universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Interpreting Deep Clustering Results

verfasst von | submitted by

Alexander Peikert B.Sc.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von | Supervisor:	Univ.-Prof. Torsten Möller PhD

Acknowledgements

I would like to thank Torsten for his valuable mentorship throughout this thesis, as well as Aleks and Daniel for their helpful input. I am also grateful to the domain experts and all participants of my usability study for their contributions to this work. Finally, I thank my parents for enabling my studies in Vienna and Marie-Christin for her support during this time.

Abstract

Deep clustering combines deep learning with clustering techniques to identify patterns in complex datasets. However, intermediate representations, such as the latent space, remain difficult to interpret, and existing tools provide limited support for model comparison during training. This thesis investigates how interactive visualizations can enhance latent space interpretability and assist domain experts in evaluating deep clustering algorithms. A visualization dashboard, named Clustorama, was designed and developed based on a requirements analysis with domain experts. The dashboard integrates multiple coordinated views, interactive filtering, dimensionality reduction options, metric visualizations, and misclustering detection. Usability and effectiveness were evaluated through a user study with participants performing tasks, like model filtering and misclustering identification, on a benchmark data set. Quantitative results showed an average task success rate of 89% and a System Usability Scale score of 74.16 (Grade B). Qualitative feedback indicated that the dashboard enabled deeper insights into latent space dynamics, improved comparison of different deep clustering models, and integrated multiple analysis tasks into a single tool. The findings suggest that Clustorama successfully addresses the identified requirements, improving the interpretability of deep clustering processes for expert users. Future work includes adding more deep clustering algorithms, improving scalability for large datasets, and enhancing guidance features to support non-expert users.

Kurzfassung

Deep Clustering kombiniert Deep Learning mit Clustering-Techniken, um Muster in komplexen Datensätzen zu identifizieren. Allerdings sind Zwischenrepräsentationen, wie beispielsweise der latente Raum, nach wie vor schwer zu interpretieren, und bestehende Tools bieten nur begrenzte Unterstützung für den Modellvergleich während des Trainings. Diese Arbeit untersucht, wie interaktive Visualisierungen die Interpretierbarkeit des latenten Raums verbessern und Domänenexperten bei der Bewertung von Deep-Clustering-Algorithmen unterstützen können. Ein Visualisierungs-Dashboard namens Clustorama wurde auf Grundlage einer Anforderungsanalyse mit Fachexperten entworfen und entwickelt. Das Dashboard integriert mehrere koordinierte Ansichten, interaktive Filterung, Optionen zur Dimensionsreduktion, Visualisierungen von Metriken und die Erkennung von falsch geclusterten Datenpunkten. Die Benutzerfreundlichkeit und Effektivität wurden anhand einer Benutzerstudie bewertet, bei der die Teilnehmer Aufgaben wie Modellfilterung und Identifizierung von falsch geclusterten Datenpunkten anhand eines Benchmark-Datensatzes durchführten. Die quantitativen Ergebnisse zeigten eine durchschnittliche Erfolgsquote von 89% und einen System Usability Scale Score von 74,16 (Note B). Qualitatives Feedback zeigte, dass das Dashboard tiefere Einblicke in die Dynamik des latenten Raums ermöglichte, den Vergleich verschiedener Deep-Clustering-Modelle verbesserte und mehrere Analyseaufgaben in einem einzigen Tool integrierte. Die Ergebnisse deuten darauf hin, dass Clustorama die identifizierten Anforderungen erfolgreich erfüllt und die Interpretierbarkeit von Deep-Clustering-Prozessen für erfahrene Benutzer verbessert. Zukünftige Arbeiten umfassen das Hinzufügen weiterer Deep-Clustering-Algorithmen, die Verbesserung der Skalierbarkeit für große Datensätze und die Erweiterung der Hilfestellungsfunktionen zur Unterstützung von Nicht-Experten.

Contents

Acknowledgements i

Abstract iii

Kurzfassung v

List of Tables xi

List of Figures xiii

List of Algorithms xv

Listings xv

1. Motivation 1
1.1. Contributions . . . 3

2. Methodology 5
2.1. Domain Experts . . . 6
2.2. Meetings . . . 6
2.2.1. Evaluation . . . 7
2.3. Development . . . 7

3. Related Work 9
3.1. Deep Clustering . . . 9
3.2. Clustering Visualization . . . 10
3.3. Deep Learning Visualization . . . 13

4. Background 17
4.1. Dimensionality Reduction . . . 17
4.2. Clustering . . . 18
4.3. Deep Learning . . . 19
4.4. Deep Clustering . . . 20
4.5. Data Visualization . . . 20
4.6. Explainability . . . 23
4.7. Usability Study . . . 23

5. Requirement Analysis 25
5.1. Users . . . 25

5.2. Data . . . 25
5.2.1. Data to be visualized . . . 25
5.2.2. Data (pre)-processing steps . . . 26
5.3. Tasks . . . 28
5.4. Design Goals . . . 29
5.5. Prototypes . . . 30
5.5.1. Low-fidelity Prototypes . . . 30
5.5.2. High-fidelity Prototypes . . . 49
5.6. Implementation . . . 54
5.6.1. ClustPy . . . 55

6. Introduction to Clustorama **57**
6.1. Start Screen . . . 57
6.2. Sidebar . . . 60
6.3. Model Table . . . 60
6.4. Bar Chart Table . . . 61
6.5. Bar Chart Matrix . . . 61
6.6. Line Chart . . . 61
6.7. Line Chart Matrix . . . 62
6.8. Scatter Plot before Training . . . 62
6.9. Training Scatter Plot . . . 63
6.10. Scatter Plot Matrix . . . 64
6.11. Confusion Matrix . . . 64
6.12. Design Decisions . . . 65

7. Use Case Scenarios **69**
7.1. $RQ_{Cluster}$ Scenario 1: Comparison and Optimization of Clustering Algorithms 69
7.2. $RQ_{Cluster}$ Scenario 2: Analyzing the Influence of Hyperparameters on Clustering Performance . . . 70
7.3. RQ_{Latent} Scenario 3: Understanding Embedding Dynamics in Latent Space During Training . . . 70
7.4. RQ_{Latent} Scenario 4: Investigating the Behavior of Individual Data Points in Latent Space During Training . . . 71

8. Evaluation **73**
8.1. Data Set . . . 73
8.2. Tasks . . . 73
8.3. Users . . . 74
8.4. Questionnaire . . . 74
8.5. Procedure . . . 75
8.6. Prototype . . . 75
8.7. Metrics . . . 76
8.8. Quantitative Results . . . 77
8.8.1. Success Rate . . . 77

8.8.2. Time on Task . . . 78
8.8.3. SUS . . . 78
8.9. Qualitative Results . . . 79
8.9.1. Bugs . . . 79
8.9.2. Uncertainties . . . 79
8.9.3. Feature Request . . . 80
8.9.4. Usability Difficulties . . . 80
8.9.5. Positive Feedback & Observations . . . 81
8.9.6. Resolved Issues after Usability Study . . . 82

9. Discussion **83**
9.1. Pitfalls . . . 83
9.2. Future Work . . . 84
9.3. Conclusion . . . 84

Bibliography **87**

A. Appendix **95**

List of Tables

4.1. Anscombe's Quartet: Data 21
4.2. Anscombe's Quartet: Mean, Variance, and Correlation 21
4.3. Channels ranked by Effectiveness from best to worst [Mun14] 23
4.4. Stages of Usability Testing [Laz05] 24

5.1. Pre-processing components I 26
5.2. Pre-processing components II 27
5.3. Prototype Overview 30

8.1. Test models for evaluation 73
8.2. Participants of the Usability Study 74
8.3. Success Rate for different users (1 = Success, 0.5 = Success with help, 0 = Failure) 78
8.4. Time on Task for different users (in Minutes: Seconds) 78
8.5. SUS score 79
8.6. Resolved Issues after the Usability Study 82

List of Figures

2.1. Design Study Methodology[SMM12] 5
2.2. Timeline following the Design Study Methodology 6
2.3. Rapid Prototyping Development 8

4.1. Anscombe's Quartet [RSVR18] 22

5.1. Low-fidelity prototype with embeddings and clustering view 31
5.2. In-depth view of metrics and clustering creation of forward pass of 5.1 . . 32
5.3. Low-fidelity prototype with juxtaposed clustering and metrics 34
5.4. Low-fidelity prototype with embedding and clustering split 35
5.5. Low-fidelity prototype with clustering scatter plot matrix 37
5.6. Low-fidelity prototype with clustering scatter plot matrix and textual clustering description 38
5.7. Low-fidelity prototype allowing multiple embedding log file uploads 39
5.8. Low-fidelity prototype with different tabs, here for embedding comparison of different models 41
5.9. Tab for clustering with visual separation measures 42
5.10. Tab for embedding clustering optimization 43
5.11. Tab for latent space and normal space comparison 44
5.12. After uploading models 46
5.13. After filtering 46
5.14. After selecting scatter plot 47
5.15. After opening options menu 47
5.16. First high-fidelity prototype 49
5.17. Second high-fidelity prototype 51
5.18. High-fidelity prototype for evaluation 53
5.19. Final Prototype 54

6.1. Zoomed in on existing Components on Start Screen 57
6.2. Clustorama with filters and epoch zero scatter plot 58
6.3. Clustorama with confusion matrix and line charts 59

8.1. High-fidelity prototype on which the evaluation was conducted 76
8.2. SUS score evaluation [BKM08] 77

Listings

5.1. Pseudocode Logger Class 56

A.1. Logger Class 95

1. Motivation

Deep embedded clustering, often referred to as deep clustering, is an emerging research area that integrates clustering methods with deep learning. This combination enables the automatic extraction of features from data, leading to enhanced clustering performance. A deep neural network obtains a deep representation of data with a smaller dimensionality than the original data. Instead of decoding the data, a clustering algorithm is applied after encoding in the so-called latent space, where the latent dimension is significantly smaller than the original. Afterwards, these clustered data points are reduced to a lower dimension with known dimensionality reduction techniques so that they can be visualized. Existing deep clustering models and algorithms are difficult to interpret since they act like black boxes, obscuring the connection between the clustering outcome and the underlying data structure. Traditional clustering methods like k-Means, DBSCAN, or hierarchical clustering utilize simple distance metrics between data points. In contrast, deep clustering networks learn to break down the dimensionality while minimizing reconstruction and clustering losses. The latent representation is also hard to interpret, although it carries much meaningful information. This leads to the following research questions :

1. RQ_{latent} How can we make the latent space more interpretable to gain further information for training and optimization?

2. $RQ_{cluster}$ How can the deep clustering process be supported to increase the understanding and make it more efficient?

Exploring the high-dimensional latent data space enables one to extract meaningful latent features (RQ_{latent} 1). These features can carry information not easily revealed in the raw data. This, in turn, can give new insights to domain experts in medicine, biology, and social sciences. Deep clustering is an unsupervised approach that does not rely on predefined labels, making it helpful in exploring unknown phenomena or novel datasets. Furthermore, it focuses on data-driven patterns, so it can lead to hypotheses or discoveries that might not emerge from traditional hypothesis-driven methods or rely on predefined hypotheses about the structure or distribution of the data. Examples of hypothesis-driven methods include k-Means, which assumes clusters are spherical and of equal size, or Gaussian Mixture Models, which consider that the data is drawn from a mixture of Gaussian distributions. Therefore, interpretable deep clustering can reveal latent patterns and structures that can drive forward discoveries across various fields of interest, such as medicine, biology, and social sciences [KBZ+21]. A brief overview of relevant algorithmic foundations for deep clustering algorithms, related concepts, and data visualization is provided in chapter 4. In medicine, this includes the prediction of future illness from genomic data, diagnosis of critical diseases by analyzing health records

at earlier stages, recognizing the crucial illness in advance, and smart clinical decision-supporting models [SSJR22]. Whereas in biology, such interpretable deep clustering can be used to help with gene clustering [HZZL02], biological sequence and structure analyses, and genome data analysis [ZHS+23]. In the field of social sciences, clustering approaches are, for example, utilized as a tool for crime analysis and prediction [ANS13].

Besides the step of latent space investigation and interpretation, an optimization of the clustering process on this latent space representation will be carried out in this research ($RQ_{cluster}$ 2). In a deep clustering process, a standard clustering algorithm is applied to a found latent representation [TZG17]. Thousands of clustering algorithms have been published with variations in their fundamental design, assumptions, target data structures, and parameters of interest [GDZ+23]. Therefore, choosing the right clustering algorithm in a deep clustering approach depends on these factors and is not trivial. Providing an efficient framework for this task can free up a big chunk of time previously occupied. To identify such a framework, a requirement analysis was conducted with domain experts, as described in chapter 5, leading to a new tool, shown in chapter 6. The overall research and development process, including the methodology and planning stages, is outlined in chapter 2.

Interpreting the process of how deep clustering results are reached, as well as the interpretation of the latent space, is challenging because complex, non-linear transformations are applied to the data by deep learning models. At each layer of a deep learning model, the data is modified in a way that is not directly interpretable, even when the final clusters are accurate. This lack of transparency and interpretability in transformations makes it challenging to relate visualized clusters to original features (RQ_{latent} 1). Therefore, deep neural networks are often considered black-box models, i.e., systems whose behavior is observed only by relating inputs and outputs. Understanding such black-box models can help enhance real-world applicability in medicine, biology, and social sciences. These insights could lead to the development of new deep clustering techniques, for example, by optimizing newly developed loss functions that are jointly optimized during model training. Joint optimization refers to simultaneously optimizing multiple objectives or losses of the deep neural network. This typically involves learning the feature representation and clustering tasks together rather than separately and consecutively. Loss functions measure how far the model's predictions are from the ground truth or desired output and usually depend on the specific task and data [RG]. Dimensionality reduction methods may distort structures or lose relationships of features in the process, which in turn impacts the interpretability of data and clusters. Then, there is the problem of visualizing high-dimensional data in two or three dimensions, which often omits details that are present in the whole embedding space, and remains the most common method for visualizing high-dimensional data. Specific tools, such as DimStiller [IMI+10], address this challenge by offering a dimensionality reduction and analysis framework through pipeline-based workflows. However, these tools do not facilitate a comprehensive investigation of the complete model training process. Therefore, it is difficult for researchers to understand how a clustering was reached and what happens during the clustering algorithm.

Another challenge is understanding a proper network architecture and determining

proper hyperparameters ($RQ_{cluster}$ 2) [YPMK23b]. Choosing the right clustering algorithm with the correct setting of hyperparameters is hard because it requires a pipeline of several computationally intensive steps. First, the shape of the obtained representation must be determined to choose the right clustering algorithm. For this task, the data has to be plotted using several different dimensionality reduction methods that must be compared beforehand. Afterward, hyperparameter tuning has to be conducted using the chosen clustering algorithm to optimize the accuracy of the deep clustering approach. Ultimately, the results must be plotted for a visual investigation. Simplifying these steps into one "pipeline" will decrease the time invested and optimize the whole procedure.

Deep clustering and data visualization are two closely associated tasks for data analysis, but are often performed independently and in an uncoordinated way by researchers at the moment [WYZ+23]. However, plenty of visual analysis tools support the analysis of clustering algorithms [BCH+22, KEV+17, YPMK23a, CD18]. Hence, this is a sign of little communication between the two communities (the clustering and the visualization community). Only tools for more general use are currently used by our domain experts [wan]. Therefore, providing researchers with a tailored tool for their needs can help them to simplify and improve their workflows. There are previously proposed solutions that interactively exploit the mapping between latent and original features to unveil the meaning of latent features while providing deeper insight into the original variables and, therefore, help to understand the relationships between the original and latent features (RQ_{latent} 1) [BRF+22].

Several approaches tackle the problem of optimizing the clustering model selection process ($RQ_{cluster}$ 2) [BCH+22, YML+15, YPMK23a, Dem17, MV15, BB19]. However, none support the integration of visual analysis of the latent space during the whole training process with mechanisms to improve the clustering approach, specifically in the context of deep clustering. Therefore, with a one-size-fits-all solution in the context of deep clustering, domain experts would not have to use multiple tools, which saves time and costs and increases their productivity. An overview of related work and existing tools can be found in chapter 3.

Current approaches in deep clustering often overlook the visualization approach, which is one way to support interpretability, with a focus on clustering performance or computational efficiency. As a result, their approaches do not utilize visualization as a tool to get insights into the clustering process. The focus of current research is primarily on proposing a new technique to increase performance and then plotting some of the specific performance results [KER+24, ZD21, WYZ+23, BWM+24, CBJD19]. My approach looks at what happens during the training of a deep clustering network.

1.1. Contributions

The six main contributions of this thesis are:

1. An overview of the new emerging field of deep embedded clustering.

2. An overview of the limitations of current non-visualization-driven approaches compared to my visualization-driven approach.
3. A requirement analysis, in cooperation with domain experts in this field and from current literature, was conducted to identify the requirements for a tailored visualization tool.
4. The design and implementation of Clustorama, a tool to ease the domain experts' work and help them understand the deep clustering process.
5. The evaluation of the developed tool.
6. A reflection on identified requirements, the tool, and possible avenues of further work.

Use cases for the tool are shown in chapter 7. The evaluation of the developed tool is presented in chapter 8. The overall lessons learned, challenges encountered, and ideas for future work are discussed in chapter 9.

2. Methodology

The tool is developed using Sedlmair et al.'s iterative design study methodology framework approach [SMM12], consisting of nine different stages and three top-level categories (see Figure 2.1).

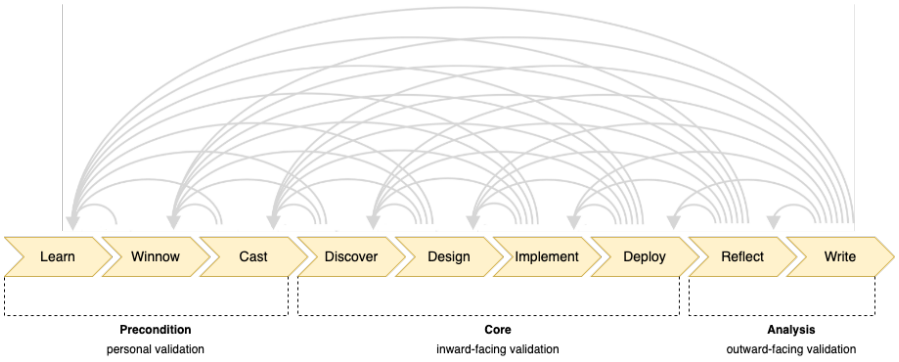

Figure 2.1.: Design Study Methodology[SMM12]

In the context of a master's thesis, I was contacted by two employees of the research group Data Mining and Machine Learning of the University of Vienna, who will serve as domain experts. First, extensive research was conducted on deep clustering and current deep clustering visualization approaches (see Learn in Figure 2.2). Problems and missing techniques to solve these problems are identified with in-depth knowledge and expertise of current tools, with the help of domain experts (see Discover in Figure 2.2). Low-fidelity prototypes are designed with different types of data transformations (e.g., dimensionality reduction or attribute permutation), visual encodings, and interaction techniques (see Design in Figure 2.2). These prototypes are presented to the domain experts to gain insights into their perspectives, reflect on designs, and develop transformations, encoding, and interactions that satisfy their tasks. Refinements were made, and new low-fidelity prototypes were developed until a joint decision was made to implement a high-fidelity prototype that could be used to analyze real-world data. These high-fidelity prototypes are again refined with the help of domain experts to improve the tool further and solve an increasing number of their challenging tasks (see Implement in Figure 2.2). At the end, a final high-fidelity prototype was handed out to conduct a usability study (see Implement in Figure 2.2). Feedback from the study is used to refine the high-fidelity prototype again, so that it can be deployed as the finished Clustorama tool. After gathering feedback, a reflection will be done and a design study paper created (see Deploy, Reflect, and Write in Figure 2.2).

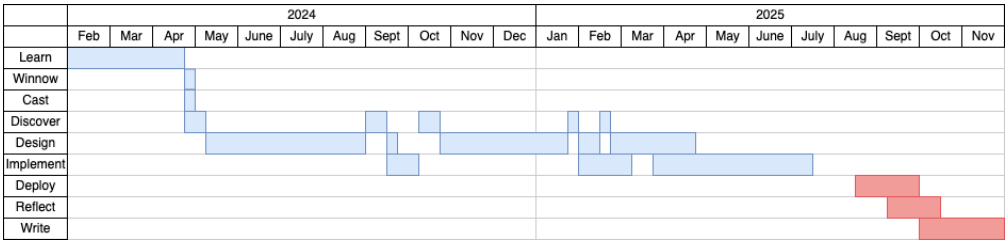

Figure 2.2.: Timeline following the Design Study Methodology

2.1. Domain Experts

The requirements were identified based on the needs of two domain experts. Both of them are from the research group Data Mining and Machine Learning of the University of Vienna. They are developing novel deep clustering methods.

2.2. Meetings

There were four individual sessions with each of the domain experts, so a total of eight meetings were held over one year. In the first meeting, the domain experts were asked about current tools in use and tasks that cannot be fulfilled with these tools, but which would be helpful to open the deep clustering black box. Further, they were asked about tasks that would provide them with valuable insights into existing methods and techniques they use. Based on these identified tasks and requirements, a total of 12 different low-fidelity prototypes with different visual encodings and interaction techniques were designed on paper. These low-fidelity prototypes were presented in the second and third meetings, and the domain experts were asked to give feedback in open questions on the developed prototypes. The open questions were related to the identified tasks and their implementation within the prototypes, as well as which prototype best suited them. Additionally, open questions were asked about the tasks to refine the understanding of these and come to an overall consensus among all parties involved. The feedback was incorporated in several rounds of refinements of the preferred prototypes to include new tasks identified in the meantime and refinement of the previously identified tasks. After the third session, a low-fidelity prototype was developed, which was satisfactory for all parties to be implemented as a high-fidelity prototype. In the fourth meeting, the high-fidelity prototype was presented with deep clustering algorithms, hyperparameter settings, and data known to both domain experts [LMPB23]. Extracted refinements based on feedback from the first high-fidelity prototype were implemented into the second high-fidelity prototype, which was handed out to the domain experts for another round of feedback.

2.2.1. Evaluation

Since an informal formative evaluation was held with the initial meetings with the domain experts, the final assessment is a more formal summative usability test on the finished high-fidelity prototypes. The domain experts received the finished high-fidelity prototypes and were asked to perform a series of tasks. These tasks include :

- Load the given training runs (.json files) and filter out the two worst ones. Which training runs did you filter out and why?
- Compare the remaining models and observe the loss curves of your chosen models. Which model has the lowest loss, and what is its value?
- Switch between the dimensionality reduction techniques and observe any changes in the embedding of the best-performing model found in T2. Which dimensionality reduction methods separate the clusters best for the given data?
- Compare the training runs of all selected models over the whole training process. Which training runs could produce similar clustering results, in terms of visual cluster separation, with a lower number of training epochs?
- Set the dimensions of the data set (8 x 8). In your best chosen model, identify a misclustered data point in the latent space and map it back to the original data. Is it misclustered because of the clustering algorithm, or is the data point of low quality?
- Display the confusion matrix of your chosen model. Which class has the least misclusterings in your preferred model?

They were observed in terms of how many of the desired tasks they could fulfill with the dashboard, how long it took to complete each task, and their level of satisfaction with the tool. Furthermore, they were asked about misleading parts within the tool.

In addition to that, four more experts from different fields were given the tool as well as the dataset and logged training runs of five different models. They are asked to use the tool, complete the tasks, and report all flaws and bugs in the dashboard from their point of view.

2.3. Development

The tool was developed using a rapid prototyping approach (see Figure 2.3) during the design and implementation phases (see Figure 2.2). First, domain experts and supervisors were interviewed to obtain and identify the most essential requirements. Then, simple low-fidelity dashboard prototypes were created "on paper", which incorporate the requirements set out by domain experts and supervisors of this thesis (see chapter 5). This leads to a fast solution that can be presented to domain experts and supervisors. These prototypes were reviewed with the domain experts and supervisors to get feedback,

which could be included in the next round of rapid low-fidelity prototyping. I reviewed this feedback, and potential solutions were derived to address the requirements and changes in a subsequent round of rapid low-fidelity prototypes (see subsection 5.5.1). The process above was repeated four times until a dashboard was agreed upon that could be implemented as a high-fidelity prototype using Vue.js and D3. Afterwards, the same process was repeated, starting with the development of low-fidelity prototypes. A more complex low-fidelity prototype emerged from the meetings and was again implemented as a high-fidelity prototype using Vue.js and D3. This prototype was evaluated through a usability study, together with deep clustering and data visualization experts.

