same way as in Section 6.2, as a single illustrative example rather than a full repeat of the quantitative validation above.

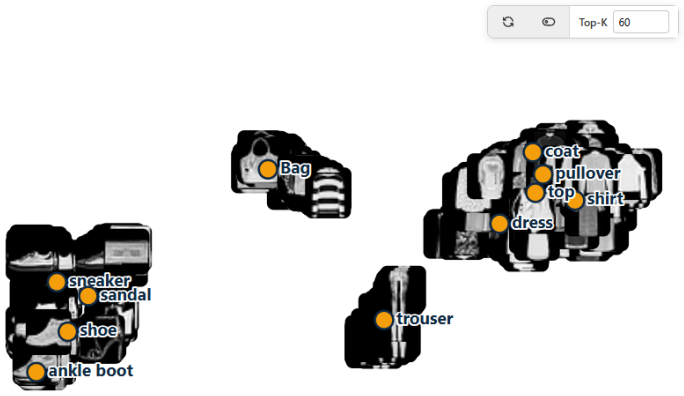


Figure 6.3: Text labels projected into a UMAP scatterplot of Fashion-MNIST image embeddings, using the same pipeline as for CIFAR-100 (Section 6.1).

6.3 Generalization and Robustness

A user's vocabulary will rarely match a dataset's class names exactly. The following two experiments therefore go beyond the fixed vocabulary used in Section 6.2: how the system handles labels it was never explicitly designed for, and how sensitive placements are to how a label is phrased.

Out-of-Vocabulary Labels

We project eight out-of-vocabulary (OOV) labels — terms that do not correspond to any CIFAR-100 class — and report the nearest ground-truth cluster for each.

Table 6.2: OOV label projections and nearest ground-truth cluster.

Label	**Nearest GT cluster**	**Distance**
vehicle	bus	0.136
furniture	wardrobe	0.256
cute animal	rabbit	0.424
socks	porcupine	0.146
sport shoes	woman	0.742
sneakers	boy	0.670
running shoes	boy	0.807
winter jacket	maple_tree	0.525

Superordinate terms such as *vehicle* and *furniture* land near semantically plausible clusters (*bus*, *wardrobe*), and *cute animal* lands near *rabbit* — showing that the system generalizes beyond CIFAR-100's own 100 class names rather than only matching them exactly. The four remaining terms name articles of clothing, for which CIFAR-100 has no corresponding class, and they do not behave uniformly. Three of them — *sport shoes*,

sneakers, and *running shoes* — all converge on the same neighborhood of person-related classes (*woman*, *boy*, *boy*), suggesting CLIP associates images of shoes with images of the people who wear them rather than placing these labels at random. The other two behave more like the arbitrary, unrelated placements one might expect from a pure vocabulary gap: *socks* lands unexpectedly close (0.146) to *porcupine*, and *winter jacket* lands near *maple_tree*, neither of which has an obvious semantic explanation. This represents a vocabulary gap in the dataset — a label can only land near clusters that actually exist — but the shoe terms show that such a gap does not always mean a landing spot is meaningless: it can still be systematically informative, just not for the concept originally intended.

Figure 6.4 shows this against the backdrop of all 100 class centroids: *vehicle*, *furniture*, and *cute animal* sit close to their nearest cluster, the three shoe terms cluster together near person-related classes, and *socks* and *winter jacket* land elsewhere without an evident pattern.

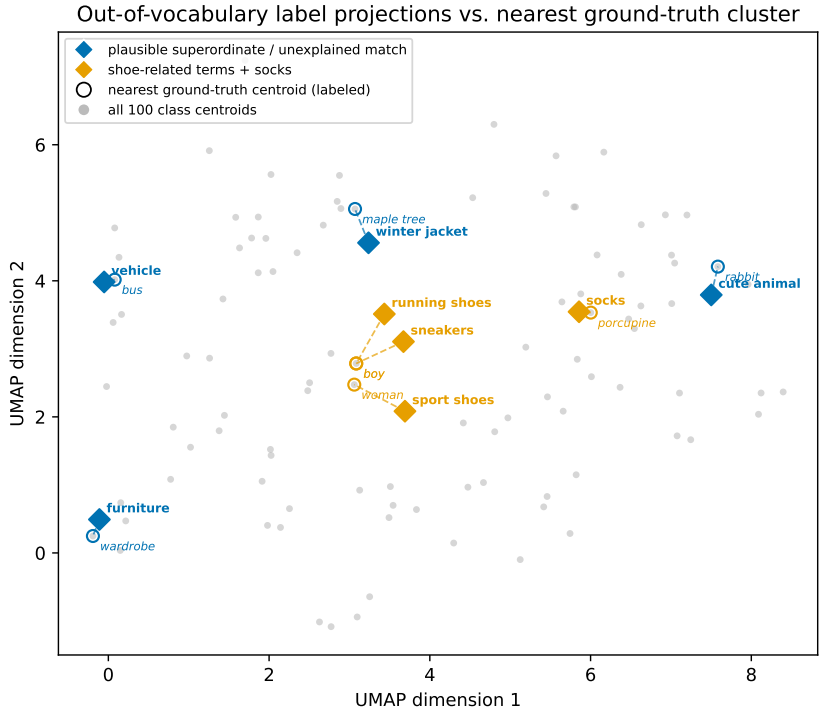


Figure 6.4: Projected positions of out-of-vocabulary labels (diamonds) against all 100 class centroids (grey dots), with a dashed line to each label's nearest ground-truth cluster. Blue: plausible superordinate matches and *winter jacket*, whose match has no obvious explanation. Orange: the three shoe-related terms, which land near person-related classes, and *socks*, whose match *porcupine* has no obvious explanation either.

Label Name Sensitivity

To test robustness to phrasing, we project synonymous label variants for six classes and measure the distances between their placements. Four of the six groups stay close together regardless of phrasing: *couch*, *bicycle*, *truck*, and *fish* variants all differ by no more than about a third of a unit, whether the difference is an underscore, added spacing, or a synonymous word choice (e.g. *truck* vs. *pickup truck*, or *fish* vs. *tropical fish*). Table 6.3 reports the full breakdown for all six groups.

The remaining two groups diverge more, though to very different degrees. The *tree* group shows moderate divergence (up to 0.60): interestingly, this is driven almost entirely by the underscore-formatted *maple_tree* — the literal CIFAR-100 class name — behaving noticeably differently from the space-separated variants *tree*, *oak tree*, and *pine tree*, even though the same underscore-vs-space change had almost no effect for *truck* (*pickup_truck* vs. *pickup truck* differ by only 0.018). The *automobile* group is the far more dramatic outlier: its variants span up to 3.67 units, roughly six times the mean intra-cluster spread. The *sedan* variant pulls farthest away from the rest. This suggests that CLIP encodes *sedan* with a narrower, more specific visual connotation than the umbrella terms "automobile" or "car," while formatting quirks such as underscores can occasionally shift a label's position by a non-trivial amount even for otherwise well-behaved classes. Both observations point to a general trade-off when writing vocabulary in this system: specific labels enable finer distinctions, but they can behave less predictably than one might expect from synonyms or minor formatting changes.