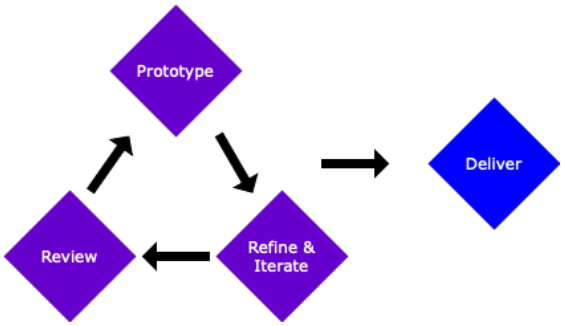

Figure 2.3.: Rapid Prototyping Development

3. Related Work

This section reviews related work on deep clustering and visualization approaches relevant to both clustering and deep clustering. It first presents the deep clustering approaches that form the basis of the visualization studies developed in this thesis. Subsequently, it examines visualization design studies that address the exploration and interpretation of clustering and deep clustering results.

3.1. Deep Clustering

A lot of research has been conducted in the field of deep clustering, with different deep clustering algorithms suiting various types of data sets and data structures: Deep single-view clustering suits data points of the same form and structure [RPY+25]. Deep clustering based on multi-view learning was developed for data points with different structures, which are often obtained from different feature collectors [RPY+25]. Deep clustering based on transfer learning, where a task with a limited amount of instances and high dimensions, can be assisted with additional information from other tasks [RPY+25]. Here, an already pre-trained model is reused as a starting point for a new model with a new task, thereby achieving a higher performance than simply training a model on a small amount of data.

Traditional clustering approaches, such as k-Means or spectral clustering, operate on the original, unprocessed data or shallow feature spaces and often underperform on high-dimensional or complex data structures. Deep Embedded Clustering (DEC) [XGF16] introduces a novel method by learning both feature representations and cluster assignments simultaneously through deep neural networks, significantly improving clustering accuracy. It initializes the model with a deep autoencoder to produce compact and meaningful embeddings before refining cluster assignments via a Kullback-Leibler (KL) divergence, discarding the decoder of the autoencoder after pretraining. This joint optimization avoids the need for reconstruction loss and focuses directly on clustering objectives. The use of a Student's t-distribution enables effective soft assignment of data points to clusters in the embedded space. Additionally, the target distribution helps to emphasize high-confidence assignments, refining clusters iteratively, and improving separation. Compared to prior methods that decouple feature learning and clustering, DEC's integrated approach yields state-of-the-art results on benchmarks like MNIST [LBBH98], STL-10 [CNL11], and REUTERS [LYRL04]. Its ability to scale linearly and generalize across domains positions it as a foundational model in unsupervised deep clustering. DEC's architecture and optimization strategy have since inspired a range of follow-up models.

Improved Deep Embedded Clustering (IDEC) [GGLY17], addresses a key limitation of

the original DEC by preserving the input data's local structure during clustering. While DEC discards the decoder after pretraining and focuses solely on optimizing clustering via KL divergence, IDEC retains the decoder and incorporates a reconstruction loss throughout training. This joint optimization of reconstruction and clustering objectives ensures that the learned latent space remains faithful to the input space, thereby producing more stable and semantically meaningful representations. IDEC's objective function adds a weighted reconstruction loss term to the clustering loss, enabling simultaneous representation learning and structure preservation. The model continues to use a Student's t-distribution for soft assignments and an auxiliary target distribution to refine clusters iteratively. Experimental results show that IDEC outperforms DEC and other baseline methods on standard benchmarks such as MNIST [LBBH98], STL-10 [CNL11], and REUTERS [LYRL04]. Notably, IDEC demonstrates improved clustering accuracy and robustness, especially on imbalanced or high-dimensional datasets. Furthermore, the integration of autoencoder reconstruction acts as a regularizer, preventing feature space distortion during clustering. This combination of local structure preservation and deep clustering makes IDEC a significant advancement in unsupervised representation learning. As such, IDEC provides a more holistic approach to clustering by unifying manifold learning and clustering within a single deep learning framework.

3.2. Clustering Visualization

Symphony [BCH+22] introduces a powerful approach to building interactive machine learning interfaces with a strong emphasis on visualization and clustering. It supports a diverse range of machine learning tasks, including dataset overview, data cleaning, and model analysis, thereby enhancing the workflows of machine learning practitioners. Symphony supports task-specific components, including clustering visualizations such as 2D projection plots of model embeddings, which enable practitioners to explore how instances naturally group in the latent space. These projection components, built using dimensionality reduction and clustering algorithms, help users visually identify clusters, outliers, and overlapping class boundaries within their data. Interactive features such as filtering, grouping, and selection are linked across components, enabling users to explore clusters from multiple analytical perspectives, for example, highlighting a subgroup in the projection view immediately reflects in confusion matrices and instance lists. Symphony's clustering-based visualizations have revealed real-world issues such as mislabeled or duplicated data and performance inconsistencies between subpopulations. These insights are crucial for debugging, improving, and communicating the behavior of machine learning models. While Symphony provides a valuable tool for clustering-based analysis and visualization, such as data duplicate discovery, its design is primarily oriented towards more general workflows. In contrast, this thesis focuses on the deep clustering training process, which Symphony does not address. Inspired by Symphony, this work adopts linked views between dimensionality-reduced projections and corresponding confusion matrices to enable intuitive cluster evaluation.

The VisRseq framework [YML+15] enables accessible and interactive analysis of se-

quencing data through rich data visualization and clustering capabilities. One of its core strengths lies in supporting standard analytical methods such as dimensionality reduction (e.g., PCA, MDS) and clustering (e.g., k-Means, hierarchical), which are tightly integrated with visual components like scatter plots, heatmaps, and dendrograms. Users can interactively explore clusters, apply filters, and link multiple views for better interpretation. For instance, users can visualize differential gene expression results, identify up- or down-regulated gene groups, and compare clustering outputs across different analysis methods. Case studies demonstrate VisRseq's utility in exploring gene expression patterns and allele-specific effects, with clustering visualizations playing a key role in pattern discovery. Its interactive scatter plots, pie charts, and subset selection tools make cluster exploration intuitive and reproducible. VisRseq's design also emphasizes extensibility, allowing developers to contribute new analysis modules with minimal coding. Although VisRseq encompasses valuable techniques for clustering and dimensionality reduction visualization, it is designed to support sequencing data. In comparison, the tool developed in this work is designed to support clustering-based model evaluation more generally within deep clustering contexts, especially on typical benchmark data sets. While the formats differ, the design of scatter plot matrices, which allows for comparing clustering outcomes across methods, such as different projections, served as inspiration for implementing scatter plot-based visual components in this work.

Clustrophile 2 [CD18] is an interactive visual analysis tool designed to support guided exploratory clustering analysis of high-dimensional data. It enables users to experiment with multiple clustering algorithms, metrics, and parameters in a flexible interface that promotes rapid iteration and exploration. The system introduces a novel feature, the Clustering Tour, which models the clustering space as an undirected graph and recommends diverse clustering results based on user feedback and analytical goals. Clustering results are visualized through a combination of scatter plots and heatmaps, with projections obtained using dimensionality reduction techniques such as t-SNE or PCA. The scatter plot conveys inter-point similarity and cluster separation, while the heatmap displays aggregated feature values across clusters. Users can dynamically filter data, isolate subsets, and interactively re-cluster selected points to identify substructures. Clustrophile 2 supports the comparison of multiple clustering instances simultaneously through coordinated views. It also incorporates automated feature selection and provides both quantitative and qualitative evaluation of cluster quality, including silhouette scores and decision-tree-based interpretability aids. The tool facilitates the understanding of uncertain data assignments by highlighting low-confidence points and suggesting alternative clusterings. Overall, Clustrophile 2 guides users in clustering-based exploratory analysis and helps to reason about differences between clusterings. Still, it completely lacks the training process analysis as well as an easy way for clustering comparison. Motivated by Clustrophile 2, this work enables easy switching between different projection techniques and allows users to utilize different clustering algorithms to support comparative analysis. The idea of presenting aggregated statistics for the models is adapted here, using histograms to aid model interpretation.

Clustervision [KEV+18] is a visual analysis system designed to support the exploration of

unsupervised clustering results through interactive visualization. It systematically applies various clustering algorithms, such as k-Means, Spectral Clustering, and Agglomerative Clustering, across a wide range of parameter settings. To evaluate and rank these results, the system employs five different quality metrics, including the Silhouette Score, Davies-Bouldin index, Calinski-Harabaz index, Gap Statistic, and S_{Dbw}, thereby offering a multi-perspective assessment of clustering quality. The Projection view uses dimensionality reduction techniques like t-SNE, PCA, or MDS to display high-dimensional data in two dimensions, allowing exploration of cluster structures. To reduce visual clutter, users can enable superpoints, which aggregate similar data points into representative markers. Additional views, such as Parallel Trends and Ranked Features, help to identify and explain which features influence cluster separation. These views use univariate statistics to rank features by importance. The Cluster Detail and Data Point views provide more granular insights into cluster cohesion, separation, and the distribution of feature values. Altogether, the system enables users to compare multiple clustering results and explore their internal structure in a visually guided manner. The dashboard does not take into account new developments such as deep clustering algorithms, but focuses on traditional clustering approaches. Clustervision exemplifies how visual interfaces can enhance the interpretability and usability of unsupervised clustering for domain experts. The approach of comparing multiple clustering results side-by-side served as an essential inspiration for this work. Furthermore, the idea of highlighting differing cluster assignments between computed and ground truth labels was adapted to support the evaluation of model consistency and clustering behavior.

ModEx [YPMK23b] introduces a flexible visual parameter space analysis tool focused on helping users understand and optimize complex algorithms, called computer models, through visualization. A central strength of ModEx lies in its interactive exploration interface, which supports a wide range of data visualizations, including scatter plots, histograms, radar charts, and parallel coordinates. In the context of clustering, ModEx enables users to sample large parameter spaces for various clustering algorithms, compute quality metrics, and explore results visually across multiple dimensions. ModEx is designed to take multiple runs of models with different parameter combinations, allowing the user to set parameter ranges to explore different parameter combinations and identify the optimal combination. Visual tools like dimensionality reduction projections and radar charts help users identify optimal cluster configurations and understand the impact of parameters on clustering quality. The system allows intuitive filtering and selection of model runs, with linked views updating in real-time to reflect chosen ranges of parameters or quality metrics. The system's ability to visually compare hundreds of clustering runs reveals hidden performance patterns and supports informed parameter exploration. All in all, ModEx bridges the gap between parameter tuning and interpretable model analysis through rich, interactive visualizations, offering a flexible architecture that is independent of specific algorithms. In contrast, the tool developed in this work targets a more specialized use case by focusing specifically on clustering model comparison and evaluation. Key ideas adapted from ModEx in this work include the use of dimensionality reduction techniques for projection-based visualizations and the comparison of clustering

runs across multiple quality metrics to enhance exploration and interpretability.

3.3. Deep Learning Visualization

A variety of visualization tools for deep learning algorithms has already been introduced. They fall in different categories, depending on the audience they address and the concepts they are visualizing. Teaching tools, like Tensorflow Playground [tfp], introduce impactful methods for teaching beginners about deep learning concepts and help them build an intuition of how deep learning works. Architecture assessment tools, like Tensorflow Graph Visualizer [tfg], comprise methods that support practitioners in visualizing network architectures. Tools for debugging and model improvement, like DeepEyes [PHVG+18], help model developers inspect the training process and improve the models. Visual explanation tools, like Grad-CAM [SDV+16], introduce studies that focus on the visual explanation of deep learning for experts [YS18].

A lot of visualization approaches in deep learning, such as SUMMIT [HPRC19], ActiVis [KAKC17], and DeepEyes [PHVG+18], focus on the internal structure of models, like activation functions and neurons. Others focus on assessing the model's training performance, like TensorFlow Model Analysis (TFMA) [tfm], TensorBoard [ten], and Weights and Biases (W&B) [wan].

ActiVis [KAKC17] introduces a scalable visualization system designed to help users interpret deep learning models' neuron activation through coordinated, interactive views. ActiVis focuses on both instance-level and subset-level exploration, enabling users to investigate how individual data points or groups of points influence neural network activations. A central visualization is the neuron activation matrix, which shows how neurons respond across different classes or subsets, revealing distinctive activation patterns. To assist in identifying clustering structures, ActiVis integrates 2D projection views using t-SNE, allowing users to observe how activations group in a lower-dimensional space. These projections reveal the clustering behavior of instances, highlighting misclassifications or overlaps between classes. Users can define custom subsets based on raw attributes or predicted outputs, and compare their average activations visually alongside individual instances. This supports the detection of patterns and anomalies that would be hard to detect in raw form. ActiVis also features a graph-based model architecture view for navigating deep network layers and selecting specific nodes for activation inspection. Overall, ActiVis lets users explore clustering and classification results on an instance level. Still, it completely lacks the evolution of clusterings and classification throughout the training, and does not let users compare multiple models simultaneously. This work was inspired by ActiVis's interactive instance-level exploration, particularly the ability to select and inspect individual data points. This idea was adopted to support outlier identification and an analysis of clustering results at the instance level within the dashboard.

DeepEyes [PHVG+18] presents an innovative system that enables in-depth analysis of neural networks during training. DeepEyes employs a progressive visual analysis approach, updating visual representations in real time as training progresses. A key contribution is the Perplexity Histogram, a novel visualization that helps identify stable layers by tracking

how consistently filters activate across input data, revealing the clustering behavior of learned patterns. Once stable layers are identified, the system provides three tightly linked visualizations, which are Activation Heatmaps, Input Maps, and Filter Maps, to explore filter behaviors and clustering of inputs. The Activation Heatmap highlights degenerated filters, aiding in the pruning of redundant or underperforming components. The Input Map visualizes receptive field activations in a 2D embedding, where clusters of similar inputs become evident, and filter responses can be analyzed interactively. The Filter Map shows how similarly different filters respond across inputs, revealing clusters of filters with overlapping functionality. These visualizations enable users to detect oversized layers, redundant filters, or underutilized input regions. By providing real-time visualizations during the training, DeepEyes supports informed architectural adjustments for neural networks. Overall, it advances model transparency and optimization in deep learning development. It focuses on the inputs and layers while ignoring the embeddings during model training. The ability to visually inspect and analyze individual input instances inspired the inclusion of image input data inspection in this work. This enables users to explore the behavior of specific data points within the clustering process and supports interpretability on the input level.

TensorFlow Model Analysis (TFMA) [tfm] is a powerful library designed to support detailed evaluation and visualization of machine learning models, particularly during and after training. It enables practitioners to analyze model performance on large-scale datasets by computing metrics over different slices of data, which helps reveal hidden biases and underperforming subgroups. TFMA provides interactive visualizations that allow users to monitor metrics across training runs, compare different model versions, and track improvements or regressions. These visual insights go beyond aggregate performance scores by showing how specific features or data segments influence the model's behavior. One of its key strengths lies in visualizing the effects of training over time and across different data subsets, making it easier to debug and refine models iteratively. This comprehensive approach to visual evaluation helps ensure that models are both accurate and reliable across real-world data distributions. By combining metric computation with intuitive visual interfaces, TFMA contributes significantly to transparent and informed model development. On the other hand, TFMA is not designed to support visualizing embeddings and clustering results. The use of metric visualizations in line charts for performance over training time inspired this work. Similar techniques are used in the dashboard to present model training dynamics and support interpretability through temporal performance tracking.

TensorBoard [ten] is a visualization toolkit within the TensorFlow ecosystem, designed to provide comprehensive insights into model training processes. The dashboard enables real-time tracking of key metrics such as loss and accuracy, facilitating the monitoring of model performance over time. It offers a detailed visualization of the model's computational graph, allowing for an in-depth understanding of the model architecture and data flow. For analyzing the evolution of model parameters, histograms and distributions display the progression of weights and biases throughout training. TensorBoard's Projector tool aids in visualizing high-dimensional embeddings by projecting them into lower-dimensional

spaces, which is particularly useful for interpreting feature representations. The Images view provides a visual representation of input data, model predictions, or intermediate outputs, which is invaluable for tasks involving image data. TensorBoard uses callbacks during model training, automatically logging relevant metrics and visualizations, making integration into the code easy. This integration simplifies the process of comparing different training runs, making it easier to assess the impact of various hyperparameters or model configurations. Moreover, TensorBoard supports the visualization of custom scalar values, offering flexibility to monitor user-defined metrics pertinent to specific projects. Overall, TensorBoard serves as a powerful tool for diagnosing issues, optimizing model performance, and enhancing the interpretability of complex deep learning models. On the other hand, it lacks functions to show embeddings over the training epochs and outlier detection. The TensorBoard toolkit inspired several aspects of this work, particularly the juxtaposition of training metrics in line charts and the visual exploration of image inputs. Additionally, the projection of high-dimensional embeddings using different dimensionality reduction techniques influenced the embedding visualization strategy used in the dashboard.

Weights and Biases (W&B) [wan] is a comprehensive platform designed to enhance the visualization and tracking of machine learning experiments during model training. It provides real-time dashboards that display key metrics such as loss, accuracy, and custom parameters, enabling users to monitor training progress effectively. The platform supports the logging and comparison of multiple training runs, facilitating hyperparameter tuning and model optimization by visualizing performance differences across experiments. W&B's integration with various frameworks allows for easy logging of model artifacts, including weights, gradients, and system metrics, which are crucial for in-depth analysis. Interactive plots and custom panels enable users to dive into specific aspects of model behavior, like overfitting or underfitting, by examining trends and anomalies in the training data. Additionally, W&B enables the user to visualize model predictions and outputs, which is beneficial for tasks involving image or text data, and supports the understanding and interpretability of the model. Furthermore, W&B's experiment tracking capabilities extend to the management of datasets and code versions, ensuring that all aspects of the training process are documented and accessible. Its robust visualization tools are instrumental in identifying potential issues early in the training process, leading to more efficient and effective model development. Unfortunately, it lets users investigate the data only through tables, therefore completely lacking the options to visualize clusters. W&B inspired key aspects of this work, particularly the ability to compare numerous experiments at once and use interactive, linked visualizations for better inspection of plots belonging to the same model.

4. Background

This chapter explains some of the background to understand this thesis better. Specific dimensionality reduction methods and clustering algorithms utilized are explained, and a general overview of deep learning, deep clustering, and data visualization is given.

4.1. Dimensionality Reduction

High-dimensional datasets with many features or observations sometimes have highly correlated dimensions. Therefore, it is necessary to incorporate all the information when, for example, visualizing the data. One problem is that plotting high-dimensional data on a two-dimensional screen is hard, since humans cannot comprehend more than three dimensions. Multiple techniques have emerged, like DimStiller, which is a system that guides the user through the process of analyzing high-dimensional data iteratively [IMI+10], and the Grand Tour, which projects a high-dimensional dataset into two dimensions for high-dimensional point clouds [LZS20, Asi85]. Parallel coordinates visualize high-dimensional data by representing each dimension as a parallel vertical axis, with data points drawn as lines intersecting these axes [HW13]. A complementary approach is the scatter plot matrix, which displays pairwise relationships between dimensions using a grid of 2D scatter plots [EDF08]. Both techniques enable users to explore correlations and structures in high-dimensional datasets without reducing them to fewer dimensions. Another technique for reducing a large set of features to a smaller one that retains the most important information is dimensionality reduction. In the following, a selection of dimensionality reduction methods utilized within my tool is presented: Principal Component Analysis, t-distributed Stochastic Neighbor Embedding, Multidimensional Scaling, and Linear Discriminant Analysis. Other popular methods, such as UMAP, Isomap, or Sequential Non-negative Matrix Factorization, are not covered in this thesis, as the domain experts favor them less and would have exceeded the scope of this work. For a broader overview of dimensionality reduction techniques, the reader is referred to relevant survey papers [Fod02, SVM14].

One linear technique for dimensionality reduction is Principal Component Analysis (PCA) [Hot33]. PCA uses an orthogonal transformation on the original features to obtain a set of uncorrelated features called the principal components (PC), which are linear combinations of the original features [Jam13]. It aims to find a linear transformation such that the variance of the data is maximized in the projected space [PB08]. Often, even a few principal components explain a big part of the variation within the data. The first PC explains most of the variation within the data, followed by the second, and so on. A graphical visualization of the first two principal components can often

help to find clusters within the data. Furthermore, PCA assumes that the underlying relationships between input data features can be captured using linear combinations and that they are numerical [Shl14]. Noisy data points can distort the result of PCA because it does not distinguish between variance within the measured phenomenon and noise [Bai12]. A principal component analysis transforms a set of correlated features into a set of uncorrelated features, thus, it is obvious to check the original data for correlation. If there is no correlation at all, PCA might be unnecessary.

Another dimensionality reduction method used within the developed tool is the nonlinear t-distributed stochastic neighbor embedding (t-SNE), which is a statistical method capable of retaining the local structure of the data while also revealing some crucial global structure [vdMH08]. It uses a symmetrized version of the SNE cost function with simpler gradients and a Student-t distribution rather than a Gaussian distribution to compute the similarity between two points in the high-dimensional space [RLJF22, vdMH08]. A similar distribution from the high-dimensional space is found in low-dimensional space [vdMH08]. The t-SNE algorithm is capable of capturing much of the local structure of the high-dimensional data while revealing global structures such as the presence of clusters at several scales [RLJF22].

A third method used in this approach is Multidimensional Scaling (MDS) [Tor52], which is also a linear method. The idea in Multidimensional Scaling is to find a lower-dimensional representation of the original data that preserves the pairwise distances [HTF17]. Two types of Multidimensional Scaling have been developed for different data, metric Multidimensional Scaling and non-metric Multidimensional Scaling [SNHMS18]. In metric MDS, a distance matrix is transformed into a set of coordinates such that the derived Euclidean distances approximate the original distances as well as possible [Abd07]. Non-metric Multidimensional Scaling is applied when the magnitude of the input dissimilarities is unreliable, too challenging to measure, or unavailable [AWC+07]. It requires magnitudes as input and concerns only ordinal information.

4.2. Clustering

Clustering or cluster analysis is the task of grouping objects that are similar to each other into the same group, called a cluster, and objects that are not similar to each other in different clusters. It is an umbrella term for a family of different algorithms applied to some data. These include data pre-processing methods, clustering algorithms, and dimensionality reduction techniques. Specific clustering algorithms differ in their understanding of the definition of a cluster, so they suit various data and have different ways to obtain a clustering. The different clustering algorithms can be subdivided into the following categories: Connectivity-based clustering, Centroid-based clustering, Model-based clustering, Density-based clustering, Spectral and Grid-based clustering.

Connectivity-based algorithms, like hierarchical clustering, were developed to provide a more deterministic and flexible framework for grouping data objects. These methods are generally classified into agglomerative and divisive approaches [AR13]. Agglomerative clustering begins with each data object as a singleton cluster and iteratively merges pairs

of clusters to construct a bottom-up hierarchical structure. In contrast, divisive clustering starts with all data objects in a single large cluster and recursively splits it into smaller clusters, forming a top-down hierarchy [AR13].

Centroid-based clustering methods like k-Means use a two-step approach, first re-assigning data points to the closest centroid, representing the cluster mean, and second re-computing the centroid. These two steps are repeated until no further change in assignments is made or a maximum number of iterations of these steps is reached [Bis06, HTF17]. The model-based clustering framework assumes data to be generated by a finite mixture model where each component represents a cluster [KR23].

Density-based clustering algorithms like DBSCAN can be considered non-parametric methods, as they make no assumptions about the number of clusters or their distribution. Dense areas in the data space are separated from each other by sparser regions [AR13]. Furthermore, the density within the areas of noise is assumed to be lower than the density in the clusters. Density-based clustering emerged from assumptions that algorithms produce spherical clusters and cannot deal well with datasets in which the actual clusters have nonspherical shapes [AR13]. Traditional clustering methods use a spherical or elliptical metric to group data points, so they will not work well when the clusters are non-convex [HTF17].

Spectral clustering can solve problems in complex scenarios, such as intertwined spirals or other arbitrary nonlinear shapes, because it does not make assumptions on the shapes of clusters [AR13]. Lastly, there are grid-based clustering methods, which are typically used for multidimensional data sets. A grid structure is created for the data space and divided into a finite number of cells, usually much smaller than the number of data points. Then, a comparison of these cells is performed, which is fast and has a low computational complexity. Rather than clustering the data points directly, grid-based approaches cluster the neighborhood surrounding the data points represented by cells [AR13].