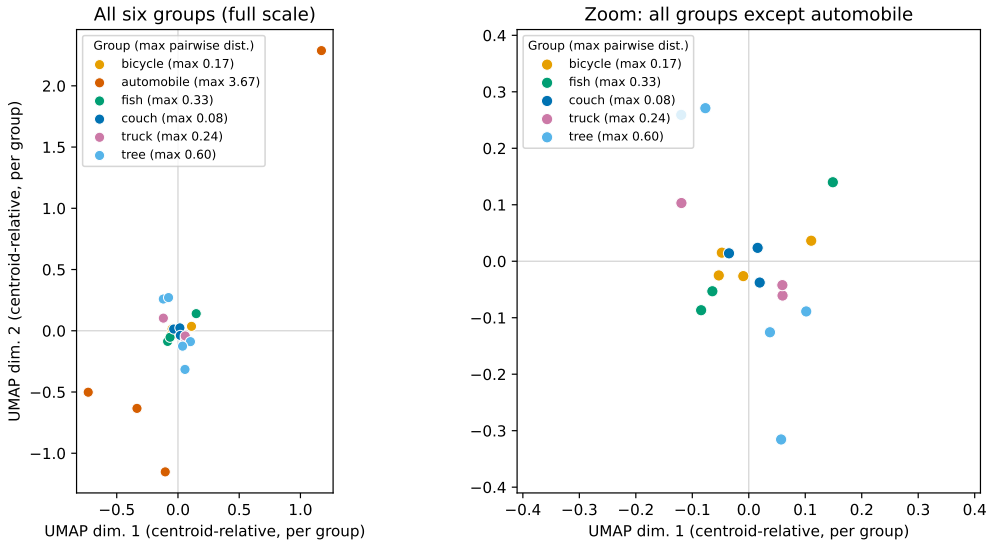


Figure 6.5: Label name sensitivity for all six tested concept groups, each recentered on its own group centroid so that spread can be compared directly across groups in a single chart. Colors are chosen to remain distinguishable for readers with color vision deficiencies. The inset zooms into the five tightly clustered groups, which are otherwise hard to tell apart next to the much larger *automobile* outlier.

Table 6.3: Pairwise distances between label name variants, all six tested groups.

Group	Variants	Min	Max
couch	couch, sofa, living room sofa	0.052	0.075
bicycle	bicycle, bike, mountain bike, road bike	0.041	0.175
truck	pickup_truck, truck, pickup truck	0.018	0.243
fish	aquarium_fish, fish, tropical fish	0.039	0.325
tree	maple_tree, tree, oak tree, pine tree, maple tree	0.044	0.602
automobile	automobile, car, sports car, sedan	0.419	3.668

6.4 Computational Performance

Accuracy alone does not make label placement useful if it cannot keep up with interactive use: Section 4.1 requires that mapping a vocabulary into the scatterplot stays within the interactive latency a user would tolerate while editing labels live. This section measures where that time actually goes.

Table 6.4: Runtime of label projection operations (warm calls, $N = 99$).

Operation	Mean	Std
CLIP text encoding	7.3 ms	4.4 ms
UMAP transform	13.9 ms*	—
Total per label	≈21.2 ms	—
Cold start (first call)	≈4.1 s†	n/a
UMAP fit (session init)	≈24 s†	n/a

*median; the mean is pulled upward by the single cold-start call and is not representative of warm-path latency. †Each occurs exactly once per session; the value is a single measurement, rounded to two–three significant figures to avoid implying a precision that a one-off timing cannot support, and no standard deviation can be computed from it.

The warm-path latency of approximately 21 ms per label is well within interactive response thresholds. For a typical vocabulary of 10–20 labels, the total projection time remains below 450 ms. The dominant bottleneck is the first `umap_model.transform()` call (∼4.1 s for a cold start), after which all subsequent calls stabilize in the low tens of milliseconds. UMAP fitting at session initialization (∼24 s) is a one-time cost.

Repeating this measurement without a fixed random seed yields the same order of magnitude for each number in this table. This is consistent with a cost dominated by model loading and matrix size rather than the specific images or labels involved.

CHAPTER 7

Conclusions

7.1 Discussion

Examining the evaluation results from Chapter 6, it is evident that placed text labels tend to be close to their corresponding image clusters, though there is noticeable variation. Aggregated over the $N \times M$ resampling protocol, the mean label-to-centroid distance across all 100 CIFAR-100 classes was $\bar{d} = 0.61$, which is close to the average intra-cluster spread of 0.65 measured in the same 2D UMAP projection space. On average, a label ends up about as far from its cluster's center, in the projected scatterplot, as a typical image of that class already is. This provides a coarse but usable signal rather than highly precise alignment.