4.3. Deep Learning

A deep-learning architecture uses a multilayered neural network trained with large data sets to solve complex information-processing tasks [BB23]. It has emerged as the most successful paradigm in the field of machine learning [BB23]. All (or most) of these neural networks are subject to learning, so adjusting their internal parameters to minimize the error between predictions and target values. They compute nonlinear input-output mappings. Each neuron in the series of layers of neurons transforms its input and passes the result to the next layer of neurons. Deep learning methods are representation-learning approaches. These representations are formed through multiple layers in a neural network, where each layer applies a transformation to its input, which is typically a linear operation followed by a nonlinear activation function. Each layer applies a transformation to the representation at one level, starting from the input. During the training, the parameters of these transformations are optimized based on a task objective, like clustering. The network learns by adjusting how it transforms data at each layer to improve performance. Deep learning models are very good at learning useful representations and intricate structures

in high-dimensional data and applying them to many application domains. Deep learning concepts perform better with increasing amounts of data and available computational resources than traditional machine learning techniques [WFW21]. We compute a loss function that measures the error (or distance) between the model's actual output and the desired output. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers [LBH15]. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted opposite to the gradient vector [LBH15].

4.4. Deep Clustering

Recent research has demonstrated encouraging clustering results by learning representations [TKC18]. In general, deep clustering approaches can be divided into three different categories. One uses two steps, where the first step is to pre-train the model to obtain a deep representation in the latent space of the input data. The second step is to cluster these deep representations with known clustering algorithms [NN19] (see section 4.2). In the second category, containing a single step, a deep neural network is initially pre-trained to minimize the loss of the learned representations and jointly optimized with a clustering loss to improve the clustering objective [TKC18]. The third category also has two separate steps, which are pre-training to obtain a deep representation and clustering on this deep representation. However, the steps alternate in an iterative loop instead of being performed in one feedforward linear fashion [NN19], where a re-training step is applied after each iteration. In all cases, clustering algorithms are performed on a deep representation instead of the original input data, which is typically lower in dimension and captures the different hidden groups within the data [NN19].

4.5. Data Visualization

"Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively" [Mun14]. The visual design space is vast and encompasses considerations for creating and interacting with visual representations. Compelling visualizations allow scientists to understand their data and communicate their insights to others [Was21]. Data visualization is a field applied in many disciplines, such as computer science [AC15]. In computer science, vis is, for example, used to explore datasets, visualize feature spaces, and evaluate model performance. Since different data sets with completely different data points (see Table 4.1) can lead to the same statistical output [Ans73] (see Table 4.2), not only a statistical analysis of data sets but a visual analysis as well needs to be conducted to get valuable insights into data sets (see Figure 4.1).

Dataset 1		Dataset 2		Dataset 3		Dataset 4	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Table 4.1.: Anscombe's Quartet: Data

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Mean (X / Y)	9.0 / 7.5	9.0 / 7.5	9.0 / 7.5	9.0 / 7.5
Variance (X / Y)	10.0 / 3.75	10.0 / 3.75	10.0 / 3.75	10.0 / 3.75
Correlation	0.816	0.816	0.816	0.816

Table 4.2.: Anscombe's Quartet: Mean, Variance, and Correlation

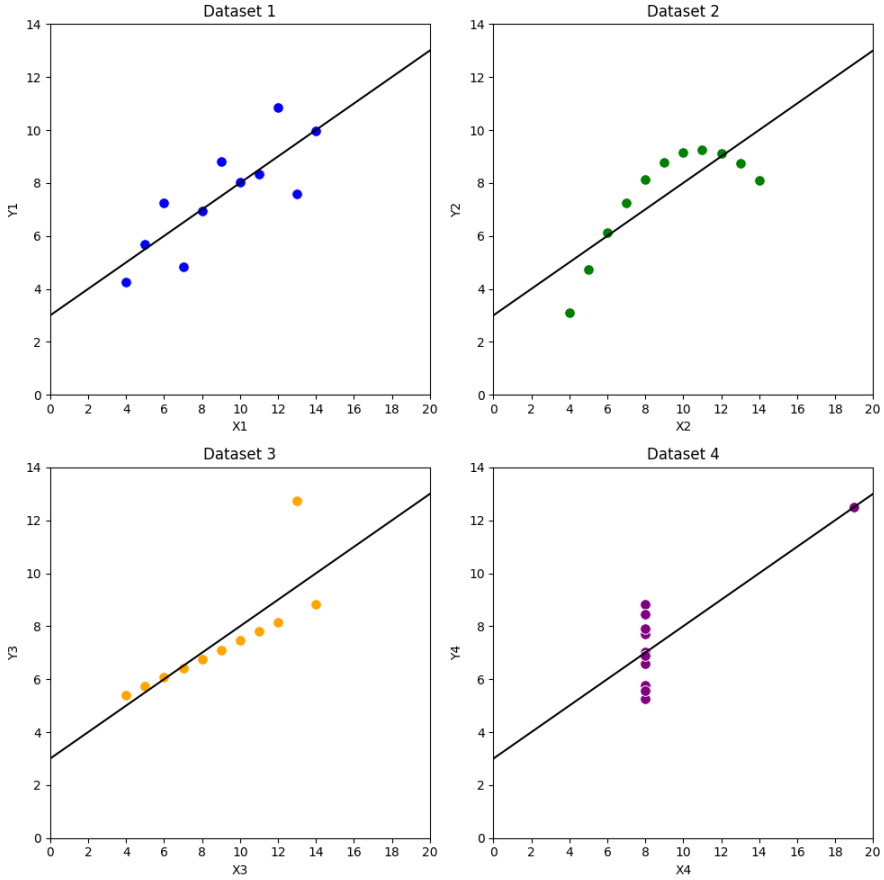

Figure 4.1.: Anscombe's Quartet [RSVR18]

The basic building blocks of a data visualization are called marks and channels. There are four elementary types of marks, which are Points (0-D), Lines (1-D), Areas (2-D), and Volumes (3-D) [Car09]. Visual channels control the appearance of marks and include Position, Color, Shape, Tilt, and Size of the marks [Mun14]. Channel effectiveness has been extensively studied over the years, which has led to the following ranking (see Table 4.3):

Magnitude Channels Ordered Attributes	**Identity Channels Categorical Attributes**
Position on common scale	Spatial region
Position on unaligned scale	Color hue
Length (1D size)	Motion
Tilt/angle	Shape
Area (1D size)	
Depth (3D position)	
Color luminance & saturation	
Curvature & Volume (3D size)	

Table 4.3.: Channels ranked by Effectiveness from best to worst [Mun14]

4.6. Explainability

As dependence on deep learning approaches continues to grow, so does the demand for more transparent and interpretable models [DDN+23, ASJ+21]. However, deep learning models are perceived as black-box methods considering the lack of understanding of their internal functioning [CSHB18]. AI researchers aim to open the black-box of neural networks and turn it into a transparent system. Currently, there are two main branches of work in Explainable AI: designing transparent models and post-hoc explanations [XUD+19]. The transparent design reveals how a model functions from the developers' point of view. It tries to understand model structure, single components, and training algorithms. The post-hoc explanation explains why a result is inferred from a user's perspective. It tries to give analytic statements, visualizations, and explanations by example [XUD+19].

4.7. Usability Study

The goal of a usability study is not to research the user, but to research the interface and involves representative participants that attempt representative tasks in a representative environment, with the participants' activities being observed and monitored [Lew06]. The general goal is to improve the quality of the dashboard by identifying areas for improvement, such as confusing or misleading parts, and areas that work well in the dashboard [LFH17]. Exploratory usability testing takes place in the early stages of development, which is known as formative usability testing. It includes easy and fast development of paper prototypes, known as low-fidelity prototypes [DF09], and is more informal with more communication between the moderator and the participants (see subsection 5.5.1). The focus is on how the users perceive the interface component, rather than on solving actual tasks with the dashboard [RC08]. On the other hand, a more formal usability test of a dashboard prototype when high-level choices have already been made is called a summative usability test [LFH17]. This is carried out on a functional dashboard prototype, called a high-fidelity prototype [DF09] (see subsection 5.5.2). Here, the following framework is utilized (see Table 4.4):

Stages of Usability Testing
Select representative users
Select the setting
Decide what tasks users should perform
Decide what type of data to collect
Before the session (informed consent, etc.)
During the session
Debriefing after the session
Summarize results and suggest improvements

Table 4.4.: Stages of Usability Testing [Laz05]

For small-sized projects, it is sufficient to let five users test the dashboard, because they will find approximately 80% of usability problems [Vir92]. Usability studies can be held synchronously in a remote online setting, which is equivalent to in-person methods. In contrast, asynchronous methods are more time-consuming since the test subjects cannot directly provide instructions if anything goes wrong [ANSS07]. Tasks should be provided to the participants and should be formulated so that the answer is not known before the task is finished. Task performance, time performance, and user satisfaction are measured to evaluate the dashboard. Additionally, user preferences, behavior, and problems are identified and used to improve the tool.

5. Requirement Analysis

This chapter analyzes the requirements for the proposed visualization tool by following a structured design study process. First, the domain situation is elaborated with an overview of the domain experts and the goal they want to accomplish with my tool (see section 5.1). Then, the data abstraction is presented, which includes the data types that can be visualized throughout the use of the tool (see section 5.2). The data types are evaluated for the original, unprocessed data, the data after pre-processing, and the embedded data in the latent space. Next, the task abstraction is conducted to show why the domain experts need this tool and what actions they want to perform on the data, which were previously not possible (see section 5.3). Afterward, the visual encodings and interactions are shown iteratively, showing how the domain experts can use the tool and how it has developed over time (see section 5.5).

5.1. Users

The tool is developed to support the work of domain experts with in-depth knowledge in deep clustering and deep learning. In particular, two domain experts from the research group Data Mining and Machine Learning of the Faculty of Computer Science of the University of Vienna provided valuable input and feedback during the development process that led to the tasks. Therefore, the tool is specifically developed for professionals who already understand deep learning and related concepts. These professionals want a much deeper understanding of deep clustering since the field is comparatively new, and not much research has been conducted on why deep clustering works. Additionally, the integration of visual analysis approaches in deep clustering is not well explored compared to more established areas in machine learning, like model performance analysis.

5.2. Data

The following section describes the type of data that is visualized by the tool. The presentation follows a step-by-step approach, beginning with the unprocessed input data and proceeding through the applied pre-processing steps to the embedded representations in the latent space.

5.2.1. Data to be visualized

Different data set types can be utilized in deep clustering. Since the original, unprocessed input data is transformed into a numeric vector in the latent space, any dataset type is

suitable as input data for this tool. The only constraint is that it can be used in deep learning approaches. These data set types can include image, text, time-series, audio, video, tabular, graph, and multimodal data. Within all these data set types, the actual attributes can have various types. These attribute types can either be categorical, so no ordering between different items is possible, ordinal with a defined ordering but without arithmetic comparisons, or quantitative, which fully support arithmetic operations.

5.2.2. Data (pre)-processing steps

Data (pre-)processing comprises several steps. Firstly, a deep neural network must be initialized with a selection of (hyper-)parameters. In addition, the labels are converted into numerical values. During the training process, the data is then normalized, and the loss and the gradient are calculated to see if the deep neural network is still learning, indicated by a decreasing loss and near-zero gradient over successive epochs. Metrics are calculated for newly found embeddings. Dimensionality reduction techniques are applied to the embeddings, which also takes place in this step. Specific techniques and parameters are described in detail in Table 5.1 and Table 5.2:

Pre-processing Components	Description	Type
Adjusted Mutual Information	Used to evaluate the quality of clusterings by quantifying the amount of information shared between two clusterings	Numeric
Silhouette Score	Measure how similar an object is to its cluster compared to other clusters; ranges from 1 to +1; a high value indicates the data points are well-clustered and separated from other clusters	Numeric
Calinski–Harabasz Index	Internal evaluation metric, based solely on data-set and clustering results, not on external ground-truth labels; it is defined as the ratio of between-cluster separation to within-cluster dispersion, normalized by degrees of freedom	Numeric

Table 5.1.: Pre-processing components I

Pre-processing Components	Description	Type
Learning Rate	Step size at each iteration while minimizing a loss function	Numeric
Dropout	A regularization technique where a number of neurons are deactivated during training at each layer; prevents overfitting	Numeric
Batch Size	Training data divided into smaller batches, each processed independently before model parameters are updated; batch size is the number of samples used in each batch during training	Numeric
Activation Function	Transforms the input of a node in a neural network into an output that is passed to the next layer; Without it, neural networks would be restricted to modeling only linear relationships between inputs and outputs	Categorical (e.g., ReLU, Sigmoid, Tanh)
Number of Epochs	One epoch is the training with all training data for one cycle, where all of the data is used exactly once in a forward pass and a backward pass	Numeric
Label-2-Index	Method that converts labels of all kinds into numerical labels	Method
MinMaxScaler	Method that transform features by scaling each feature to a given range; Doesn't reduce the effect of outliers, but scales them down to a fixed range	Method
Gradient Norm	Overall magnitude of gradients of model's parameters	Numeric
Principal Component Analysis	Orthogonal transformation on original features to obtain a set of uncorrelated features, which are linear combinations of original features	Method
t-SNE	Probability that pairs of data points in high-dimensional space are related is determined, then a low-dimensional embedding with a similar distribution is chosen	Method
Multidimensional Scaling	Lower-dimensional representation of original data that preserves pairwise distances is found	Method
Accuracy	Metric that measures how often the model correctly predicts an outcome; calculated by dividing the number of correct predictions by the total number of predictions	Numeric
Adjusted Rand Index	Similarity measure between two clusterings by considering all pairs of samples and counting pairs assigned to the same or different clusters in predicted and true clusterings	Numeric

Table 5.2.: Pre-processing components II

5.3. Tasks

This section contains a list of tasks the domain experts want to tackle using the developed tool. The tasks were collected in meetings with each of the two domain experts, together with a supervisor who kept track of the identified requirements. Next, the resulting prototypes from the identified tasks were presented to the domain experts in one-on-one meetings and refined with new requirements in an iterative process. After the meetings, a joint meeting with both domain experts and both supervisors was held to obtain the final list of tasks. Each task is identified by a unique identifier, which will be referred to later. The domain experts want to ...

T1 **Track embeddings** over epochs. Right now, there is no such tool that supports this task. The domain experts expect to learn how latent space embeddings are processed internally while a deep neural network is trained.

T2 **Cross compare** the embedding development with loss development. The domain experts need this functionality to investigate how the reduction in loss value interacts with the embedding and clustering quality. For example, how jumps in the loss correspond to differences in the embedding.

T3 **Switch projections** between different dimensionality reduction methods for the embeddings. Switching between dimensionality reduction techniques to investigate different types of input data, for example, with linear or non-linear relationships. For some data sets, certain dimensionality reduction techniques produce a better visual separation than others.

T4 **Evaluate algorithms** by calculating the final cluster assignment for various deep clustering algorithms. The domain experts want to investigate which deep clustering algorithm performs best on the training data. Various deep clustering algorithms should be made available to visualize in the dashboard.

T5 **Evaluate hyperparameters** by calculating the final cluster assignment for different hyperparameters of the neural network. The domain experts want to change these parameters of the deep clustering algorithms, especially for unlabeled data, to find the optimal number of clusters if they are not known beforehand in an unsupervised scenario.

T6 Do a **custom loss analysis** by investigating the loss development over time for custom losses. The domain experts want to test new deep clustering approaches by minimizing different (combinations of) losses. They want visual support and a visual link on how the embedding changes with the loss change.

T7 **Evaluate clusters** with an overview of several clustering metrics for different cluster assignments. The domain experts want to understand how well a deep neural network performs for clustering tasks. Therefore, each clustering should be evaluated during the training to support the domain experts' decision-making in choosing the best model for their tasks.

T8 **Compare models** with different embeddings for different training runs with different custom losses. To evaluate different trained models on their ability to achieve a reasonable clustering, domain experts need to upload more than one training result into the tool and have a side-by-side comparison. Poorly performing models should be removed from the tool.

T9 **Trace misclusterings** and map misclustered data points in the latent space to the original data. The domain experts can investigate why a particular data point is misclustered.

T10 Do a **benchmark comparison** of for different benchmark data sets. Deep clustering algorithms are usually evaluated on various benchmark data sets like MNIST [LBBH98], STL-10 [CNL11], and REUTERS [LYRL04]. The tool should be suited for these benchmark data sets.

5.4. Design Goals

This section presents the overarching design goals that guide the development of the tool. The goals were derived from insights shared by the domain experts during individual meetings with each expert, accompanied by a supervisor who helped to document relevant requirements and expectations. Following these meetings, early design concepts were developed and iteratively refined in one-on-one sessions with each expert to better align with their workflows and preferences. Finally, a joint session with both domain experts and both supervisors was conducted to consolidate and validate the design goals. The system should ...

DG1 **Avoid heavy computations** or data transformations. Computations for large data sets would take a long time. Dynamic features like the embedding visualization over epochs for different dimensionality reductions would take too long and make the tool impractical.

DG2 **Seamless integrate** with the existing workflow. The domain experts do not want to change their existing workflow. The dashboard should serve as an extra step in the analysis phase.

DG3 Have **high performance** and be capable of supporting a large number of models without experiencing lags. After a logfile is uploaded, moving through epochs, jumping to an epoch, switching projection methods, or getting information about certain data points should happen without lag.

DG4 Provide **clear linkage** between model information, embedding evolution, confusion matrices, and loss progression. Domain experts want to have all the information quickly available and not look up which information belongs to which model.

DG5 Support **outlier detection** and identification of outliers and potential clustering failures, while also allowing users to access detailed, data point–specific information on demand.

5.5. Prototypes

This section presents the prototypes to demonstrate the tool's development process. The development process started with low-fidelity prototypes drawn on paper or digitally (see Design in Figure 2.1). It is a convenient way to get rapid prototyping (see Figure 2.3), which could then be presented to users to get their feedback. This feedback was then incorporated into a further round of prototyping, until it was reasonable to create a high-fidelity prototype (see Implement in Figure 2.1). An overview of the low-fidelity prototypes can be found in Table 5.3.

Figure	Summary
5.1	Compare multiple embeddings, losses and scores side-by-side
5.3	Compare multiple embeddings of different runs side-by-side with scores
5.4	Comparing multiple clustering methods with scores side-by-side
5.5	Compare clustering runs in a matrix
5.6	Get textual information about scores
5.7	Compare two embeddings side-by-side as well as clustering runs in a matrix
5.8	Have different tabs for different stages of the workflow
5.11	Compare the embedding before training with embedding over training
5.12	Compare multiple embeddings over the complete training process

Table 5.3.: Prototype Overview

5.5.1. Low-fidelity Prototypes

In the following section, the low-fidelity prototypes are shown in the order of creation. Low-fidelity prototypes in the same subsection are created in parallel to present the domain experts with a variety of designs. Each subsection represents one round of presentation of low-fidelity prototypes to the domain experts, and received feedback.

5.5.1.1. First Round of Low-fidelity Prototypes

This section contains the first low-fidelity prototypes created for the domain experts' tasks.

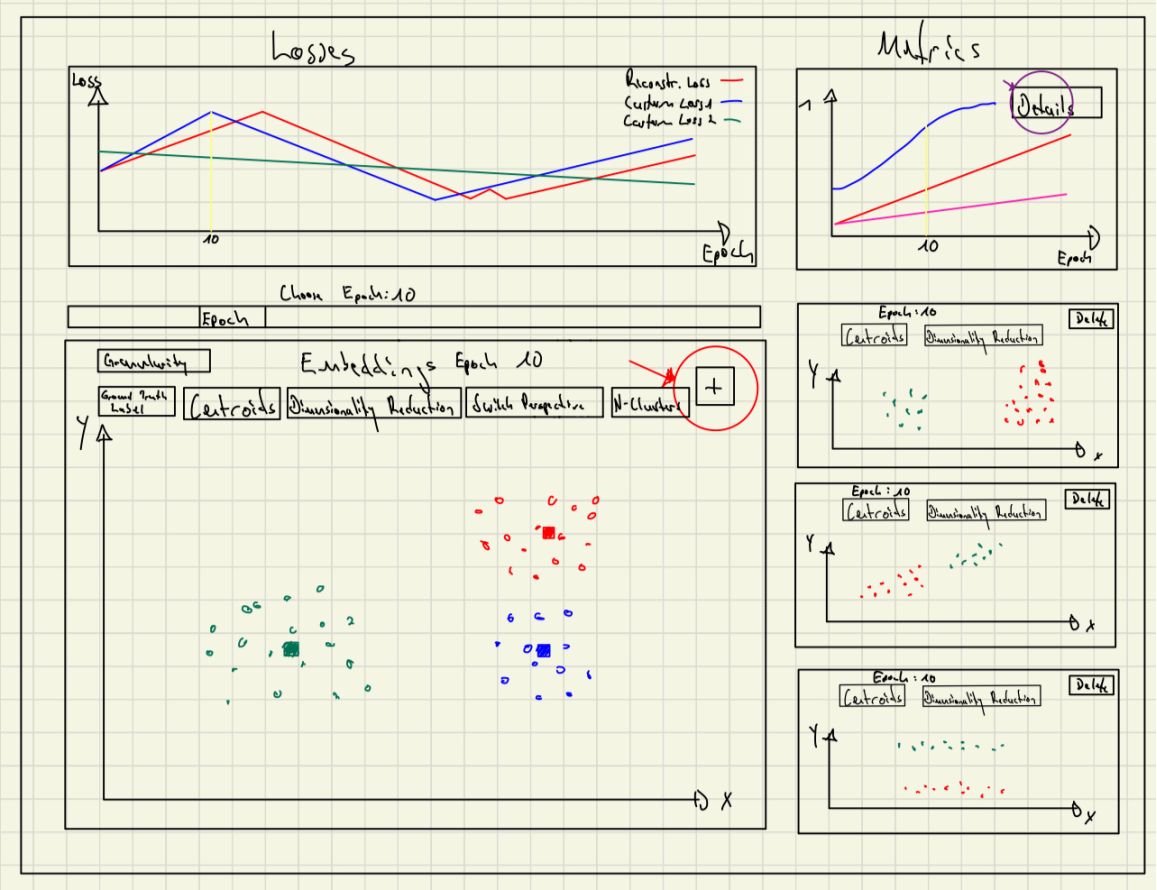

Figure 5.1.: Low-fidelity prototype with embeddings and clustering view

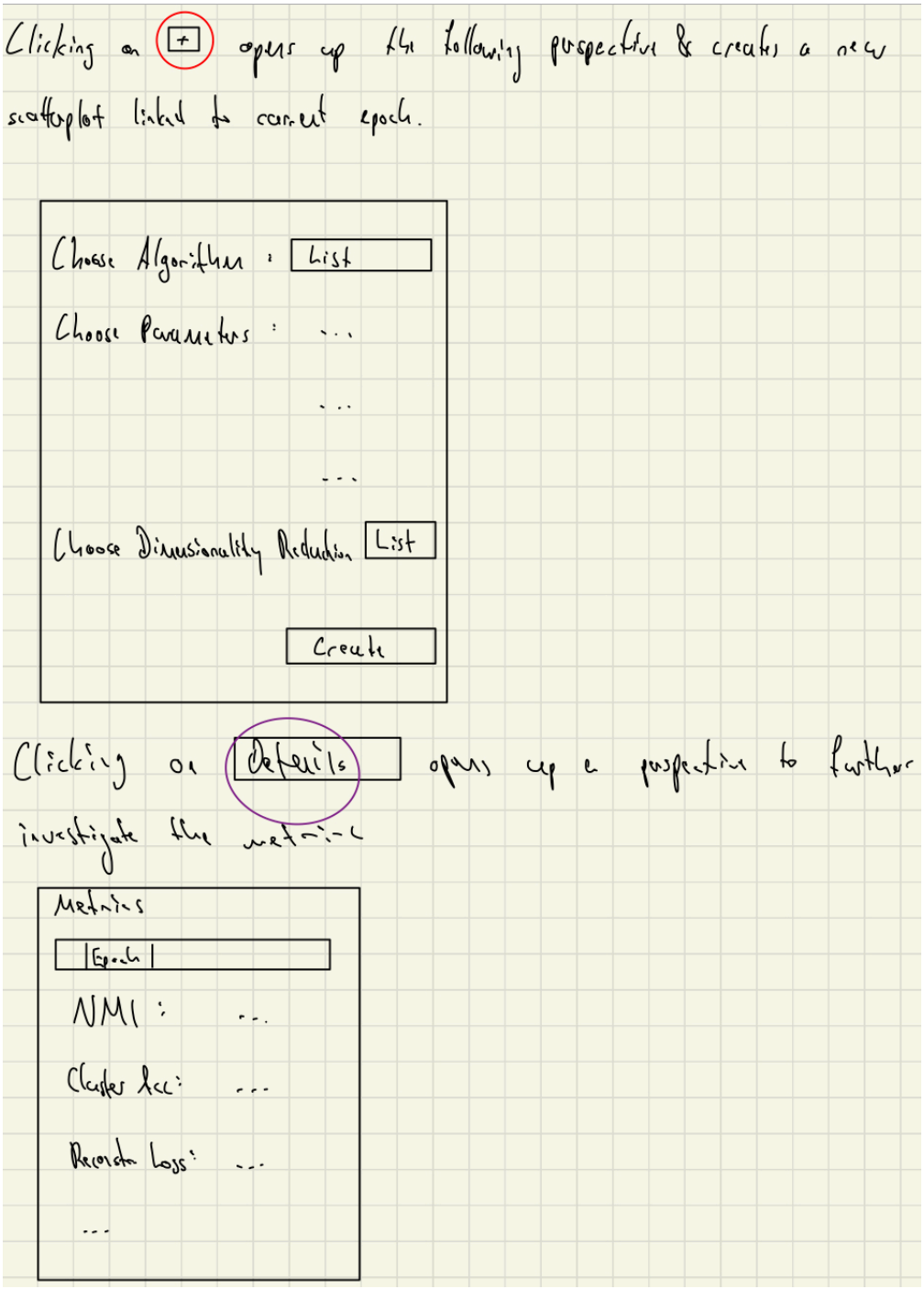

Figure 5.2.: In-depth view of metrics and clustering creation of forward pass of 5.1

The first prototype (see Figures 5.1 and 5.2) contains two line charts for the loss and clustering metrics comparison to support the domain experts in evaluating the quality of

the trained neural network. This enables the investigation of custom loss developments over time (T6) and provides an overview of relevant metrics (T7). The clustering metrics can be evaluated through a details button, which opens up a field with tabular data that can be investigated for each epoch. A scatter plot shows the embeddings throughout the training process. These embeddings could be dynamically investigated with a slider above the scatter plot. This supports the detailed exploration of clustering results on the embeddings at different stages of training (T1). A juxtaposition with the line chart for loss development enables the users to compare the embedding development together with the loss development (T2). Several buttons provide a variety of functionalities to the domain experts. They can change the granularity of the clusters to superclusters, change between computed and ground truth labels, toggle the clustering centroids, switch between the dimensionality reduction techniques, and change the number of clusters that a clustering algorithm should find. With a plus button, a bar opens where the user can configure a new scatter plot with a new clustering algorithm, with a parameter configuration, and dimensionality reduction. These parameters are specific to the chosen clustering algorithm. K-Means takes the k value and the maximum iterations, whereas DBSCAN takes an epsilon value and the minimum number of data points. These functionalities support the flexible analysis of clusterings using different dimensionality reduction methods and clustering algorithms with different hyperparameters (T3, T4, T5). Furthermore, the dashboard allows side-by-side comparison of different models by showing their embeddings in juxtaposed scatter plots (T8).

Feedback: The feedback of the initial prototype revealed several foundational usability issues, particularly concerning how information was presented and accessed. One primary concern was the use of pop-up elements for displaying detailed metric information. Users criticized this approach for its poor usability, noting that important values should be immediately visible without requiring additional interaction. Furthermore, it was not straightforward to determine which clustering showed good performance since no metrics corresponded to the clusterings, leading to confusion and making it difficult to conduct a reliable performance comparison. In addition to these usability concerns, feedback also addressed the design of the clustering visualization components. Specifically, it was suggested that clustering results should not be computed for every epoch during training, as this introduces unnecessary complexity and computational overhead. Instead, clustering should be limited to the final epoch, when the model has converged, to provide a more stable and interpretable result. Embedding, cluster, loss, and metric visualization suited them, since they are familiar with the visualizations for this kind of data.

5.5.1.2. First Iteration of Low-fidelity Prototypes

The feedback for the first prototype was integrated, leading to the first iteration of low-fidelity prototypes, presented in this section.

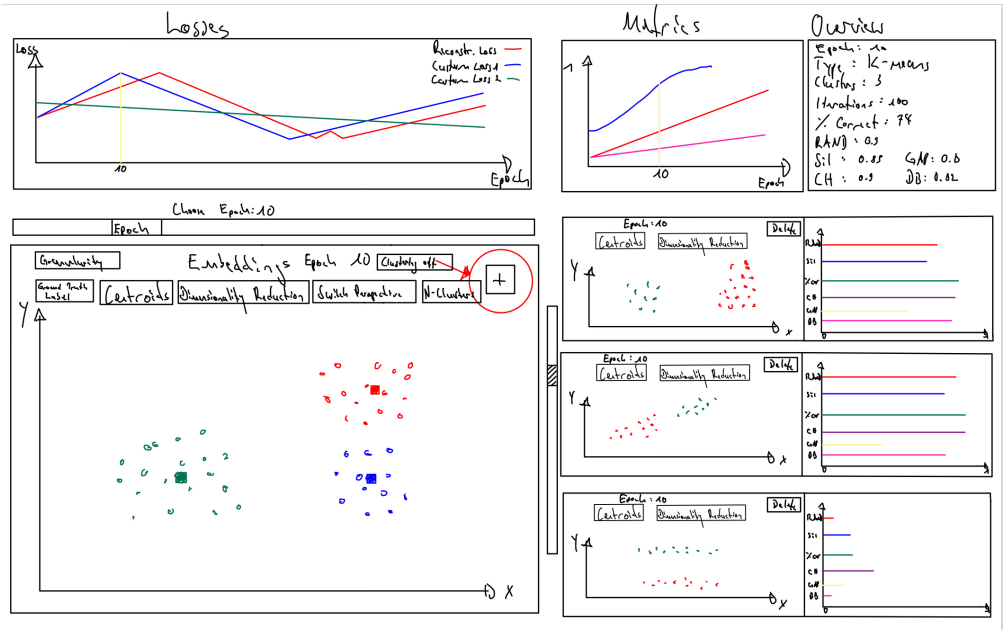

Figure 5.3.: Low-fidelity prototype with juxtaposed clustering and metrics

After the first round of reviews, the first low-fidelity prototype was iteratively refined and varied, leading to two new prototypes (see Figures 5.3 and 5.4). The prototype (see Figure 5.3) allows the domain expert to upload multiple training runs of a deep neural network trained with different losses. One main embedding will be shown as slightly bigger in the bottom left, and the embeddings that the domain experts want to compare will be shown to the right, next to the main embedding. This supports the side-by-side comparison of different models through juxtaposed scatter plots (T8). The main embedding scatter plot has the same functionalities as the prototype before (see Figure 5.1), allowing experts to track embedding changes over training via an epoch slider (T1). The small scatter plots have a bar chart next to them, so the domain experts get a quick overview of the embedding quality, providing a quick overview of embedding quality for cluster assignments (T7) and a side-by-side comparison of different models (T8). The domain experts can individually switch between the dimensionality reduction methods (T3) and turn cluster centroids on and off. All these scatter plots are linked via an epoch slider, which can be found above the main scatter plot. On the upper third of the dashboard, there are several vital insights on the embeddings required by the domain experts. Losses and metrics are shown for each epoch, and a general tabular overview is presented. The juxtaposition of the line chart with the scatter plot aids the comparison of loss and embedding development (T2), where the users can also investigate custom loss functions (T6). The "+" can be utilized to create a new scatter plot and metrics combination of a new clustering algorithm with different hyperparameter configurations (T4, T5).

Feedback: In this prototype iteration, feedback focused primarily on issues related

to layout and information density. Users noted that the textual overview of model metrics occupied a disproportionately large portion of the dashboard interface. While the inclusion of metric information was generally appreciated, its current presentation was seen as inefficient in terms of screen space usage. This reduced the space available for more interactive or visual components and impaired the overall balance of the layout. The feedback suggests a need to condense the textual content or integrate it into more space-efficient visual summaries to maintain a more straightforward and more functional interface.

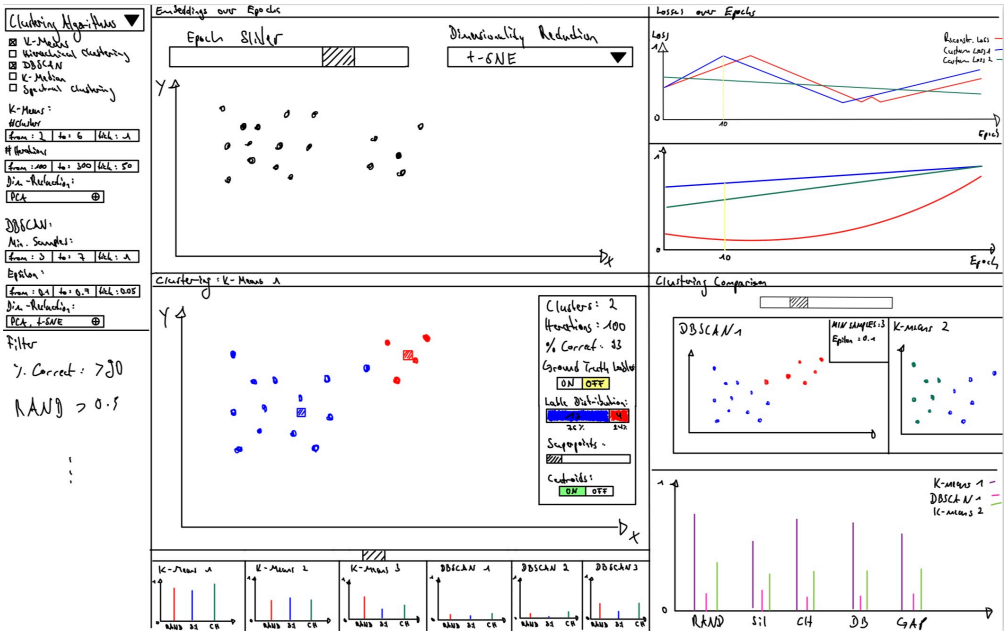

Figure 5.4.: Low-fidelity prototype with embedding and clustering split

A second prototype (see Figure 5.4) focuses not only on the embeddings (see Figure 5.3) but also on clusterings applied to data forwarded through the deep neural network. The upper half focuses on one embedding, where a scatter plot shows the embedding for a selected epoch of a chosen dimensionality reduction method. Users can navigate through epochs to observe how the embedding space evolves during training, thereby visualizing internal changes over time (T1). The scatter plot also features multiple dimensionality reduction methods, from which users can select one. They can switch between these methods to better understand the structure of the embedding and select the dimensionality reduction method that suits the data set best (T3). Right next to this scatter plot is a line chart visualizing several losses and metrics for the whole training process to evaluate the quality of the obtained network and enable an easy comparison of loss and embedding (T2, T6). The complete left side shows a sidebar where the user can configure a variety of clustering algorithms that should be applied to forwarded data (T4). The user can choose a range of parameters and run this configuration (T5). If a configuration is run,

multiple scatter plots, one for each configuration, are shown on the lower half of the dashboard. The domain experts can choose a clustering they want to focus on with a click, which is displayed in greater detail with several further configuration methods that can be applied. The focused clustering provides the user with a range of information, such as the number of clusters, iteration, percentage of correctly clustered data points, and label distribution. In addition to that, the user has the options to turn on centroids of the clustering algorithm used, create superpoints, and turn on the ground truth labels (T9). On the right of the scatter plot with the clustering in focus, there are the rest of the clusterings. Underneath the scatter plots are bar charts denoting the quality of the clusterings with a selection of reasonable clustering metrics (T7). The prototype is designed to work with commonly used benchmark datasets (T10).

Feedback: The feedback on this dashboard prototype revealed several important considerations related to comparison functionality and visual clarity. A key shortcoming identified was the lack of direct comparison between different clustering results. While one clustering was prominently displayed, its visual dominance overshadowed the others, leaving insufficient space for the other clusterings, making comparative analysis difficult. This issue was further exacerbated by the small size of certain elements, like the bar charts used to represent metric values. Furthermore, while clustered bar charts were deemed valid when comparing a small number of models, concerns were raised about their scalability. As the number of models increases, this visualization technique may become cluttered and difficult to interpret. In addition, users found the placement of the bar charts misleading, as they appeared under the "false" or less intuitive scatter plots, which introduced confusion about which visualizations are linked. In terms of filtering and overall usability, the prototype's current mechanisms were seen as insufficient. Text-based filtering was described as ineffective, and users expressed the need for more robust methods to navigate and exclude irrelevant clusterings. A notable limitation was the inability to filter out clusterings that were not of interest, since only ranges could be filtered out, which constrained the user's ability to focus on specific analyses. Feedback also addressed component design and the structural organization of dashboard features. While the configuration options for clustering algorithms were generally praised, users also noted a more apparent structural distinction between the forward pass clustering and the embedding exploration sections. In particular, the juxtaposition of metric and loss values was found to be more effective when presented side by side, enabling a more coherent interpretation of model performance. At the same time, the overall separation between the embedding analysis and the clustering analysis was viewed positively, contributing to a more modular and focused user experience. However, a missing legend in the metric line chart reduced the clarity of the presented information and was identified as a key area for improvement.

5.5.1.3. Second Iteration of Low-fidelity Prototypes

Given the feedback from the domain experts, they preferred the second prototype of the previous iteration. The focus shifted to iteratively refining this prototype, leading to two new, improved varieties presented in this section.

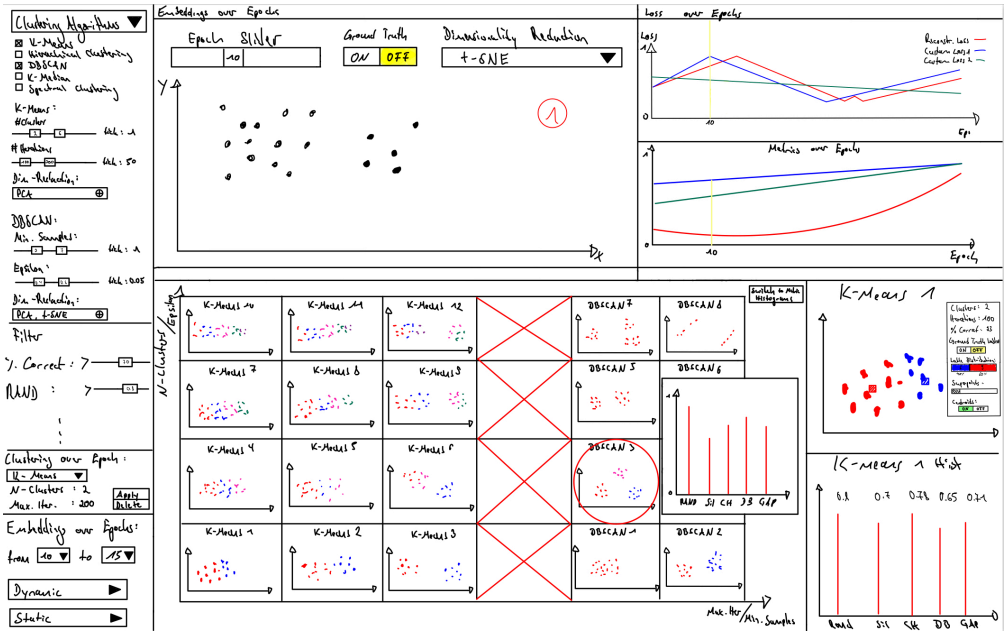

Figure 5.5.: Low-fidelity prototype with clustering scatter plot matrix

Focusing on the second dashboard of the second iteration (see Figure 5.4) was the preferred option after talks with the domain experts and supervisors. The juxtaposition of embeddings, metrics, and losses in the upper half suited them well, as it grouped the model training-related components. Some reasonable changes that incorporate the given feedback were made to increase usability. The number of clusterings was brought together in a scatter plot matrix, as seen on the bottom left. Now, an easy comparison over multiple clustering algorithms, with different hyperparameter settings, is possible (T4, T5, T8). When hovering over one of these clusterings, a bar chart is shown to give the domain experts an overview of several clustering metrics to evaluate the quality of the clustering (T7). On clicking on one of these scatter plots, they are shown on the right of the scatter plot matrix for a more detailed view of the chosen clustering. Other options on the scatter plots and line charts are kept the same as those on the previous dashboard prototype (see Figure 5.4) (T1, T2, T3, T6, T9, T10). Within the sidebar component, the domain experts can now filter out clusterings from the scatter plot matrix based on user-configurable parameters. In addition to that, two buttons can be used to get a dynamic or static visualization of the embeddings during the training, replacing the previously utilized epoch slider component.

Feedback: In the feedback, several aspects of the dashboard's interaction design and usability were critically assessed. A recurring point of confusion concerned the placement and functionality of the epoch slider. Users noted that its current implementation appeared disconnected from the rest of the interface, as it did not correspond to any elements, like the line charts' x-axis, leading to ambiguity about its purpose. Within the embedding exploration section, interaction mechanisms for switching between dimen-

sionality reduction methods and ground truth labels were highlighted. It was suggested that toggling between these views could be more intuitively handled via simple buttons, allowing for quicker transitions and a more straightforward interface. Concerning visualization and comparison limitations, several usability gaps were noted. While the scatter plot matrix approach received positive feedback, users pointed out that the feature to enlarge plots was only marginally effective, since the size increase was considered too small to provide a substantial benefit. Furthermore, although the scatter plot matrix enabled detailed clustering analyses, the dashboard lacked functionality for comparing two clustering results side by side, which limited comparative investigation. Additionally, while the newly introduced filtering mechanism using a slider was regarded as a step forward, users proposed the implementation of a range slider, similar to the one used in the improved clustering configuration, to offer more granular control when filtering data by score or metric ranges. Another significant limitation was the inability to upload and compare multiple models simultaneously. The dashboard supports only a single model upload, which restricts comparative model evaluation and cross-model analysis during model training, an essential feature in model selection and performance interpretation. Despite these critiques, the prototype received positive remarks for several improvements. In particular, the clustering configuration interface was praised for its clarity and usability. This reflects progress in addressing previous concerns related to clustering algorithm customization and parameter selection. Additionally, the overall adoption of scatter plot matrices for visualizing clustering was seen as a valuable addition, even though its potential for comparative analysis could be further enhanced through design refinements.

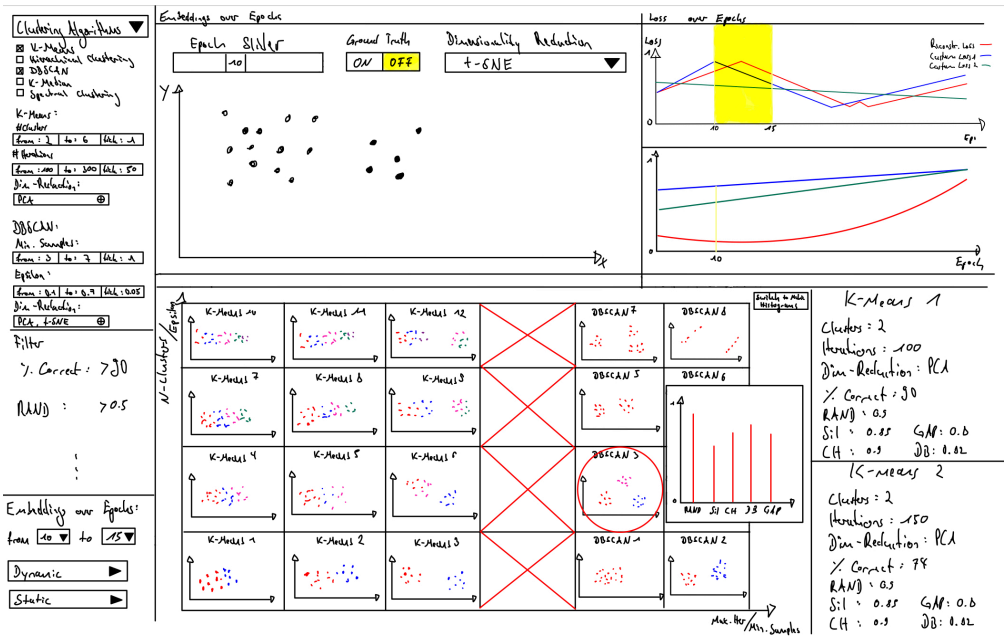

Figure 5.6.: Low-fidelity prototype with clustering scatter plot matrix and textual clustering description

The next low-fidelity prototype (see Figure 5.6), which was developed in parallel, replaces the focus scatter plot and bar chart with tabular metrics to compare two different clusterings. It provides the domain experts with the same information as the previous prototype (see Figure 5.5), but in a textual instead of a visual channel, thereby also covering the same tasks. To further support the training embedding investigation, the user can either click on the line chart in the top right to statically show the embedding or brush an area and dynamically show it from start to end. After presenting the two prototypes, the domain experts agreed that the first one suited their work better since they wanted to investigate the clusterings visually.

Feedback: In this variation of the prototype, feedback focused primarily on the effectiveness of the information presentation. Specifically, the textual comparison feature introduced in place of the bar chart and scatter plot was deemed unnecessary. Although it would allow a comparison of two clusterings, users noted that the added text did not provide substantial value beyond what could be conveyed through simpler print statements. As such, the textual representation was perceived as redundant and less effective than visual alternatives for enabling quick interpretation and comparison. This suggests that visual elements remain the preferred option in focus, discarding the need for textual analysis from that iteration onward.

5.5.1.4. Third Iteration of Low-fidelity Prototypes

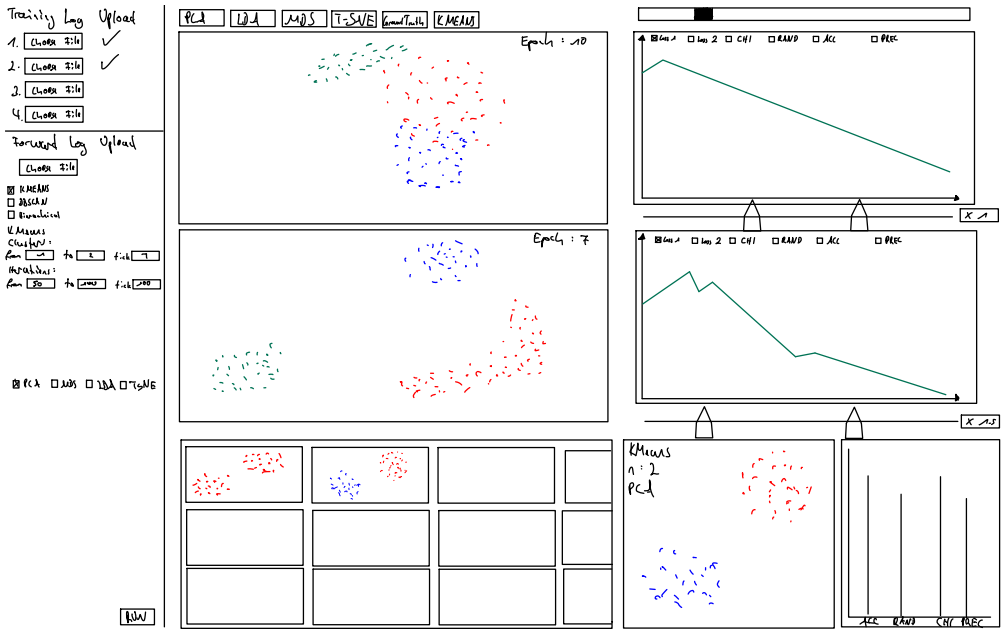

Figure 5.7.: Low-fidelity prototype allowing multiple embedding log file uploads

The need for the domain experts to compare multiple networks simultaneously leads to another round of rapid prototyping, resulting in the next prototype (see Figure 5.7).

Here, the sidebar is adjusted so that log files can be uploaded directly and not in the dashboard code. A new pair of scatter plots and line charts is created for each uploaded training log file, which can be controlled independently, making it easier for the user to analyze and compare models in detail (T1, T2, T6, T7, T8, T10). The scatter plot shows the embedding for each epoch, whereas the line chart shows the particular run's loss and clustering metrics. The bottom part focuses on a forward pass through the network and remains the same as in the previously preferred prototype (see Figure 5.5) (T4, T5, T8). The sidebar remains the same and can be used to set hyperparameter ranges and clustering algorithms. The dimensionality reduction method, ground truth labels, and k-Means labels can be switched with buttons above the uppermost scatter plot, making it easier for the user to switch between desired projection methods and colorings over all the models (T3, T9).

Feedback: The prototype was positively received overall, with users acknowledging that it worked well and provided a strong foundation for further development. Despite this positive assessment, users identified several areas where improvements could enhance the dashboard's visualization and data insight capabilities. One critical limitation was the absence of any means to evaluate the clustering algorithm's performance on the original, unprocessed data. This gap limits the user's ability to contextualize the effect of clustering and dimensionality reduction techniques. Additionally, while the slider-based interaction enabled dynamic exploration of embedding changes over time, users expressed a desire for static representations as well. The focus on dynamic exploration did not allow for the comparison of the embedding at two different points in time during the training. Static snapshots would allow for more deliberate comparisons between key epochs or models, complementing the dynamic functionality. From a feature and functionality perspective, several enhancements were recommended. As the prototype introduced the ability to upload multiple models, users noted the growing need for a filter mechanism that would support focusing on the models and not the clusterings. Additionally, users suggested removing the clustering applied to forwarded data in favor of emphasizing the training process, its outcomes, and the overall model quality, which they considered more informative for model evaluation. The ability to upload more models was also requested, alongside the visualization of key hyperparameters used during training. Providing this metadata would improve transparency and allow for deeper analysis of the relationship between model configuration and performance.