It is important to note what this distance actually measures: it is computed entirely within the two-dimensional UMAP layout, after both image and text embeddings have already passed through CLIP's high-dimensional embedding space and been projected down. A large label-to-centroid distance could in principle originate from either step — CLIP might place the label's high-dimensional embedding far from the true concept, or UMAP might project two embeddings that are close in 768 dimensions to distant points in 2D. The evaluation in this thesis cannot distinguish between these two sources of error, since only the combined, projected result is observed (see Limitations below).

Analyzing where labels go wrong is more insightful than looking at average distances. The worst-performing classes were almost exclusively people-related categories (e.g., *man*, *boy*, *baby*) alongside a few visually diffuse or heterogeneous categories (e.g., *sea*, *ray*). A plausible explanation is that these categories are inherently close to one another, both visually and semantically: a photograph of a baby and a photograph of a boy share far more visual content than, say, a photograph of a truck and a photograph of a rose, and CIFAR-100 itself subdivides people into several fine-grained classes [Kri09]. However, this thesis cannot verify whether that closeness originates primarily in CLIP's

high-dimensional embedding space, is introduced or amplified by the subsequent UMAP projection, or both — for the same reason noted above, only the combined, projected result is observed. What the evaluation does show is that no amount of projection tuning could be expected to fully separate categories that start out this close, regardless of which step is ultimately responsible.

This trend is further supported by the out-of-vocabulary results, which demonstrate that superordinate labels such as *vehicle* or *furniture* land near plausible clusters. Yet these explicit classes do not exist in the CIFAR-100 dataset [Kri09]. This shows that the system generalizes well beyond its predefined vocabulary. Footwear terms tell a more nuanced story: CIFAR-100 has no footwear class either, yet *sport shoes*, *sneakers*, and *running shoes* all converge on the same neighborhood of person-related classes rather than landing at random, suggesting CLIP associates shoe imagery with the people who wear them. *Socks*, by contrast, lands near *porcupine* with no apparent explanation. This represents a vocabulary gap in the dataset rather than a failure of the projection itself, because a label can only land near clusters that actually exist — but a gap in the dataset's vocabulary does not automatically mean the resulting placement is meaningless.

Additionally, three smaller observations point in the same direction:

- Rewording a label barely alters its position (e.g., *truck* vs. *pickup truck* shifts by just 0.16 units). A notable exception is *sedan*, whose projected position differs noticeably from *automobile* — consistent with *sedan* carrying a narrower meaning than *automobile* — highlighting the importance of precise vocabulary selection.
- Aggregating over the full $N \times M$ resampling protocol (Section 6.1) confirms that the worst-performing classes are largely stable: four of the five hardest classes are the same whether measured from a single run or averaged over 200 runs, though the fifth position is more sensitive to which images happen to be sampled. This suggests that the core difficulty — people-related and visually diffuse categories — is an inherent property of the combined embedding-and-projection pipeline rather than a sampling artifact, even if the exact ranking at the margin is not perfectly deterministic.
- The system's runtime remains highly practical. After a one-time cold start, encoding and projecting a label takes roughly 21 ms, which is within the real-time budget specified in Section 4.1.

7.2 Answering the Research Questions

RQ1 asked to what extent zero-shot models can place user-provided vocabulary meaningfully into similarity-preserving scatterplot visualizations. The results indicate that they can do so to a meaningful but limited extent. Labels align reliably for visually distinct categories and generalize successfully to unseen vocabulary. However, no amount of projection tuning can fix the underlying overlaps in CLIP's own high-dimensional

embedding space, and, as discussed above, this thesis cannot separate how much of the remaining error is contributed by CLIP versus by the UMAP projection step itself.

RQ2 asked how labels can be used as a direct interactive exploration method in scatterplots. This was addressed through interaction design rather than an empirical user study: the interface for concept creation (Section 4.4) allows users to define vocabulary and see it placed into the scatterplot immediately, and the label-driven spatial regrouping lets users inspect which images a label is actually associated with. The evaluation results in Chapter 6 illustrate this in practice — for instance, the out-of-vocabulary and label-sensitivity experiments are themselves examples of this exploration method in use. That said, it remains an intentional design claim rather than a measured outcome whether this interaction loop actively improves user understanding compared to a static, non-interactive layout, as it was not tested with real users.