The following developed low-fidelity prototype separates the different areas the domain experts want to investigate into different tabs, to present the domain experts with a solution that imitates their workflow. First, they investigate the model during the training and afterward evaluate the clustering on unseen data. Overall, there are three tabs: one for a training log comparison of different models (see Figure 5.8), one for a deeper investigation of a single training run of a model (see Figure 5.9), and one for a forward pass investigation for one model (see Figure 5.10). In this prototype, visual separation and quality measures are used for the first time.

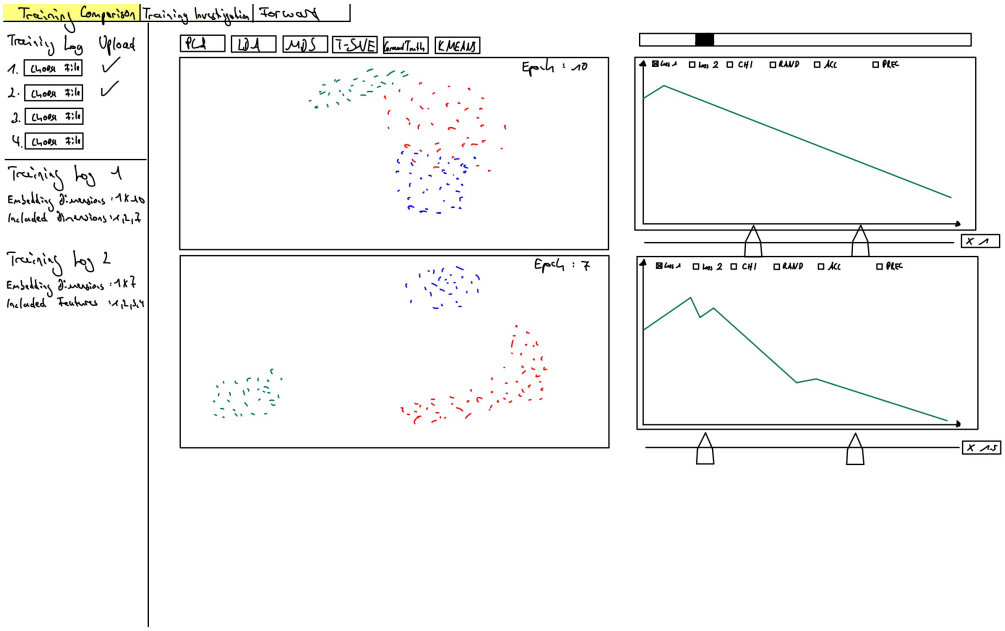

Figure 5.8.: Low-fidelity prototype with different tabs, here for embedding comparison of different models

In the training comparison tab (see Figure 5.8), the user retains the same ability to upload up to five different training logs for models (see Figure 5.7). For each of these log files, a new pair of embedding scatter plots, as well as loss and metrics line charts, is displayed. Additionally, the user can choose which features to include before the dimensionality reduction occurs (T1, T2, T6, T7, T8, T10).

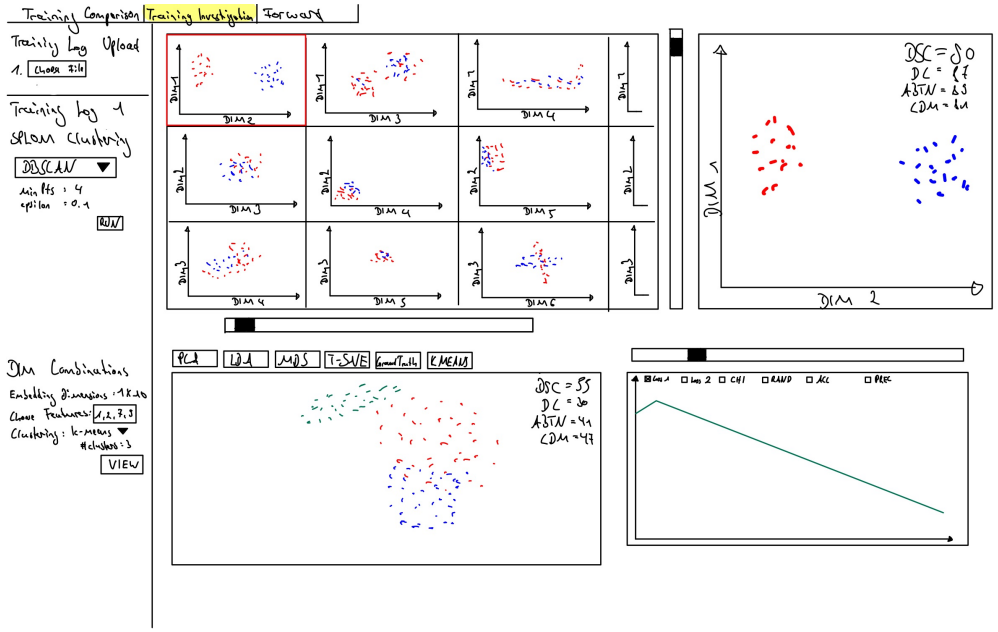

Figure 5.9.: Tab for clustering with visual separation measures

The user heads over to the training investigation tab to investigate and identify the dimensions that carry information (see Figure 5.9). There, he can upload a log file of a single model in a sidebar. A scatter plot matrix is displayed, plotting all the found features in the embedding against each other so the user can identify the dimension that best separates the data points. The user can choose from different clustering algorithms and fine-tune their parameters for each plot within the scatter plot matrix (T5). With a click on a single scatter plot in the matrix, the scatter plot will be displayed bigger right next to the scatter plot matrix, together with various visual separation measures for each category. Additionally, the user can reduce the embedding to certain features of interest and then apply a chosen clustering algorithm (T4). A scatter plot is displayed on the bottom left, where the user can switch between different dimensionality reduction techniques (T3) and switch on ground truth labels (T9) or k-Means labels. Further, visual separation scores display how the displayed scatter plot performs. Right next to the scatter plot is a line chart containing losses and metrics over the training epoch. Above this line chart is a slider to go through the different training epochs (T1).

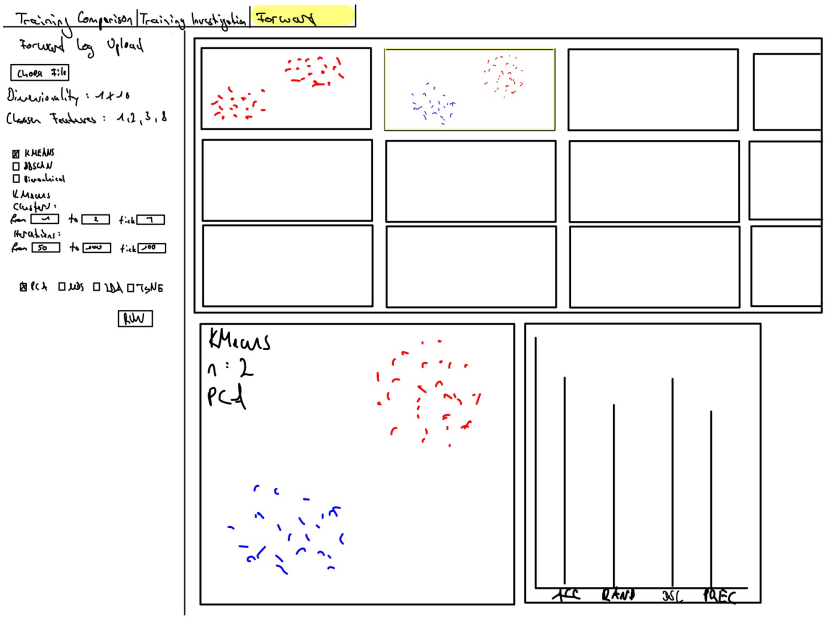

Figure 5.10.: Tab for embedding clustering optimization

If the user wants to investigate a forward pass through the network, he has to go to the third tab (see Figure 5.10). It has the same functions as in the prototype before (see Figure 5.7) and adds a function for the user to remove certain features before clustering and reduction.

Feedback: In the feedback on this tab-based prototype, several limitations were identified that affected its effectiveness for detailed model analysis. A significant concern was the inability to assess the performance of the clustering algorithm on the original, unprocessed input data. Without this baseline, users found it difficult to judge the effectiveness of clustering results with the original feature space. Additionally, tracking changes in the embedding space across the whole training process was cumbersome because only a dynamic view was available. Further, the tab structure required users to manually switch between different parts of their workflow, which disrupted the continuity of analysis and made it easier to get lost in the process. While this prototype introduced a structured and organized interface, it was not considered the preferred option among users. The main reason was that not all relevant information was readily visible or accessible at once, requiring constant navigation between tabs. However, despite these shortcomings, the prototype was acknowledged as a useful experimental version. Users recommended incorporating the insights gained from this iteration into another iteration of a low-fidelity prototype, since elements would stand out more effectively for testing.

5.5.1.5. Fourth Iteration of Low-fidelity Prototypes

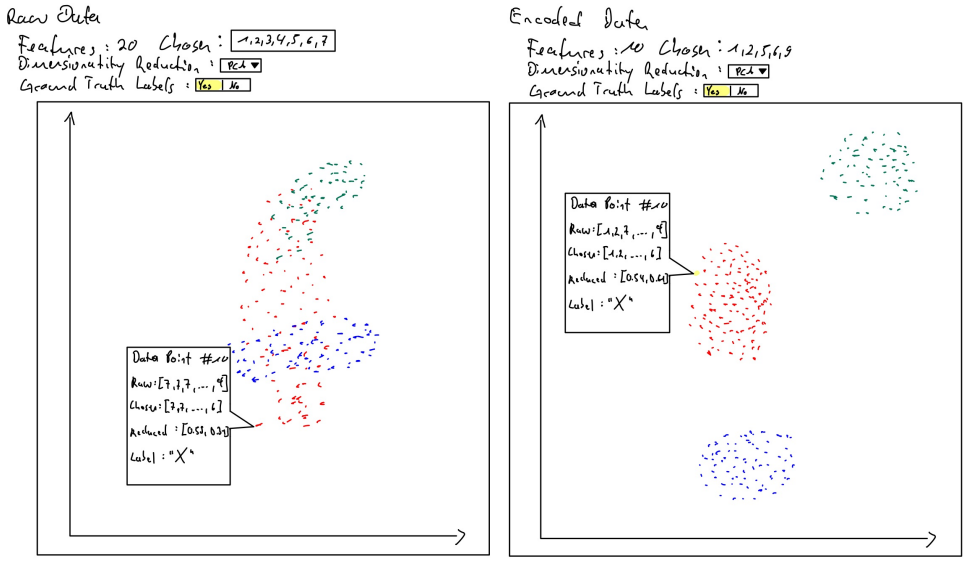

Figure 5.11.: Tab for latent space and normal space comparison

In the next iteration, another tab was added to compare the latent and the normal space (see Figure 5.11). The user can upload two files, where one contains the normal data and the other the embedding of the last epoch. Both get plotted as a scatter plot and juxtaposed. There are three different options for the user. He can choose certain features he wants to include in the scatter plot and exclude others. He can choose between several different dimensionality reduction methods to be applied to the data to bring it down to two dimensions. Additionally, he can decide whether he wants to get the ground truth labels shown or not. Furthermore, the data points are linked in both scatter plots and have tooltips to give the user-specific information on the data point he is investigating.

Feedback: In the iteration of this dashboard prototype, several points of constructive feedback were raised, particularly concerning layout and information accessibility. A key issue was the use of tabs in the interface to separate different parts of the dashboard. Participants expressed a preference for having all relevant information available at a glance, rather than being split across multiple tabs. This fragmentation was seen as a barrier to maintaining a holistic overview, especially when evaluating and comparing different models. Additionally, it was noted that the interface did not indicate which model or clustering result was currently being viewed. This ambiguity could lead to confusion, particularly for domain experts engaging in detailed model interpretation, with multiple models uploaded. Another important theme that emerged from the feedback was the comparability of models and embeddings. Users highlighted that the dashboard lacked a mechanism to compare embeddings with different dimensionality reduction methods

applied from different epochs across multiple models side by side, since only a global change of dimensionality reduction is possible. This limitation made it challenging to observe temporal or model-specific differences in learned representations. Furthermore, there was no filter or selection mechanism to manage or narrow down the list of available models, which becomes increasingly problematic as more models are uploaded and visualized. To support comparative analysis more effectively, it was also suggested to provide a histogram overview of training metrics across all models. In terms of content scope and missing information, several omissions were noted. The forward pass functionality was considered unnecessary for the prototype's current purpose and could be omitted in favor of a more focused emphasis on the training phase. Moreover, the prototype lacked transparency regarding the hyperparameters used in training each model. Including this information would enhance reproducibility and help users better understand model behavior. Finally, the prototype did not support uploading or displaying more than five models, limiting its usefulness in scenarios where grid searches for model evaluation are critical.

5.5.1.6. Fifth Iteration of Low-fidelity Prototypes

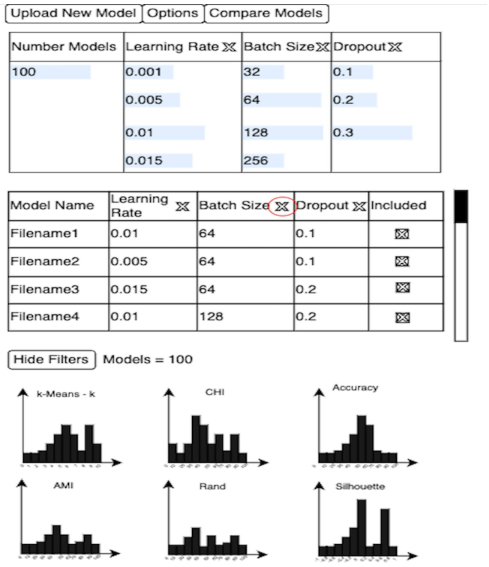

Figure 5.12.: After uploading models

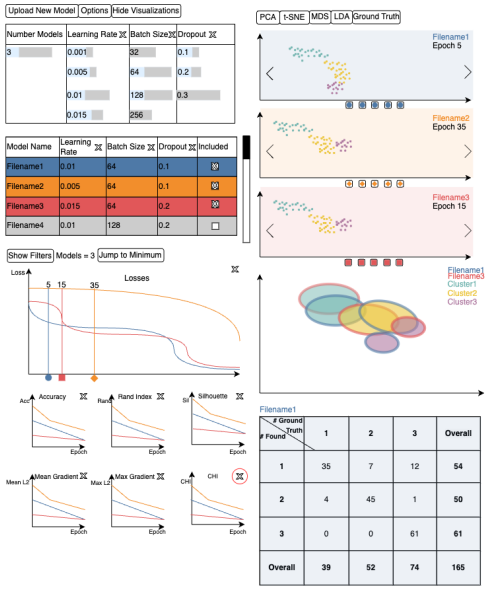

Figure 5.13.: After filtering

The next prototype incorporates the feedback and is scaled up for uploading more models. For each clustering algorithm (T4) with a different hyperparameter setting (T5), the user creates a new log file, which is uploaded to the model. In the overview, the entire range of models is displayed with information about their parameters and scores (see Figure 5.12). The user gets insights into particular models as well as an overview of all the models. The "Table Chart" (top) summarizes all the model's hyperparameters in horizontal bars. Each bar represents the number of models with a particular hyperparameter, giving the user an overview of the uploaded models. The users also get an individual view about the models in the "Model Information Table" (middle), where they can delete hyperparameter columns that they do not need. One column lets the user set a checkbox to either include or exclude models from the table. If models get excluded, then their part in the bars of the "Table Chart" and "Bar Chart Matrix" (bottom) gets grayed out to signal to the user in which bins the model appears. This step is meant to be a high-level view that allows users to get into the data and give context to the next steps in the visualization and initially provide them with metric scores of the models (T7).

In the first view (see Figure 5.12), the user can filter out models of no interest, which can be done via the "Included" column. Models that have a poor performance, for example, in terms of accuracy, can be filtered out and will not be considered further in the process. The first zoom can be done via a "Compare Models" button, which leads to the models being displayed throughout the training (see Figure 5.13). The "Bar Chart Matrix" becomes hidden, and "Line Charts" (bottom left) are shown. They provide

a detailed analysis of the different metrics throughout the training process of selected models, enabling loss and metric development over the training process (T6, T7). The big line chart can be used as a slider for the epochs in the "Scatter plots" (top right) to dynamically investigate the embedding during the model training (T1) and compare it with the loss development (T2). For each included model in the "Model Information Table", a scatter plot is created, allowing the user to compare different embeddings for different models with different hyperparameters (T8). Within the scatter plots, there are arrows on each side to jump either one epoch back or forward. Boxes underneath the scatter plots can be used to jump 10 epochs backward or forward. Above the uppermost scatter plot, there are buttons to choose which dimensionality reduction method should be used to project the data point at a given epoch (T3). Furthermore, there is a button to switch between ground truth and found labels, enabling the user to perform outlier detection (T9). The user can zoom in on data points of interest. Under the scatter plots is a procrustes analysis plot (middle right) where clusters get mapped into the same space to compare model clusters directly with each other. One "Confusion Matrix" (bottom right) can be shown by clicking on a scatter plot at the time and depicts data points found by the deep clustering algorithm compared to the ground truth labels (T9). The lines, the background of the scatter plots and confusion matrix, and the selected models in the model information table are colored the same if they belong to the same model to draw a visual connection for the user.

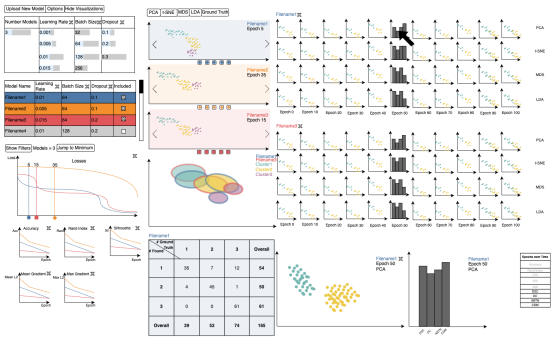

Figure 5.14.: After selecting scatter plot

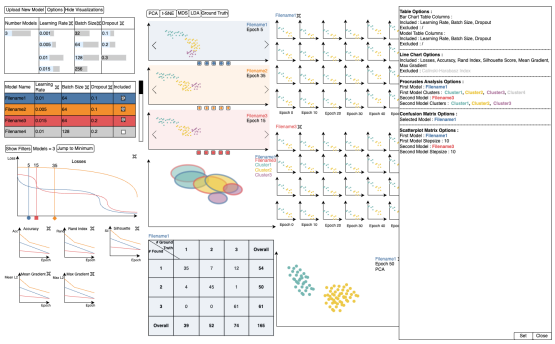

Figure 5.15.: After opening options menu

The user can get further details on demand (see Figure 5.14) by double-clicking on a scatter plot and expanding a "Scatter Plot Matrix" (top right). Every tenth epoch is shown in parallel over the whole training process, giving the user the ability to statically track the embeddings throughout the model training (T1) and compare them to other models (T8). The user can hover over one of the small multiples and switch the view of all small multiples of the epoch to visual separation scores. He can also click on one of the small multiples to get the visual separation score as a bar chart and a detailed view of the epoch (bottom right). By hovering over data points, clusters, or lines, a tooltip displays the model, epoch, and exact value, allowing the user to further explore the models without

cluttering the screen. By clicking the "Options" button (see Figure 5.15), the user can expand a "Sidebar" (right). There, he can include and exclude columns, line charts, and scatter plot matrices. Additionally, he can set further options as to which model he wants to investigate via the confusion matrix, the stepsize of the scatter plot in the scatter plot matrices, and the clusters to examine in the procrustes analysis.

Feedback: During the presentation of the dashboard prototype to the domain experts, several usability and design issues were identified. One of the primary concerns is related to the layout and organization of visual components. Users reported confusion arising from the spatial arrangement of elements, specifically, scatter plots, scatter plot matrices, and confusion matrices belonging to the same model were not positioned nearby. This made it more challenging to perform coherent, model-specific analyses. It was also noted that configuration options were too hidden, being placed in a collapsible sidebar that reduced their visibility and discoverability. Furthermore, the prototype lacked dedicated confusion matrices and scatter plot matrices for all models, limiting the comparative analysis. Providing these elements for each model would allow users to better understand performance differences between models and how dimensionality reduction affects each one. In terms of interactivity and usability, the current implementation was found to be unintuitive in several respects. Users struggled with retrieving details about individual models via clicking the scatter plot, indicating that the click-based interface lacked clear guidance. The placement and labeling of buttons for selecting label coloring and dimensionality reduction methods also led to confusion, as their proximity and visual similarity made it challenging to distinguish between their functions. Additionally, the currently active dimensionality reduction method was not indicated, making it unclear which transformation was being applied at a given moment. Another significant limitation was the absence of functionality for identifying misclustered data points, which restricted the user's ability to explore clustering errors and improve model interpretations. Concerning visualization and redundancy, some components were considered redundant or unnecessarily complex. For instance, the second scatter plot displayed at the bottom of the interface was deemed redundant, as its epoch selection function was already well-supported by the other scatter plot and the interactive slider. Similarly, the use of hover interactions to display metrics on the scatter plot matrix was considered unnecessary, given that the line chart and line chart matrix already conveyed the most relevant performance indicators. Users also noted that including all dimensionality reduction methods within the small multiples of the scatter plot matrix cluttered the view and reduced interpretability. Limiting this view to the key techniques, PCA, MDS, and t-SNE, would allow for larger and more meaningful scatter plots. In scenarios where no specific coloring was selected, embeddings should still be visually distinguishable, for example, by using different colors for each dimensionality reduction method. Moreover, the lack of legends in the line charts hindered their interpretability, especially when multiple models were compared simultaneously. Finally, several suggestions emerged concerning interpretability and information completeness. One crucial missing element was the ability to compare dimensionality-reduced embeddings with the original, unprocessed data, which would help users better understand the transformation effects of various

algorithms. Additionally, the clustering results should not be limited to ground truth labels alone. Still, the dashboard should also display labels produced by the deep clustering algorithms and k-Means to enable a comprehensive performance assessment. To further support interpretability, the name of the clustering algorithm applied in each case should be clearly shown in the corresponding results table. It was suggested to incorporate this feedback into a high-fidelity prototype.

5.5.2. High-fidelity Prototypes

The following section shows the evolution from the first to the final high-fidelity prototype, which was developed in an iterative process.

5.5.2.1. First High-fidelity Prototype

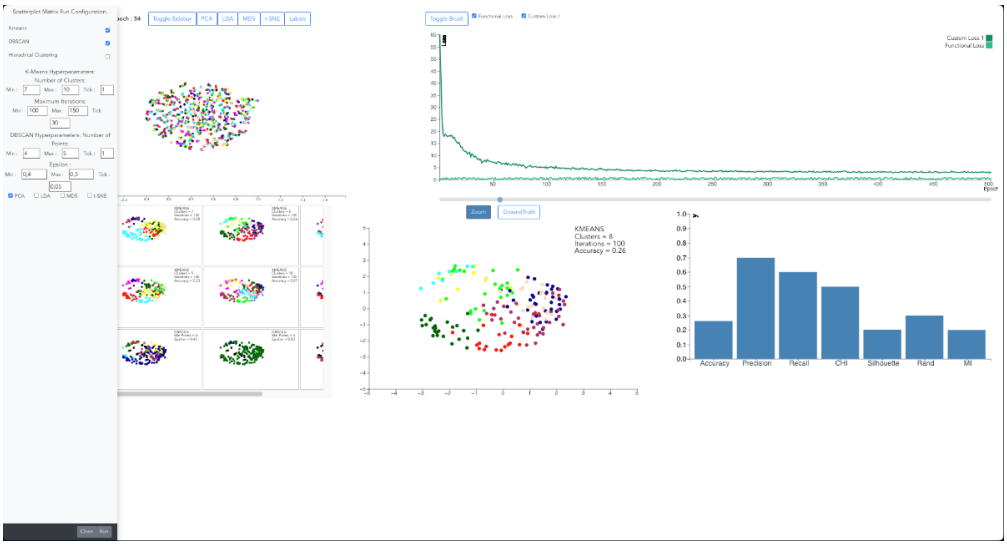

Figure 5.16.: First high-fidelity prototype

The first iteration of the high-fidelity prototype (see Figure 5.16) shows a dashboard created solely with the Vue.js framework and the D3 library, supporting a wide range of input data sets (T10). It is based on the Prototype Figure 5.5. A user must manually load one log file created during the training and one log file created during a forward pass into the tool. It shows a line chart on the top right to investigate the loss development over the training epochs (T6), next to a scatter plot, to compare the loss development with the embedding development (T2). An epoch slider can dynamically investigate the embedding visualization on the top left (T1). With a click on the line chart, the user can jump between training epochs of interest, showing the embedding in the scatter plot. Above the line chart, there are boxes to ignore certain losses that were tracked but are not of interest to a domain expert's current investigation. Left to the line chart, the scatter plot shows

the embedding for a particular epoch for one training of an autoencoder. The user can choose between four dimensionality reduction techniques: Principal Component Analysis, Linear Discriminant Analysis, Multidimensional Scaling, and t-distributed Stochastic Neighbor Embedding (T3). The ground truth and k-Means labels could be investigated at each epoch through the two buttons above the scatter plot (T9). For this function, gold standard labels must be provided to ClustPy during the deep neural network training. The left button on the scatter plot can be used to toggle a sidebar shown on the dashboard's left side. Here, the user can choose between different clustering techniques that should be applied to the embedding obtained through a forward pass (T4). When choosing a clustering technique, the user can configure the parameters of the clustering algorithms they wish to utilize (T5). He can apply a range of parameters and one dimensionality reduction technique to clustering. If the user clicks the run button at the bottom of the sidebar, a clustering is shown for every combination of parameters chosen in a scatter plot matrix on the bottom left. Each clustering shows the algorithm used, the parameters, and the accuracy. A user can further investigate a specific clustering by clicking on it in the matrix. He gets the clustering presented in a single scatter plot where he can zoom in to investigate specific data points and switch between found and ground truth labels (T9). These ground truth labels must be provided to ClustPy during a forward pass through the deep neural network. Further, the user gets various clustering metrics shown in a bar chart on the right of the scatter plot to investigate the quality of the found clustering (T7).