The thesis proposal originally included two further research questions, on comparing preprocessing and clustering strategies, and on comparing different zero-shot models. Both were scaled back during implementation. The final system uses a single, fixed pipeline (one CLIP variant and cosine-distance UMAP) to prioritize a thorough evaluation of that one pipeline over a superficial comparison across several. Two design choices were critical for system stability: normalizing embeddings to unit length and fitting the projection once per session rather than per interaction. Whether alternative models or preprocessing strategies would shift this balance remains an open question, and is picked up again in the Future Work below.

7.3 Limitations

In short, the system's greatest strength is its flexibility. It is ready to use right out of the box and does not require any labeled training data [RKH+21]. By projecting user-defined text directly into the same embedding space, any input can immediately serve as a candidate annotation. This approach successfully meets the core requirements of zero-shot annotation and real-time interaction, as outlined in Chapter 4.

This setup introduces clear structural trade-offs and limitations, however.

- **Unseparated error sources.** As discussed above, the evaluation measures the combined effect of CLIP's high-dimensional embedding and UMAP's 2D projection. It cannot attribute a given placement error to either step individually.
- **Model dependencies.** The system relies heavily on its base architectures. CLIP works well for distinct concepts, but it struggles with abstract, overlapping, or entirely missing categories.
- **Real-time performance vs. quality.** To maintain fluid interactions, the UMAP projection is only fitted once on a sample and then reused (Algorithm 4.2). This saves vital computing time, but provides a slightly less complete view of the overall embedding space.

- **Stability vs. flexibility.** Fixing image positions prevents a scatterplot from constantly jumping when editing vocabulary. Yet the layout cannot adapt dynamically if a new vocabulary structure requires a different layout. Furthermore, since UMAP preserves local neighborhoods rather than exact distances, a label's position depends entirely on the specific set of points projected together.
- **Balanced datasets only.** All experiments in Chapter 6 use datasets with an equal number of images per class. How the system behaves on imbalanced, real-world collections, where some concepts are far more common than others, is untested.
- **Small, user-defined vocabulary.** The system was evaluated with vocabularies of at most a few dozen hand-picked labels. It does not address the case of a large, pre-existing vocabulary the user might want to search or filter through, which would raise new challenges around suggesting a reasonable subset of labels to place.
- **Label overlap at scale.** With many labels placed at once, anchor points and their text can visually overlap, and the current implementation does not resolve this beyond rendering anchors at the highest z-order (Section 4.3).
- **Dataset generality.** CIFAR-100 is a generic, well-understood benchmark that general-purpose models such as CLIP are known to handle well. Domain-specific datasets, for example in medical imaging, are likely to be far more challenging for an out-of-the-box, general-purpose vision-language model, and would probably require a specialized, fine-tuned encoder instead.

This thesis does not present a flawless annotation tool. Rather, it proves that projecting text directly onto an image scatterplot is a fast, workable way to make clusters interpretable.

7.4 Future Work

While the current system demonstrates the viability of zero-shot label placement in scatterplots, there are several avenues for future exploration.

The most direct follow-up is to isolate the two error sources discussed above: a controlled comparison against an alternative low-dimensional projection, such as t-SNE, or against different UMAP hyperparameters, would clarify how much of the observed placement error is due to CLIP's embedding space itself versus the projection step. The $N \times M$ resampling protocol introduced in Section 6.1 would extend directly to such a comparison, by substituting the projection method between runs instead of only the UMAP seed. More broadly, a comparative evaluation of different vision-language models and preprocessing strategies would help clarify whether the failure patterns observed in Chapter 6 are universal or specific to the current pipeline.

Expanding the evaluation scale is also essential. Testing the pipeline on larger, more diverse, and imbalanced datasets beyond CIFAR-100 and Fashion-MNIST would reveal how

well accuracy holds up in domains further removed from common web imagery. To make the tool more robust for non-expert users, we should implement out-of-vocabulary handling and automated spell-checking to prevent projection failures caused by unrecognized terms.