Feedback: The feedback highlighted key usability issues in interaction design, comparative analysis, and model handling. While the scatter plot matrix was well-received, its enlargement feature was seen as ineffective since the lack of a side-by-side clustering comparison limited comparative insights. A filtering, for example, in the sidebar would be beneficial to include precise score or metric filtering. A significant limitation was the inability to upload and compare multiple models simultaneously, restricting cross-model evaluation during training. On the positive side, the clustering configuration interface was praised for clarity and usability, and the use of scatter plot matrices for clustering visualization was considered a valuable addition. However, it requires refinement for enhanced comparative analysis. It was decided to do another round of low-fidelity prototyping before refining this prototype any further.

5.5.2.2. First Iteration of High-fidelity Prototypes

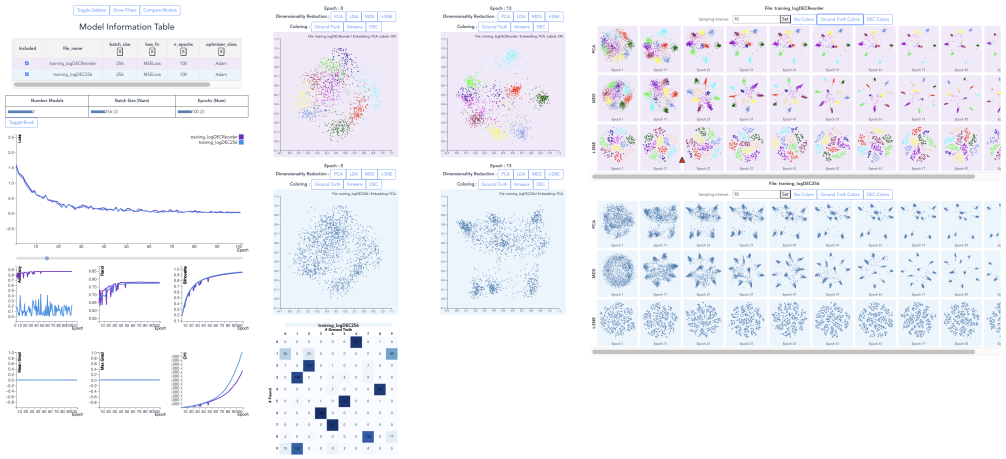

Figure 5.17.: Second high-fidelity prototype

The second iteration of the high-fidelity dashboard prototype (see Figure 5.17) is also based on Vue.js and D3, supporting a wide range of input data sets (T10), and is based on the feedback of prototype Figure 5.14. When the tool is initialized, there is an empty table on the top left, an empty bar chart table below, and three buttons for the sidebar to hide, to show filters, and to compare the models uploaded. Next, the user would have to upload their models through an upload button on the sidebar. The two tables will get filled with precomputed models, covering different clustering algorithms (T4) and a variety of hyperparameter settings (T5). The upper table contains hyperparameters of each model, represented by a row in the table. These hyperparameters comprise model name, loss function, number of epochs, optimizer, learning rate, deep clustering algorithm name, and batch size. Users can delete columns of the table that they do not need for comparing or investigating the models. Furthermore, they can go back to the default table via a button on the sidebar. The lower table summarizes the number of models with the same hyperparameters using a bar chart. If there are, for example, two models with a batch size of 64, the bar chart will have one bar with length two. The user can take it, deselect models, and see how many models are left to compare, which changes the bars and grays out the number of excluded models. Histograms summarize how many models fall in each of the bins in terms of metrics and scores (T7). If a particular model gets deselected, the bars gray out, depending on their clustering scores and metrics. Users can also hide these bars and bring them back if needed. And if fewer than five models are selected, they can click the "Compare Models" button. First, two scatter plots for each model selected and seven line charts appear, while the bar charts will be hidden. These scatter plots are side-by-side, allowing the user to compare the different embeddings for the various models (T8). The upper left line chart serves as an epoch slider to go through the epochs either by clicking statically or by using the slider dynamically (T1, T2). It

shows the loss for all selected models over the model training (T6). Line charts on the bottom left depict different scores over the training epochs for each model, indicated by the same color as in the model information table (T7). Furthermore, there are two scatter plots for each model, which are aligned. The left scatter plot applies a dimensionality reduction method to the raw data before the first training epoch. Users can see how well their algorithm performed compared to when they do not use any deep clustering algorithm (T1). Right next to it is another scatter plot, which shows the clustering for the current selected epoch. Above each scatter plot, there are two rows with buttons from which the user can select one option from any row. The first row shows four different dimensionality reduction methods from which the user can choose how to reduce the clustering to two dimensions (T3). In the second row, the user can choose which type of coloring the clustering should have, namely no coloring if he wants to investigate how the data develops over the training epochs, ground truth coloring that colors after the gold standard labels if provided, k-Means clustering which colors the labels according to a k-Means algorithm at the chosen epoch, and dec clustering, short for deep clustering, which colors the data according to the last labels found by the deep clustering algorithm after the last epoch (T9). With a click on the right scatter plot, a static scatter plot matrix on the right, and a confusion matrix under the last model is shown. The confusion matrix depicts the found labels in contrast to the ground truth labels (T9). The scatter plot matrix gives a static overview of the embedding throughout the entire training process (T1). There are three rows in each scatter plot matrix, for three different dimensionality reduction methods on the data, and each column is a training epoch. Users can then set a sampling interval of the training epoch in the box above the scatter plot matrices. Again, three buttons to show the coloring for computed or ground truth labels are above each scatter plot matrix. Users can observe how the clustering evolves throughout the training at a glance, rather than only through dynamic interaction. A red triangle links to the current epoch chosen in the line chart. The background color of each model is consistent across all plots in the dashboard, indicating which plots belong to which model.

Feedback: The feedback from domain experts identified layout and interactivity as the primary areas for improvement. In terms of layout, related visual components, such as confusion matrices for the same model, were placed far apart, making coherent model-specific analysis harder. The absence of dedicated confusion matrices for each model limited comparative analysis. For interactivity, the click-based interface for retrieving model details was unintuitive and led to confusion. The currently active dimensionality reduction method and labeling were not indicated, and there was no functionality for misclustering identification, restricting error exploration. It was suggested to improve this high-fidelity prototype further until a version for usability testing is finished.

5.5.2.3. Second Iteration of High-fidelity Prototypes

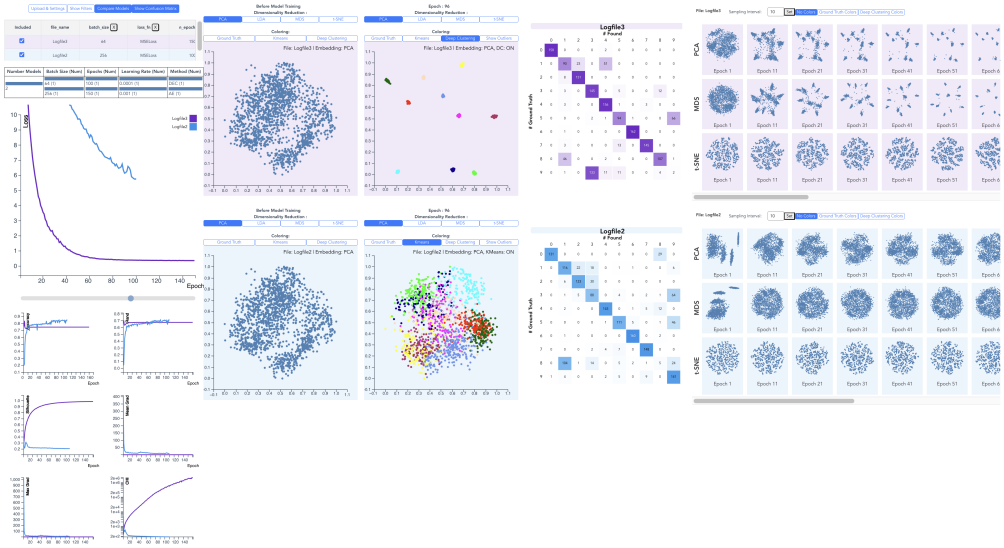

Figure 5.18.: High-fidelity prototype for evaluation

Next, minor updates were conducted for the high-fidelity prototype (see Figure 5.18), which was handed out to the participants of the usability study (see chapter 8). These updates were only for layout purposes, while keeping the tasks that can be fulfilled identical to the previous prototype (see Figure 5.17). These layout changes included making confusion matrices visible for all models at the same time in the same row, activated by an additional button. Therefore, the left side of the dashboard, containing the line chart, line chart matrix, model table, and table chart, shrank in size, resulting in the line chart matrix's order to go from two rows and three columns to three rows and two columns. Additionally, the whitespace between the scatter plots in the middle of the dashboard was minimized. At last, the button layout changed for both of the scatter plots. Feedback for this prototype can be found in detail in chapter 8.

5.5.2.4. Final Implementation

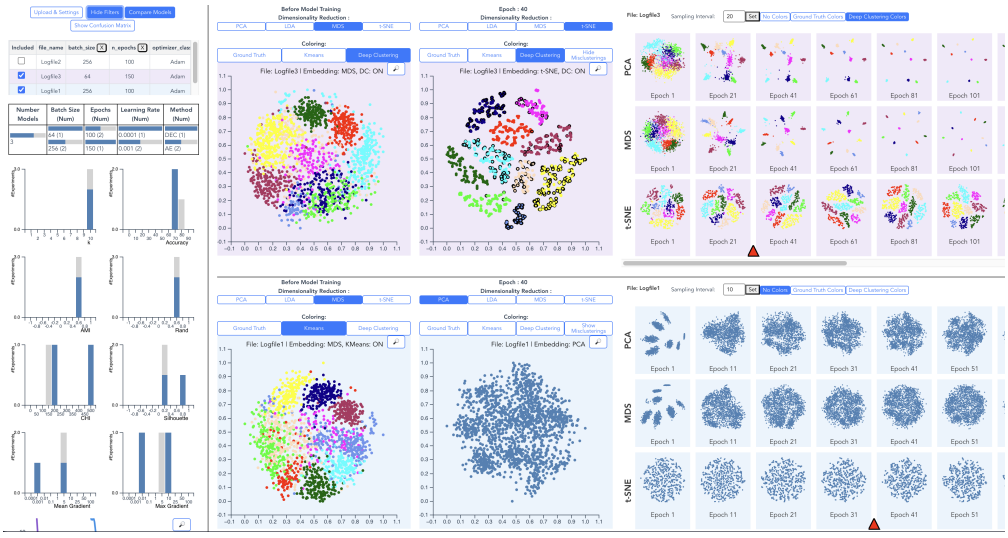

Figure 5.19.: Final Prototype

The prototype used in this stage (see Figure 5.19) includes the implemented improvements based on the results of the evaluation (see chapter 8). For a comprehensive overview of these modifications, see chapter 6. The most important changes include the ability to toggle the confusion matrix, which now replaces the scatter plot at epoch zero to improve the scaling of these components. Additionally, scatter plots and line charts can be viewed in full-screen mode by clicking a button. Furthermore, borders have been introduced to clarify the distinction between the models.

5.6. Implementation

During the first meeting, one of the domain experts presented ClustPy (see subsection 5.6.1), an internally developed tool. ClustPy became the foundation for data pre-processing and model training throughout this project. Since one of the domain experts was actively involved in ClustPy's development, we decided to extend it to support the needs of the interactive visualization dashboard. There were three options for us to modify ClustPy to gather the data needed for an interactive visualization interface.

The first option is to log everything using callbacks within ClustPy. In this approach, a custom ClustPy logger class (see Listing A.1) was implemented to hook into the training process via callbacks. This logger captures embeddings, loss values, clustering assignments, and metric scores at each epoch and stores them in a JSON file for use in the dashboard. This approach fulfills Design Goal 1 by performing all time-intensive computations during the training. It enables a fast and dynamic exploration of the data in the dashboard, since no post-processing or other computations are needed. On the other hand, using this

approach increases the training time due to the logging overhead.

The second option is to log only essential data and compute the rest in a separate post-processing step. This approach would record only minimal information during training, such as the embedding, and in a separate step, perform the necessary data extraction and transformation afterward, such as the dimensionality reduction. With that, this approach keeps training fast and logging lightweight and reduces runtime overhead during model training. On the contrary, it introduces an extra processing step that delays visualization and adds unnecessary complexity to the workflow.

The third option is to log only essential data and compute all transformations on-the-fly in the dashboard. In this option, the dashboard itself would compute all dimensionality-reduced embeddings, clustering results, and metrics based on model outputs. It would keep training and logging very simple, but would introduce heavy computations to the frontend. This was tested and resulted in long computations, leading to poor usability since the time between jumping epochs increased to 20 seconds, preventing dynamic investigations.

To fulfill Design Goal 1, we had to ensure that all necessary data for visualization was already available when the dashboard was launched. In particular, interactive features such as navigating embeddings over epochs or switching between dimensionality reduction techniques would not be feasible if the dashboard had to compute this data on demand. Therefore, the decision was made to go with the first option, because options two and three lack sufficient data on demand, making dynamic investigations impossible.

5.6.1. ClustPy

ClustPy is a Python library [clu] for advanced clustering algorithms. The package provides a simple way to perform clustering in Python. ClustPy provides a variety of algorithms from different domains, including partition-based clustering, density-based clustering, hierarchical clustering, deep clustering, and some alternative clustering methods. It also includes methods often needed for research purposes, such as plots, clustering metrics, or evaluation methods. Further, it integrates various frequently used datasets through largely automated loading options. ClustPy does not focus on efficiency but on the possibility of trying out a wide range of modern scientific clustering methods. This should also make lesser-known clustering methods, like Auto-encoder Based Data Clustering (AEC) [SLH+13] or Deep Embedded Cluster Tree (DeepECT) [MPB20], accessible simply and conveniently [clu].

Since the deep clustering methods should be investigated during the training, and the ClustPy code should not be heavily modified, the choice was to make a logger and add callbacks within ClustPy if the user decides to log. All vital information for the dashboard is logged and saved in JSON format. The modification was then shown to the domain experts and approved by both of them. Each time a user trains an Autoencoder, at each training epoch, the reduced embeddings with four different dimensionality reduction methods, the loss, different metrics, and the k-Means labels are logged. Furthermore, the original data is reduced and logged, so the user can investigate how the original data looks in the space compared to the reduced data by a neural network. At the end of the training

process, the model information and the computed labels by the clustering algorithm are logged, as well as the ground truth labels. Each log file for a model can be loaded into the visualization tool that is based solely on the Vue.js framework and the D3 library. The implementation of the described workflow is outlined in the pseudocode presented in Listing 5.1, while the complete source code is provided in the Appendix Listing A.1.

```
for each model:
    for each epoch:
        normalize embedding
        calculate k-Means
        dimensionality reduction with PCA, LDA, MDS, t-SNE to 2
    components
        calculate Loss, Accuracy, AMI, ARI, CHI, Silhouette Score
        log reduced embeddings, K-Means labels, metrics

    log model hyperparameters
    log deep clustering algorithm labels
```

Listing 5.1: Pseudocode Logger Class

6. Introduction to Clustorama

Some of the feedback from chapter 8 was integrated, resulting in the final dashboard design for this thesis. Below, the tool with every element and interaction is introduced. Furthermore, design decisions made for Clustorama are presented, alongside the tasks and design goals the dashboard fulfills.

6.1. Start Screen

When the user first opens the dashboard, there is no data to be shown, since nothing has been uploaded yet. The user is presented with a nearly empty screen with the empty Model Tables, the empty Bar Chart Table, and four buttons in the top left corner of the dashboard (see Figure 6.1). Two of the buttons are deactivated, which are the "Compare Models" and "Show Confusion Matrix" buttons. They activate if there are between one and five models uploaded and included in the tables. Two buttons are active from the start, which are "Upload & Settings" as well as the "Hide Filters" button. The "Upload & Settings" button opens the Sidebar, and the "Hide Filter" button toggles the visibility of the Bar Chart Matrix after models are uploaded.

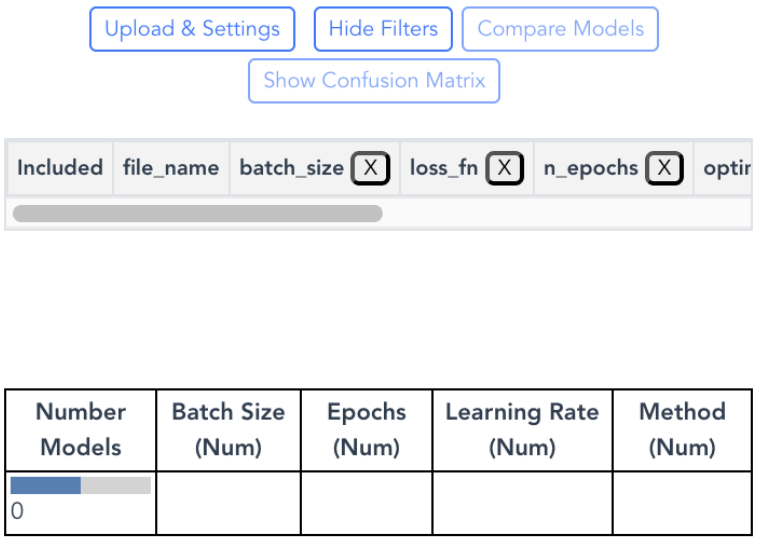

Figure 6.1.: Zoomed in on existing Components on Start Screen

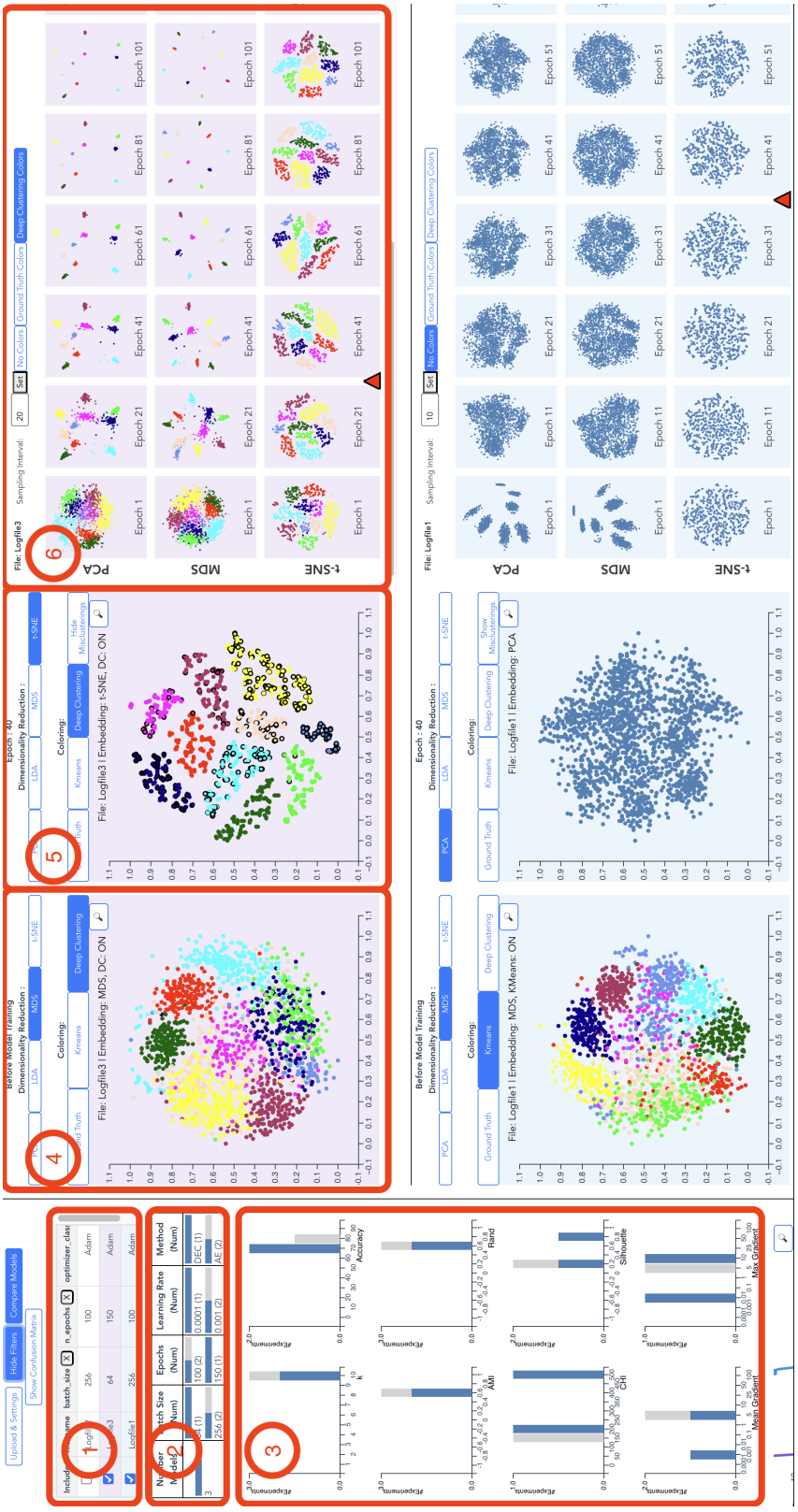

Figure 6.2.: Clustorama with filters and epoch zero scatter plot

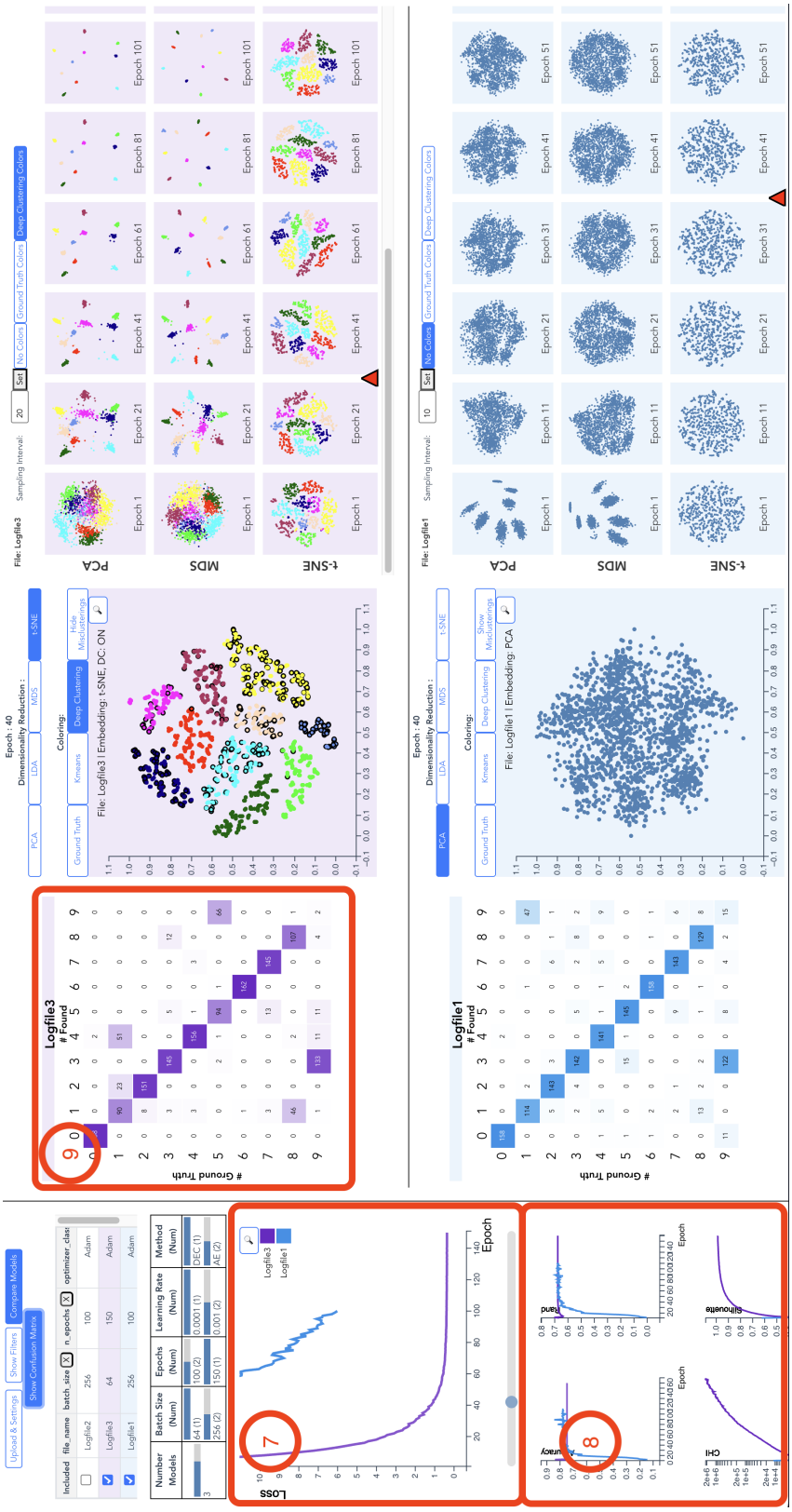

Figure 6.3.: Clustorama with confusion matrix and line charts

6.2. Sidebar

The Sidebar opens by clicking the "Upload & Settings" button. It can be closed by either clicking on the close button in the bottom right corner or by clicking in an empty area outside of the Sidebar component. First, the user is presented with information about ClustPy and the new ClustPy Logger class. These are needed to create log files of the models that are uploaded into the dashboard, for inspection during training, and to integrate seamlessly with their current workflow (DG2). There are two links to the respective GitHub repositories. Next, there is a section for the user to upload model log files. They have to be in the .json format, and the user can upload multiple models at the same time, to avoid unnecessary computations and data transformations in the tool (DG1). Below that, the next section in the Sidebar component can be used if the data set contains image data. Users can input the dimensions in pixels of the picture in the data set, for example, 28 x 28 for the MNIST digits data set. After clicking the "Save" button, hovering over a data point in the scatter plots shows the image from the data set. Lastly, there are two buttons to reset the column deletion in the Model Table component and to open the Documentation component. The Documentation component contains information on the dashboard components and provides an overview of the intended workflow.

6.3. Model Table

The Model Table component (see 1 in Figure 6.2) is empty when loading the dashboard. After uploading models via the Sidebar component, they appear in the Model Table component, as one model per row, allowing for the upload of models with different clustering algorithms with different hyperparameter settings (T4 & T5). The model information, such as hyperparameters, the loss function (T6), algorithm names, and model names, is extracted and placed in the correct columns, giving the user an overview of the different models. Users can delete columns from the table by clicking the "X" in the table header, for example, if the information is the same for all models and therefore does not provide any additional value. Deleted columns can be restored with a button at the bottom of the Sidebar component. The first column of the table includes checkboxes, which are ticked after a model is uploaded and can be unticked by the user. They indicate if a model is included for further investigation or if it is discarded due to bad metrics. Therefore, it serves the user as a filter function. If the user includes between one and five models, the "Compare Models" button at the top of the screen activates. It allows the user to have an in-depth view of the included models with various components. After clicking on the "Compare Models" button, the included models in the Model Table component get assigned a color that remains the same across all other components in the dashboard and signals the affiliation to the model.

6.4. Bar Chart Table

Like the Model Table component, the Bar Chart Table component (see 2 in Figure 6.2) is also empty when loading the dashboard. After uploading models via the Sidebar component, they appear in the Bar Chart Table component, as stacked bars based on the hyperparameter or information bar they fall into (T5). It covers the most critical information for the domain experts, including the total number of models, the batch size, the number of epochs, the learning rate, and the method used in the deep clustering approach (T4). Each blue bar covers one bin the value falls into, for example, if there are two models uploaded with 100 epochs each, there is one bar in the epochs column with a length of two. On the contrary, if there is one model with 100 epochs and one model with 150 epochs, there are two blue bars, each with a length of one. In brackets behind the bin are the total number of models that fall into this particular bin. Furthermore, if a user decides to exclude a model via the Model Table component, the bar gets grayed out for the number of excluded models. For example, if there are two models with 100 epochs and the user decides to exclude one model via the Model Table, the bar of length two gets altered into a blue bar of length one and a gray bar of length one.

6.5. Bar Chart Matrix

The Bar Chart Matrix component (see 3 in Figure 6.2) appears after the first model is uploaded into the dashboard. It contains eight histograms that depict the state of the model in the last epoch of the training (T7). The "k" histograms visualizes the number of clusters, "Accuracy" measures the accuracy in the last epoch of the training, "AMI" is the adjusted mutual information score, "Rand" is a histogram for the Rand Index, "CHI" stands for the Calinski–Harabasz Index, and "Silhouette" is the Silhouette Score. "Mean Gradient" and "Max Gradient" inform the user about the gradients of the whole neural network during the training. Users can hover over the bars of the histogram and get further information. Four different values appear for "Range", "#Experiments", "Included", and "Excluded". "Range" describes the bin the user is hovering over, for example, "Range: 80" on the accuracy histogram shows the values for models with an accuracy between 80% and 90%. The "#Experiments" value shows the user the number of models that fall into this bin, with "Included" showing the names of the models in the bin. "Excluded" shows the number of models in a bin that are excluded via the Model Table component. Additionally, suppose a model gets excluded via the Model Table component. In that case, the part gets grayed out from the bins the model falls into in the different histograms.

6.6. Line Chart

The Line Chart component (see 7 in Figure 6.3) appears after the user clicks on the "Compare Models" button and provides the user with an overview of the loss development over the whole training process, depending on the loss used during the training (T6).

Each line represents one model and is colored according to the color assigned to the model in the Model Table component (T8). A legend in the top right corner additionally helps the user to keep track of the color assignment. Underneath the line chart, the user can use a slider to switch the training epoch for all components in the dashboard (T1). Hovering over the line chart creates a vertical line for the user to better track the loss with the values on the x-axis (T2). Additionally, a tooltip shows up to give the user an overview of the current epoch he is hovering over, and the loss values for each model at the current epoch. Furthermore, the minimum loss, together with the epoch, is presented to the user in brackets. Clicking within the line chart makes the selected epoch jump to the x value in all other components (T1). In the top right corner, the user can find a button with a magnifying glass to expand the line chart to the full screen. Lastly, the user can zoom in on the line chart by holding the "Ctrl" button and using the mouse wheel or touchpad.

6.7. Line Chart Matrix

The Line Chart Matrix component (see 8 in Figure 6.3) appears together with the Line Chart component after the user clicks on the "Compare Models" button. It provides an overview of different metrics over the whole training process (T7), for all models that are included, enabling a side-by-side comparison for the metrics (T8). These metrics include Accuracy, Rand Index, CHI, Silhouette score, and the gradients. Therefore, they cover a subset of the matrices from the Bar Chart Matrix Component across the entire training process, and not just the last epoch. On hover, the user gets presented with a similar tooltip to that in the Line Chart Component.

6.8. Scatter Plot before Training

The Scatter Plot component (see 4 in Figure 6.2) for the original, unprocessed data appears after the user clicks on the "Compare Models" button. It shows the original, unprocessed data broken down into two dimensions with dimensionality reduction techniques. Therefore, the user gets an overview of what the original, unprocessed data would look like without running it through a deep neural network (T10). When the scatter plot appears, it presents the data without coloring, broken down into two dimensions with Principal Component Analysis. Two rows of buttons allow the user to change the labels and the dimensionality reduction method that is applied to the data. The first row lets the user choose between four dimensionality reduction methods: PCA, LDA, MDS, and t-SNE (T3). Changing the dimensionality reduction method without the coloring results in the button of the current method getting highlighted, and the color of the data changes. The second row of buttons can be activated to change the coloring based on the found labels for the chosen dimensionality-reduced data. Users can choose between no labels at all, the ground truth labels, k-Means labels for the current epoch, or the labels of the deep clustering algorithm found at the last epoch (T4). These labels are calculated on the high-dimensional data before it is reduced to two dimensions. Selected dimensionality reduction techniques, as well as labels and the filename, are shown at the

top of the scatter plot. The background coloring is the same as the color assigned in the Model Table component, signaling the user affiliation of the plots. Hovering over a data point creates a tooltip that provides the user with further information about the specific data point (DG5). The user sees the data point coordinates as well as the index in the training dataset. Furthermore, if a coloring other than "Ground Truth" is selected, the found label by the respective algorithm, as well as the ground truth label, is shown. With that information, the user can determine if data points are assigned to the correct clusters (T9). In addition to that, if the user is working on image data and has set the dimensions of the images via the Sidebar component, a small picture gets shown in the tooltip, together with information about a match or mismatch between ground truth and found label. This helps the user to identify misclustered data points and determine whether the misclustering was due to data quality or the algorithm used. Like the Line Chart component, the scatter plots also have a button with a magnifying glass. Clicking the button expands the scatter plot for the size of the whole dashboard. Holding the "Ctrl" button also allows the user to zoom in on the data, either with the mouse wheel or the touchpad.

6.9. Training Scatter Plot

The Training Scatter Plot component (see 5 in Figure 6.2) adopts the functions of the scatter plot for the original, unprocessed data, and extends some of them (T10). This scatter plot shows the embedded data in the latent space for the current selected epoch. The selected epoch is assigned via the Line Chart component, either by clicking in the chart or using the slider underneath. By dragging the slider in the line chart, the user can dynamically inspect the embedding throughout the training (T1). The currently selected epoch gets shown in the first row of text above the scatter plot. In the second row of buttons, the user can find an additional button to hide and show misclustered data points. Clicking the button colors the border of data points, where the found and ground truth labels do not match, in black. This function makes it easier for the user to identify single misclustered data points and assess the overall clustering quality (T9 & DG5). Otherwise, dimensionality reduction methods are PCA, LDA, MDS, and t-SNE, like in the scatter plot component for the original, unprocessed data (T3). The coloring options are also the same, with no labeling, ground truth labeling, k-Means labeling, and labeling after the found labels of the deep clustering method (T4). Users can also expand the scatter plot over the whole screen by clicking the magnifying glass button. Selected options get shown as text, together with the filename at the top of the scatter plot. The background is the same for each model, as they get assigned in the Model Table component. Hovering over a data point provides the user with the position, the ground truth label, and the index (DG5). If labels other than ground truth labels are selected, both labels are shown side by side, allowing the user to check for any mismatches. Furthermore, if the user is working on an image dataset, they can set the dimensions of the images in the Sidebar component. Afterward, these pictures are shown if the user hovers over a data point, together with an indication of a match between ground truth and selected label. Lastly,

the user can zoom in on both small and full-screen views of the scatter plot by holding the "Ctrl" button and simultaneously using the touchpad or mouse wheel.

6.10. Scatter Plot Matrix

The Scatter Plot Matrix component (see 6 in Figure 6.2) gives the user a static overview of the embeddings over the whole training process at once (T1 & T2). Furthermore, it lets the user compare the different dimensionality reduction methods' ability to separate the clusters visually (T3). It comprises small multiples of the Training Scatter plot for the three most used dimensionality reduction methods from the domain experts, which are PCA, MDS, and t-SNE (T8). The scatter plot matrix appears after the user clicks the "Compare Models" button, with every tenth epoch of the training shown as a small multiple. The user can find the name of the model he is looking at in the top left corner as text, with the background of the small multiples being colored the same color the model gets assigned in the Model Table component. At the bottom of each small multiple is a text indicating the epoch that the small multiple depicts. Next to the model name, the user can find several options. The sampling interval is initially set to ten and indicates the number of small multiples to be created. For example, if the training is 100 epochs long, the user gets shown ten small multiples for each dimensionality reduction method. The user can set the number of small multiples shown with the box next to it, and click the "Set" button. Setting the sampling interval, for example, to 20, then only shows every twentieth epoch of the training, starting with the first. Next to the sampling interval, the user can find three more buttons to set the labels for the small multiples. Users can select between no labels, ground truth labels, and the labels found by the deep clustering algorithm (T4 & T5). Selecting one of these applies it to all the small multiples at the same time. Therefore, it not only shows the evolution of the embeddings throughout the training, but also the evolution of the particular clusters. If the user utilizes the slider in the Line Chart component, a red triangle spawns, indicating where the current epoch selected is on the scatter plot matrix. If the user selects, for example, epoch 30 in the scatter plot, a red triangle spawns between the two small multiples closest to epoch 30. This indicates to the user the past and future development of embeddings.

6.11. Confusion Matrix

The Confusion Matrix component (see 9 in Figure 6.3) is displayed after the user clicks the "Compare Models" button and then the "Show Confusion Matrix" button. It replaces the Scatter Plot component for the original, unprocessed data. This view can be swapped back if the user clicks the same button again. Each included model has one confusion matrix, with the model's name shown at the top of the matrix. In addition to that, the confusion matrix keeps the same color that the model gets assigned in the Model Table component. Each column shows the ground truth label for a found label, and vice versa; each row shows the found labels for each ground truth label. Higher values have a higher

background saturation for their cell. The higher the values in the cells on the diagonal are compared to values in cells not on the diagonal, the better the clustering is.

6.12. Design Decisions

This section presents the findings from feedback sessions and user testing of the dashboard, designed using rapid prototyping. Feedback from thesis advisors and domain experts was instrumental in refining the dashboard's design. The reasoning for choosing specific visual encodings for different dashboard components is explained below.

Why is the dashboard split into two parts/columns?

The dashboard is intentionally divided into two main sections, arranged side by side. The left column provides users with an overview of all uploaded models, allowing them to quickly grasp the available data and assess general trends or patterns. In contrast, the right column is dedicated to more detailed information about selected models for visualization. This layout supports a natural user workflow that begins with exploration and gradually transitions into more focused, in-depth analysis. By structuring the interface this way, the design follows Shneiderman's "overview first, zoom and filter, then details-on-demand" [Shn03], a well-established principle in the field of information visualization that promotes intuitive and efficient interaction with complex data.

Why are settings and model uploads in a collapsible sidebar?

Settings and model uploads are placed in a collapsible sidebar to reflect their auxiliary role in the overall workflow. These elements are primarily configured at the beginning of the analysis process and are typically not needed afterward. By making the sidebar collapsible, users can hide these controls once the setup is complete, thereby reducing visual clutter and maintaining focus on the central analysis components of the dashboard. This approach promotes a cleaner, more user-friendly interface by separating configuration from exploration and interpretation tasks.

Why does the bar chart matrix collapse after comparing the models?

Another key design decision involves the behavior of the bar chart matrix, which is used during the initial stage of model comparison. This matrix allows users to filter and select the models that are most relevant to their interests. Once this filtering is complete and a comparison is initiated, the matrix collapses to free up screen space for more detailed visualizations. At this stage, the line charts take over as the primary visual representation, offering a finer-grained view of the same data with greater clarity and interpretability. Since the essential comparative information from the bar charts is carried forward and expanded upon in the line charts, keeping the matrix visible would add unnecessary redundancy and visual noise.

Why does the scatter plot matrix not contain all the dimensionality reduction methods of the scatter plots?

The scatter plot matrix was designed to align visually and structurally with the individual scatter plots. In the dashboard layout, the scatter plot matrix and the scatter plots are displayed in the same row to facilitate seamless comparison. Including an additional dimensionality reduction method in the scatter plot matrix would have introduced significant layout constraints. If another projection were added, either the individual plots within the scatter plot matrix would have to be reduced in size, making them harder to interpret, or the overall scatter plot matrix component would need to be enlarged. The former would cause the individual data points in the scatter plot not to be distinguishable. The latter would reduce the number of epochs that could be shown simultaneously, thereby negatively impacting users' ability to compare multiple epochs. Furthermore, in typical screen resolutions from 13 up to 27 inches, increasing the width of the scatter plot matrix would result in only one model being visible at a time, eliminating the possibility of effective side-by-side comparison. Given these limitations, the decision was made to restrict the scatter plot matrix to the three most commonly used dimensionality reduction techniques among the target users. This compromise ensures usability without sacrificing comparability.

Why is the line chart used to change epochs?

The line chart was selected as the primary interface for changing epochs due to its strong association with loss development across training epochs. Since users are typically interested in identifying epochs where model performance changes significantly, it is natural for their attention to focus on the loss curve (T2). Placing the epoch selection mechanism directly within and underneath the line chart enables users to cross-reference epoch indices and loss behavior without switching contexts. A vertical line indicator in combination with a tooltip supports this interaction by providing a precise visual reference for the currently hovered epoch. This design makes it easier to target a specific epoch accurately, reducing the likelihood of user error or approximate selection. The integrated approach reinforces intuitive interaction with the model training timeline and supports precise temporal analysis.

Why only allow details for a maximum of five models to be plotted?

To avoid overwhelming users and maintain readability, the system limits the number of models shown in detailed views to a maximum of five. Each model is assigned a unique color for identification across visualizations (DG4). However, as the number of models increases, the color palette becomes saturated, leading to confusion and visual overload. The user workflow is designed first to filter down the initial model set based on high-level comparisons, and then to analyze a small subset of promising models in detail. This aligns with the dashboard's goal of helping users identify the best-performing model. Limiting detailed views to five models enforces a structured and manageable comparison process,

ensuring clarity and interpretability of the visualizations.

Why does the user need to scroll if he plots more than two models, instead of scaling the plots to a smaller size?

When users choose to compare more than two models in detail, the dashboard introduces horizontal scrolling rather than scaling down the size of the visualizations. This decision was based on discussions with domain experts, who emphasized the importance of being able to view two models side by side in full detail (T8). Their typical workflow involves comparing two models, identifying the weaker one, and eliminating it iteratively. This process relies heavily on side-by-side comparison to make informed judgments. While displaying three models side-by-side is technically feasible on larger screens, typically 23 inches and above. On smaller screens, doing so requires downscaling the individual components, which results in a significant reduction in readability and interpretability. Visual elements become harder to distinguish, and critical information may be lost or overlooked. In interviews, domain experts expressed a strong preference for larger, more readable components, even if this meant requiring horizontal scrolling when comparing more than two models on a smaller screen. As a result, the design prioritizes clarity and visual fidelity over fitting all content on the screen at once. This approach ensures that the quality of each visualization is preserved, supporting accurate comparisons and better decision-making during model evaluation.

Why can the user not display the epoch zero scatter plot and the confusion matrix at the same time?

The epoch zero scatter plot and the confusion matrix serve different purposes at distinct stages of the workflow. The epoch zero scatter plot is primarily used at the beginning of the exploration process to visualize the unprocessed model embeddings before training has had any effect. This allows users to assess the structure of the initial data and observe how it evolves throughout the training process. In contrast, the confusion matrix is typically used at the end of the workflow. Domain experts apply it during the final evaluation phase to assess model performance, particularly in relation to clustering accuracy and misclusterings. At this point, the initial state of the embeddings is no longer relevant, and focus shifts toward understanding how well the model clustered the embedded data. To optimize screen space and improve readability, the decision was made to replace the epoch zero scatter plot with the confusion matrix. This enables the confusion matrix to be rendered at a larger scale, enhancing readability. Additionally, placing the confusion matrix side by side with the epoch's scatter plot allows for more effective identification of misclustered regions or problematic data points. This juxtaposition helps users connect clustering performance with label evaluation, ultimately supporting a deeper qualitative assessment of the model. Overall, the exchange of both visualizations integrates well with the user's workflow, since both are needed at different stages of the evaluation (DG2).

Why does the line chart matrix not cover all the metrics from the bar chart metrics?

The line chart matrix was designed to visualize the evolution of clustering metrics throughout model training. However, not all metrics from the bar chart matrix were carried over to the line chart matrix. One such example is the number of clusters "k", which remains constant for each model and does not change across epochs. Since line charts in the dashboard are intended to represent evolution over time, plotting a static value such as k would not offer any additional insight and could introduce unnecessary visual clutter. After excluding this non-temporal metric, a design decision had to be made regarding which of the remaining metrics should be included. During feedback sessions, domain experts expressed a preference for a clean and symmetrical layout within the line chart matrix. To maintain this structure and avoid underutilized visual space, the RAND index was identified as the least used metric by the experts and was excluded from the final design. This trade-off prioritizes clarity and usability, ensuring that the most relevant information is displayed without overwhelming the user or compromising the visual harmony of the matrix.

Why is the coloring only applied after the models should be compared?

Color encoding is deliberately introduced only after the user has selected models for comparison. This decision is rooted in the scalability of the tool, which is designed to handle dozens of uploaded models simultaneously (DG3). Applying distinct colors to all models from the beginning, for example, in the model table, would result in color repetition due to the limited number of distinguishable hues. This repetition would likely confuse rather than improve clarity. Moreover, before the comparison step, the interface does not present any detailed views where color encoding would serve a meaningful purpose. At that stage, the components focus on high-level filtering or selection rather than visual differentiation. Introducing color prematurely would not convey meaningful information and could instead distract or mislead users. Once a subset of models is selected for detailed comparison, colors are applied consistently across visualizations to establish a strong visual identity for each model (DG4). Hue is used to uniquely distinguish models, while luminance and saturation levels are adjusted based on visual context. For example, lower saturation or brightness is used in the background of elements to indicate shared affiliation across components, while more vivid colors highlight active marks within a specific visualization. This layered color strategy enhances interpretability without overwhelming the user and reinforces model identity throughout the comparison process.

7. Use Case Scenarios

The following chapter presents a set of use case scenarios designed to illustrate the practical application and capabilities of the developed data visualization dashboard for deep clustering analysis. Each scenario is derived from realistic tasks the users encounter when working with deep clustering algorithms, hyperparameter optimization, and latent space interpretation. The use cases demonstrate how the dashboard supports various analytical objectives, such as selecting the most suitable clustering algorithm, understanding the influence of hyperparameters on model performance, exploring the structural evolution of embeddings in latent space, and investigating the behavior of individual data points during training. By integrating multiple interactive visualizations, the dashboard enables users to combine quantitative performance metrics with qualitative visual inspection. Together, these scenarios provide a comprehensive overview of the dashboard's role as an exploratory and diagnostic tool for deep clustering workflows, highlighting its potential to facilitate decision-making, enhance interpretability, and optimize both model configuration and training efficiency.

7.1. $RQ_{Cluster}$ Scenario 1: Comparison and Optimization of Clustering Algorithms

A user aims to gather insights to support the training and optimization of deep clustering algorithms and models. The starting point is a given dataset, for which it is unclear which deep clustering algorithm yields the best performance. To address this, the user experiments with several non-optimized models from different deep clustering algorithms and loads the log files in the developed dashboard. Using the filter mechanism, he filters out models that perform poorly over all presented metrics until he has at most five models left by using the histograms and the model table. The visualizations enable direct comparison of algorithm and model performance, beyond simple numbers, to prevent situations like Anscombe's Quartet [RSVR18]. In this case, a deep clustering approach has a higher accuracy, but a much worse overall performance, for example, in terms of cluster separation. In particular, differences in embeddings, misclusterings, and relevant metrics become apparent. The user views the embedding development throughout the training, using the training scatter plot and the scatter plot matrix. He identifies the deep clustering algorithm that separates the clusters best and compares it to the unprocessed data to see if the algorithm is needed. Based on these insights, the user selects the most promising algorithm and performs a hyperparameter optimization using grid search. The resulting training logs are then loaded into the dashboard. By repeating the process for models with the same deep clustering algorithm and filtering out the poorly performing

model configurations, the user can identify optimal parameter combinations. Additionally, by comparing different training trajectories, it becomes evident how many epochs are necessary to achieve meaningful separation of clusters in latent space. This enables further optimization of training duration and computational resources, for example, by applying early stopping when no significant improvement can be observed.