On the vocabulary side, a natural extension is to draw label suggestions from a large, pre-existing vocabulary rather than relying entirely on user-typed terms, helping users discover concepts they might not have thought to search for themselves. Relatedly, better handling of large numbers of simultaneously placed labels — for instance, using existing label-placement algorithms to resolve overlap — would make the interface more usable in vocabulary-heavy sessions.

Technically, the system workflow could expand by converting the manual, one-label-at-a-time focus-score mechanism into an automated classification pipeline that predicts top-1 labels across all classes using calibrated confidence thresholds. Additionally, future iterations could explore label-aware layout recomputation, which would allow the spatial projection to dynamically warp and adapt as the user's vocabulary grows. This would trade some visual stability for a more accurate conceptual structure. More broadly, alternatives to scatterplots altogether, such as Star Coordinates, could be explored as a different way of relating a user-defined vocabulary to the underlying data, rather than the whitespace-expansion approach used in this thesis (Section 4.4).

Finally, the core design claims surrounding **RQ2** must be validated through a formal user study. Directly comparing our interactive loop to static or automatically labeled scatterplots is necessary to determine if user interactivity improves data comprehension.

Overview of Generative AI Tools Used

The following generative AI tools were used under the author's direct supervision during the creation of this thesis:

Claude was selectively used as an editing and brainstorming assistant. It supported the refinement of the chapter structure, formatting, and condensation to improve clarity. It also assisted in drafting the abstract, resolving LaTeX compilation errors, and generating code for TikZ-based diagrams based on structural descriptions provided by the author.

DeepL was used for language refinement, spelling corrections, and rephrasing individual sentences within the introductory and non-technical sections of the thesis.

Coding assistants (GitHub Copilot and Codex) were used exclusively for boilerplate code generation, syntax completion, and debugging during system implementation and evaluation scripting.

All AI-assisted content, code, and visualizations were critically reviewed and verified for factual and technical accuracy by the author, who also performed manual edits. No text passages or core scientific arguments were generated automatically without substantial revision.

List of Figures

1.1 The UMAP scatterplot of image embeddings from CIFAR-100 is shown before and after projecting user-defined text labels into it. Without labels (a), the viewer has to inspect the images in each cluster to identify what each cluster of thumbnails represents. In (b), however, each cluster is labelled directly, including those for concepts that the dataset was never explicitly labelled with, such as "vehicle" or "a picture of a lion". 1

3.1 CLIP maps images and text into a shared embedding space, enabling direct comparison via vector similarity. 6

4.1 System overview: image features are extracted, normalized, and persisted offline; a dimensionality reduction is fitted once on this store to produce a stable 2D layout; user-provided text labels are then encoded, normalized, and projected into that same 2D space at runtime. 8

4.2 The core interaction loop: inspecting the images in the scatterplot informs which concepts the user adds to the concept list (*discover concepts*); projecting that list back into the scatterplot then shows where those concepts are located among the images (*map labels*), which in turn informs further discovery. . 11

4.3 Label-driven spatial regrouping: (a) without regrouping, only the anchor marks the concept; (b) with regrouping enabled, associated images move toward the anchor. 13

5.1 System architecture: client-server communication between the React frontend and the FastAPI backend. 15

5.2 The JDE Sandbox layout, composed of a DataView, FrameView, and KnowledgeView. This general layout is part of the existing framework and was not modified for this thesis. 16

5.3 The UMAP scatterplot view of the concepts *crab* and *spider* with both anchors placed at the bottom right, shows the ground-truth cluster membership via colored image borders and active top-k support highlighting control (`Top-K` control, top right). 16

5.4 Sequence of a single label projection triggered from the Concept Map: the frontend sends the edited vocabulary to the backend, which encodes it via CLIP and projects it via the already-fitted UMAP model before returning updated positions. 17

6.1 Schematic illustration of the intra-cluster spread metric for one well-positioned class (*pickup_truck*) and one class where the label lands just outside the cluster (*ray*). The circle radius (intra-cluster spread) and the label's distance from the ground-truth centroid are the real, measured values from Table 6.1; the individual gray points are a schematic illustration of how tightly a class's own images cluster around their centroid, not their actual 2D positions. 21
6.2 Projected label positions (triangles) versus ground-truth class centroids (circles) for the five best- and five worst-placed classes from Table 6.1, shown in blue and orange respectively. Connecting lines show the label-to-centroid error. 22
6.3 Text labels projected into a UMAP scatterplot of Fashion-MNIST image embeddings, using the same pipeline as for CIFAR-100 (Section 6.1). . . . 23
6.4 Projected positions of out-of-vocabulary labels (diamonds) against all 100 class centroids (grey dots), with a dashed line to each label's nearest ground-truth cluster. Blue: plausible superordinate matches and *winter jacket*, whose match has no obvious explanation. Orange: the three shoe-related terms, which land near person-related classes, and *socks*, whose match *porcupine* has no obvious explanation either. 24
6.5 Label name sensitivity for all six tested concept groups, each recentered on its own group centroid so that spread can be compared directly across groups in a single chart. Colors are chosen to remain distinguishable for readers with color vision deficiencies. The inset zooms into the five tightly clustered groups, which are otherwise hard to tell apart next to the much larger *automobile* outlier. 25

List of Tables

2.1 Positioning of this thesis relative to manual and automatic scatterplot annotation approaches. 4

4.1 Notation used in Algorithms 4.1–4.4. 9
4.2 Computational complexity and typical runtime cost per phase. 13

6.1 Best and worst label placements by mean distance to ground-truth centroid, aggregated over the $N \times M = 200$ resampling runs (CIFAR-100). 21
6.2 OOV label projections and nearest ground-truth cluster. 23
6.3 Pairwise distances between label name variants, all six tested groups. . . . 26
6.4 Runtime of label projection operations (warm calls, $N = 99$). 26

List of Algorithms

4.1 Offline Image Feature Extraction 9
4.2 Session Initialization — Projection Fitting 10
4.3 Runtime — Text Label Projection 11
4.4 Runtime — Focus Score Computation 12

Bibliography

[CKB08] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31, 2008.

[CMS95] Jon Christensen, Joe Marks, and Stuart Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[Fur86] George W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*, pages 16–23, 1986.

[JSL+25] Qi Jiang, Guodao Sun, Tong Li, Jingwei Tang, Wang Xia, Yunchao Wang, Li Jiang, and Ronghua Liang. AutoMA: Automated generation of multi-level annotations for time series visualization. In *2025 IEEE 18th Pacific Visualization Conference (PacificVis)*, pages 80–90, 2025.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[RBL+17] Donghao Ren, Matthew Brehmer, Bongshin Lee, Tobias Höllerer, and Eun Kyoung Choe. ChartAccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 230–239, 2017.

[RDQR24] Md Dilshadur Rahman, Bhavana Doppalapudi, Ghulam Jilani Quadri, and Paul Rosen. A survey on annotations in information visualization: Empirical insights, applications, and challenges. *arXiv preprint arXiv:2410.05579*, 2024. Also published at IEEE VIS 2025, DOI: 10.1109/VIS60296.2025.00053.

[RHLM25] Donghao Ren, Fred Hohman, Halden Lin, and Dominik Moritz. Embedding Atlas: Low-friction, interactive embedding visualization. In *2025 IEEE Visualization and Visual Analytics (VIS)*, pages 191–195, 2025.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[SOWC24] Danqing Shi, Antti Oulasvirta, Tino Weinkauf, and Nan Cao. Understanding and automating graphical annotations on animated scatterplots. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pages 212–221, 2024.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[Wol24] Dominik Wolf. Joint human-machine data exploration sandbox. Technical report, TU Wien, September 2024.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[ZLC+19] Ying Zhao, Feng Luo, Minghui Chen, Yingchao Wang, Jiazhi Xia, Fangfang Zhou, Yunhai Wang, Yi Chen, and Wei Chen. Evaluating multi-dimensional visualizations for understanding fuzzy clusters. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):12–21, 2019.