7.2. $RQ_{Cluster}$ Scenario 2: Analyzing the Influence of Hyperparameters on Clustering Performance

A user aims to gain a deeper understanding of how different hyperparameter configurations influence the clustering process and model behavior. To this end, the user performs a grid search focused on the hyperparameters of interest and loads the resulting training log files into the dashboard. Within the tool, the user compares multiple models to investigate how variations in specific hyperparameters affect the clustering outcome and how quickly meaningful cluster separation is achieved. This is facilitated by visualizations such as scatter plots and scatter plot matrices, which reveal differences in the structure of the latent space across models. In addition, the user examines how hyperparameter changes impact model performance in terms of loss, accuracy, and other relevant metrics using line charts and line chart matrices. The confusion matrix further reveals how certain hyperparameter configurations lead to better or worse identification of specific clusters. Through this comprehensive visual analysis, the user gains detailed insights into the relationship between hyperparameters and model behavior, enabling more informed decisions when tuning deep clustering algorithms and models.

7.3. RQ_{Latent} Scenario 3: Understanding Embedding Dynamics in Latent Space During Training

A user seeks to develop a deeper understanding of the latent space and how the learned embeddings evolve during the training process of a deep clustering model. To achieve this, the user uploads a log file generated for a benchmarked deep clustering algorithm that has been trained with optimized hyperparameter settings. Using the visualization dashboard, the user explores how the latent space embeddings change throughout training epochs, particularly in relation to the decreasing loss. He turns on the clustering labels predicted by the deep clustering algorithm and observes the progression of cluster separation in the latent space. Through this analysis, the user recognizes how the reduction in loss correlates with improved structural organization in the latent space, gaining insights into how the model incrementally refines its representation to bring data points belonging to the same cluster closer together. Furthermore, he analyzes the spatial position of the

clusters in the latent space in relation to each other. He gains insights into how the deep clustering algorithm places more related clusters closer to each other in the latent space. This provides valuable interpretability regarding the model's training dynamics and the objectives it implicitly pursues during optimization.

7.4. RQ_{Latent} Scenario 4: Investigating the Behavior of Individual Data Points in Latent Space During Training

A user aims to gain a deeper understanding of how specific data points behave in the latent space throughout the training process of a deep clustering model. To this end, the user uploads a log file for a benchmarked deep clustering algorithm that was trained with optimized hyperparameters on an image dataset. Within the dashboard, the user enables misclustering identification and image data representation and visualizes the labels predicted by the deep clustering algorithm. By dynamically navigating through the training epochs, the user observes the evolution of the latent space embeddings. During this exploration, the user identifies that specific misclustered data points were separated and assigned to incorrect clusters later in the training process. A detailed investigation of these misclustered points reveals that they initially resided spatially closer to their correct cluster but were gradually shifted toward another cluster as training progressed. The user sees how the assigned probability for a cluster by the deep clustering algorithm changes the spatial position of the data point embedding during the training. This analysis provides valuable insights into the temporal and spatial dynamics of clustering behavior at the individual data point level in the latent space and highlights potential limitations or instability in the model's training process.

8. Evaluation

The following section contains the evaluation methods and results. It was conducted on a high-fidelity prototype of the dashboard (see Figure 8.1).

8.1. Data Set

For the evaluation, the users are presented with five log files of different settings of hyperparameters as well as several different deep clustering methods (see Table 8.1). Since the tool is created to inspect the model training and optimize the deep clustering algorithm and its hyperparameter settings, the dataset on which the deep clustering was performed remains the same. For this test setting, the UCI ML hand-written digits data set is used [AK98]. It contains 1797 pictures of size 8 x 8 pixels of hand-written numbers, leading to 64 features per sample, ranging from 0 to 16. There are a total of 10 classes with around 180 samples per class. Three different clustering algorithm setups were logged during the training. These are Feed Forward Autoencoder with k-Means clustering, Deep Embedded Clustering (DEC), and Improved Deep Embedded Clustering (IDEC).

Model ID	Algorithm	Epochs	Batch	Layers	Learning Rate
M1	AE + k-Means	100	256	d-500-500-2000-m	0.001
M2	AE + k-Means	100	256	d-100-200-m	0.001
M3	DEC	100 + 150	256	d-500-500-2000-m	0.001 + 0.0001
M4	DEC	100 + 100	64	d-100-200-m	0.001 + 0.001
M5	IDEC	100 + 100	256	d-500-500-2000-m	0.001 + 0.0001

Table 8.1.: Test models for evaluation

8.2. Tasks

For the usability test, participants were asked to complete a total of 6 tasks that covered the tasks set by the domain experts in the requirements analysis.

T1: Load the given training runs (.json files) and filter out the two worst ones. Which training runs did you filter out and why?

T2: Compare the remaining models and observe the loss curves of your chosen models. Which model has the lowest loss, and what is its value?

T3: Switch between the dimensionality reduction techniques and observe any changes in the embedding of the best-performing model found in T2. Which dimensionality reduction methods separate the clusters best for the given data?

T4: Compare the training runs of all selected models over the whole training process. Which training runs could produce similar clustering results, in terms of visual cluster separation, with a lower number of training epochs?

T5: Set the dimensions of the data set (8 x 8). In your best chosen model, identify a misclustered data point in the latent space and map it back to the original data. Is it misclustered because of the clustering algorithm, or is the data point of low quality?

T6: Display the confusion matrix of your chosen model. Which class has the least misclusterings in your preferred model?

8.3. Users

The users (see Table 8.2) in the conducted usability study included two visualization and UI/UX experts and four clustering and deep clustering experts.

User ID	Gender	Field of Expertise
U1	Male	UI/UX Research
U2	Male	Deep Clustering Research
U3	Female	UI/UX Research
U4	Male	Deep Clustering Research
U5	Female	Clustering Research
U6	Male	Clustering Research

Table 8.2.: Participants of the Usability Study

8.4. Questionnaire

All participants evaluated the tool using the System Usability Scale (SUS) [B+96]. The SUS is a standardized, ten-item Likert-scale questionnaire ranging from 1 (strongly disagree) to 5 (strongly agree), providing a reliable overall measure of perceived usability. This approach ensures a consistent assessment across participant groups and allows for comparability of results.

SUS Questionnaire:

S01 I think that I would like to use this system frequently

S02 I found the system unnecessarily complex

S03 I thought the system was easy to use

S04 I think that I would need the support of a technical person to be able to use this system

S05 I found the various functions in this system were well integrated

S06 I thought there was too much inconsistency in this system

S07 I would imagine that most people would learn to use this system very quickly

S08 I found the system very cumbersome to use

S09 I felt very confident using the system

S10 I needed to learn a lot of things before I could get going with this system

The answers to the six tasks were given in free-text form.

8.5. Procedure

The study was conducted online via Zoom. Participants were provided with access to the dashboard, five log files, a consent form, and a document containing task descriptions, the SUS questionnaire, and open-ended response fields. After signing the consent form, participants were informed that they could share their screen. The session recording began at this point. Each participant read the task aloud before attempting to solve it, indicating the starting point for the time-on-task measurement. Upon completing a task, they were given time to document their answer immediately. They were then asked to reflect on the task by indicating its difficulty, describing what they liked, and providing any additional comments. If a participant was unable to solve a task or requested assistance, support was provided, but only half a point was given to the success rate score for the task. The tasks were solved in the same order for every participant, since they build upon each other and the intended workflow of the domain experts. After completing all tasks, participants were instructed to stop screen sharing and immediately complete the SUS questionnaire. Subsequently, they were asked a series of open-ended questions regarding any confusion or issues encountered during the session, potential improvements, usability concerns, and whether they gained insights into the deep clustering process. Thereby, providing qualitative feedback for the tool. Finally, the session recordings were reviewed to identify bugs, points of uncertainty, and remarkably effective aspects of the dashboard.

8.6. Prototype

The evaluation was based on the prototype outlined in the requirements section (see subsubsection 5.5.2.3):

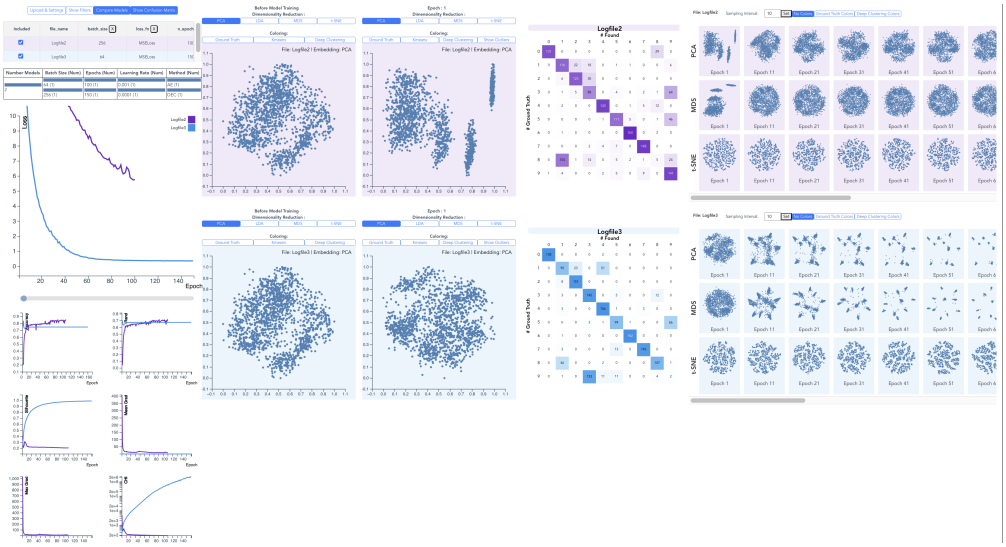

Figure 8.1.: High-fidelity prototype on which the evaluation was conducted

8.7. Metrics

This section discusses the quantitative metrics used to evaluate the usability study. As the first evaluation metric, the **success rate** is calculated as the percentage of users who completed a task in the study. A value of 0.8 or better indicates a generally good experience, whereas a value below 0.7 indicates usability issues with the tool. In this study, one point is awarded for every task that was solved without any help, half a point for every task successfully solved with help, and zero points if the participant cannot solve a task. Next, **time on task** measures how efficiently users perform specific tasks compared to other users.

The **SUS score** is calculated with the following formula :

$$SUS = 2.5 \left(20 + \sum(\text{SUS01}, \text{SUS03}, \text{SUS05}, \text{SUS07}, \text{SUS09}) - \sum(\text{SUS02}, \text{SUS04}, \text{SUS06}, \text{SUS08}, \text{SUS10})\right) \tag{8.1}$$

Its result is a score between 0 and 100, with different articles producing a different grade for each score. Lewis and Sauro created a ranking resembling school grades with a SUS score from 0 - 51.6 resulting in an F grade, 51.7 - 62.6 resulting in a D, 62.7 - 72.5 resulting in a C, 72.6 - 78.8 a B, and everything above an A grade [LS18]. The grades from C to A are also subdivided into C-, C, and C+, with the same division for A and B as well. Bangor, Kortum, and Miller created a separate rating based on an empirical study (see Figure 8.2).

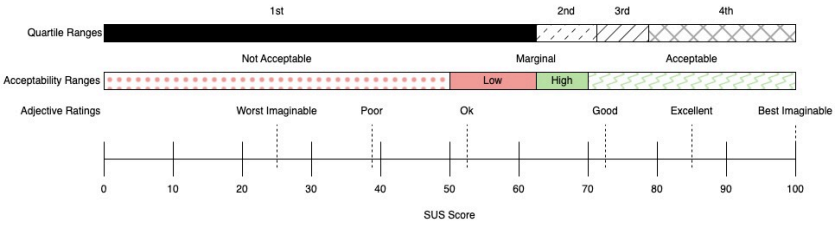

Figure 8.2.: SUS score evaluation [BKM08]

In addition to the quantitative measures described above, qualitative feedback was collected from participants during and after the usability study with open-ended questions. This feedback focused on subjective impressions, perceived strengths and weaknesses of the dashboard, and suggestions for improvements and new features. The qualitative insights complement the numerical metrics by identifying potential usability issues that may not be reflected in the quantitative results alone.

8.8. Quantitative Results

This section presents an objective evaluation of the system's performance using three key metrics: success rate, time on task, and System Usability Scale (SUS) scores. These measures provide insight into the system's efficiency, effectiveness, and perceived usability.

8.8.1. Success Rate

The success rate (see Table 8.3) for the different participants results in a score of 89%, indicating a generally good experience. Most participants were able to complete the tasks without assistance, indicating that the core functionality is generally understandable and reliable. Problems occurred with the model upload of User5, due to his hardware. The solution to the task, which involved filtering out models, was given, so the user could upload the remaining model that suited their hardware. User3 had general problems since his role as a UI/UX researcher with no computer science background led to issues in the domain-specific field of deep clustering. This suggests that non-expert users may need additional onboarding or contextual guidance when working with deep clustering, a domain-specific concept.

User	T1	T2	T3	T4	T5	T6	Overall
U1	1	1	1	1	1	1	$\frac{6}{6} = 1$
U2	1	1	1	1	1	1	$\frac{6}{6} = 1$
U3	0.5	0	1	0.5	1	1	$\frac{4}{6} = 0.67$
U4	1	1	1	1	1	1	$\frac{6}{6} = 1$
U5	0	1	1	1	0.5	1	$\frac{4.5}{6} = 0.75$
U6	1	1	1	1	0.5	1	$\frac{5.5}{6} = 0.92$
Score	$\frac{4.5}{6} = 0.75$	$\frac{5}{6} = 0.83$	$\frac{6}{6} = 1$	$\frac{5.5}{6} = 0.92$	$\frac{5}{6} = 0.83$	$\frac{6}{6} = 1$	0.89

Table 8.3.: Success Rate for different users (1 = Success, 0.5 = Success with help, 0 = Failure)

8.8.2. Time on Task

The time on task (see Table 8.4) gives an overview of how long the users spent on the different tasks. Here, it shows that users spend a lot of time on the first task, averaging over 11 minutes compared to less than 5 minutes for most other tasks. This can be attributed to higher computational demands and task complexity at the start. In contrast, the short execution time for Task 6 indicates that it was both easy to understand and computationally efficient.

User	T1	T2	T3	T4	T5	T6	Overall
U1	14:33	2:15	7:37	9:43	2:30	2:14	38:52
U2	12:17	2:13	3:59	3:15	3:54	0:34	26:12
U3	14:19	4:05	3:46	3:35	7:04	3:36	35:25
U4	6:25	3:11	3:15	5:04	6:31	0:43	25:09
U5	14:33	1:39	4:29	3:59	4:43	0:54	30:17
U6	7:17	1:56	3:54	1:34	3:52	0:56	19:02
Average	11:34	2:34	4:30	4:32	4:46	1:30	29:10

Table 8.4.: Time on Task for different users (in Minutes: Seconds)

8.8.3. SUS

The SUS score for the different participants (see Table 8.5) averaged 74.16, which equals a good result or grade B. However, the wide variation between participants (40 to 92.5) indicates differing perceptions of usability. While some participants rated the system as highly usable, others experienced notable challenges. The low SUS score of User3 again points to a gap for users without a technical background. In contrast, the high scores from several others confirm that the system can deliver an excellent user experience under the right conditions.

User	S01	S03	S05	S07	S09	S02	S04	S06	S08	S10	Result
U1	5	4	5	5	4	1	1	1	2	1	92.5
U2	4	3	3	2	3	3	1	4	3	1	57.5
U3	2	4	4	2	1	2	5	1	4	5	40
U4	5	4	4	4	3	2	2	1	1	1	82.5
U5	3	3	5	4	4	1	1	1	2	1	82.5
U6	4	4	5	4	5	2	1	1	1	1	90
Avg	3.83	3.66	4.33	3.5	3.33	1.83	1.83	1.5	2.16	1.66	74.16

Table 8.5.: SUS score

8.9. Qualitative Results

This section summarizes findings from recorded user interviews, including reported bugs, uncertainties, feature requests, and positive feedback. In addition, it highlights issues that have been addressed and resolved after conducting the usability study.

8.9.1. Bugs

During the usability test, one participant experienced issues with the responsiveness of the tool, indicating that the user interface did not adapt optimally to their screen size or device. Additionally, a bug was observed where one user encountered two vertical sliders displayed simultaneously in the Model Table component, instead of the intended single slider. Both bugs occurred on smaller devices with a screen size of 13 inches.

8.9.2. Uncertainties

Several usability uncertainties were identified during the evaluation. Participants expected the sidebar to collapse automatically when clicking outside of it, indicating that this behavior aligns more closely with common user interface conventions. Additionally, users suggested maintaining a consistent order of diagrams between the Bar Chart Matrix component and the Line Chart Matrix component to facilitate easier comparison and reduce cognitive load. From usability experts, there was also uncertainty regarding how different metrics are ranked in terms of clustering performance and what specific values indicate promising results. This lack of clarity made it more difficult for users to filter out models efficiently. Furthermore, some users found the labels on the buttons confusing. In particular, it was suggested to rename the "Show Outliers" button to "Show Misclusterings". Finally, users expected some form of feedback when clicking the "Save" button in the image data settings, such as a visual indicator that the settings had been successfully saved.

8.9.3. Feature Request

The participants suggested several improvements to enhance the functionality and user experience of the dashboard, thereby requesting multiple new features. One request was to include an overall row and column in the confusion matrix to display total values for each class, which would provide a more comprehensive overview of classification performance and the data set used. Currently, it is hard for users to identify if there is any skewing in the dataset, such as having one class overrepresented. An "Overall" column in the confusion matrix would solve this problem. It was also noted that the confusion matrix should be able to be turned on and off, allowing users to toggle its visibility, so that they are not distracted if they are not completing any tasks that require the confusion matrix. To improve data exploration, users recommended that each plot include a "Focus" button. This feature would allow individual visualizations to be expanded to full-screen view in a pop-up, while graying out the background, enabling more detailed inspection without distractions from other components. Another feature request relates to the dimensionality reduction techniques and labeling in the scatter plot components. Since it can be changed in each scatter plot individually, there should be an option to also change the dimensionality reduction method and the labels globally through buttons. For the loss visualization, users expressed interest in additional interactivity. Specifically, they requested that hovering over the line should display the minimum loss value along with the corresponding epoch, where this value occurs, making it easier to identify the minimum immediately. When uploading, including or excluding a model, users expect some form of response or visual feedback. A progress bar showing the remaining time or indicating which model is currently being processed would help to manage time expectations and provide better system transparency. Lastly, it was suggested that hovering over a score for an extended period should trigger a brief explanation of the metric's meaning, aiding interpretability and supporting less experienced users in understanding the metrics.

8.9.4. Usability Difficulties

Several usability difficulties were identified during the study. One common issue was the zoom functionality, which was triggered unintentionally during regular scrolling. Participants suggested that zooming should only occur when holding the "Shift" key in combination with the scroll action, to prevent accidental zoom and improve navigation control. Another concern was the visual layout of the dashboard. Users requested more spacing between the "columns" to reduce visual clutter and improve readability. In addition, it was noted that model rows should include the filename as a heading to indicate which model each row refers to. Upload times also posed a challenge and were highly dependent on the device used in the study, resulting in users with better devices completing task one in half the time than users with worse devices. Participants felt that the duration of uploading model files was sometimes unclear or too long without sufficient feedback, which led to uncertainty about the system's status. Lastly, users indicated that some regions of the interface could benefit from clearer grouping. They recommended the use of visible boxes or borders around related elements, such as tables, metrics, and model

details, to better signal their affiliation and improve the overall structure and orientation within the interface.

8.9.5. Positive Feedback & Observations

Participants found the upload and settings section to be informative, noting that the provided context was helpful and that model upload and image data options were easy to locate. The menu options of the dashboard were perceived as well-chosen and logically organized. In the scatter plots, using the same color for different models made them easy to locate across all dashboard components. The implementation of the confusion matrix was described as straightforward and intuitive. Participants also highlighted the tool's ability to provide very detailed insights into the clustering process. They reported that the dashboard enabled them to identify models and algorithms that achieved good clustering performance after relatively few training epochs. This insight was viewed as particularly beneficial for optimizing training time and reducing computational resource usage. In connection with this, users could leverage the confusion matrix to better understand class-specific performance, identifying which classes were consistently well classified and which contributed to reduced model accuracy. In several cases, participants gained a new understanding of the data itself, including which classes had higher or lower accuracy and how these patterns affected the model's overall performance. For example, it was observed that in one model (see M3 in Table 8.1) one class (class 9) was frequently misclassified as another (class 3), leading to reduced accuracy, whereas another model with a higher accuracy had more misclusterings over all classes (see M1 in Table 8.1). These observations enhanced users' ability to interpret model and algorithm behavior beyond accuracy scores, since they can see that the model with a lower accuracy is better at predicting all classes except one (class 9). Furthermore, the visualization of the latent space over time gave participants insight into how data points evolved during training. In models with lower accuracy, data clusters gradually emerged from an initially unstructured cloud, whereas in some higher-accuracy models, the latent space retained a more dispersed appearance. This contrast helped users better understand the internal dynamics of model learning and reinforced the idea that accuracy alone is not always sufficient for evaluating performance. The misclustering feature was particularly valuable for identifying and interpreting specific failures. Participants were able to analyze individual misclustered data points and determine whether these errors were due to poor data quality or limitations within the algorithm. This functionality supported a more diagnostic and interpretive use of the tool, supporting the model training by letting users exclude, for example, wrongly labeled or poor-quality data points. Finally, the inclusion of histogram-based filters enabled users to identify underperforming models across multiple evaluation metrics. This helped guide further optimization by revealing which algorithms or hyperparameter configurations consistently failed to produce good results, thereby allowing users to make informed decisions about which settings were unsuitable for the given dataset.

8.9.6. Resolved Issues after Usability Study

Uncertainties, feature requests, and usability difficulties were identified in the usability study and rectified thereafter. Resolved issues can be found here (see Table 8.6), which led to the final dashboard in this thesis (see chapter 6). Open issues will be addressed in the future.

Issue	**Resolved?**
Sidebar collapses by clicking outside it	Yes
Consistent order of diagrams between Bar Chart Matrix and Line Chart Matrix	Yes
Indicating how different metrics are ranked among each other	No
Indicating what values for metrics indicate good results	No
Labels on buttons confusing	Yes
Feedback after clicking the "Save" button for image data	No
Overall row and column in confusion matrix	No
Toggle confusion matrix	Yes
"Focus" button to expand individual visualizations to full screen	Yes
Global buttons to change the dimensionality reduction technique/labels in scatter plots	No
Hovering should display the minimum loss along the corresponding epoch	Yes
Visual feedback model upload	Yes
Visual feedback including/excluding a model	No
Unintentionally trigger zoom during regular scrolling	Yes
More spacing between the "columns"	Yes
Clearer grouping with boxes or borders	Yes
Shorter upload times	No
Overall	**10/17**

Table 8.6.: Resolved Issues after the Usability Study

9. Discussion

This chapter reflects on the feedback of the domain experts and users as well as the project as a whole. The first part deals with encountered pitfalls and lessons learned during the project. Furthermore, this chapter discusses potential future enhancements for Clustorama. Lastly, it provides a conclusion and summarizes the key takeaways from this project.

9.1. Pitfalls

During and after the development of the dashboard, several challenges and limitations became apparent.

One of the early pitfalls was handling inconsistent or incomplete data during logging. The functionality of algorithms for the logging process must be understood to create a suitable logging class. As a visual analysis expert, understanding specific steps in the clustering process, like the reordering of data points in batches, posed a challenge and was time-costly. This led to incomplete and inconsistent log files, not suited for data visualization.

As more features were added and log files became bigger, particularly for visualizing large datasets or multiple models simultaneously, the frontend performance became a bottleneck. Especially, the upload of models, which was intended for tens of models, became suitable for only around ten models, depending on the user's hardware.

Designing intuitive and meaningful interaction mechanisms for filtering, tooltips, zooming, and linking is less intuitive than initially anticipated. Some design decisions had to be iteratively refined based on user feedback or observed usability issues. Other design decisions had to be discarded after investing a significant amount of time and iterations into them.

Another pitfall is that the topic of deep clustering is too large. It was not possible to integrate the logging mechanism into all deep clustering algorithms that are implemented in ClustPy. Fixating on a few deep clustering algorithms, which were benchmarked with known data sets, was the priority. Next steps include adding the logging mechanism to more and more of ClustPy's deep clustering algorithms in the future.

A further pitfall was the bias towards visualization techniques based on the tools the domain experts had in use. The domain experts have become accustomed to visualizations from tools already in use and are therefore difficult to convince of alternative visualization proposals.

9.2. Future Work

The feedback from the usability study showed great potential for the tool to be used, as well as some open issues that can further improve the user experience with the tool. These insights motivate future work on the tool, discussed in this section.

Adding interpretability, in particular for the metrics, would increase the dashboard's value for model assessment. Explanations of different metrics, as well as a ranking of their importance to the clustering process, should be present in the dashboard. Furthermore, it would make the tool usable for a broader audience, like beginners in the clustering and deep clustering domains.

Integrating new functionalities requested by the users would further enhance the dashboard's reception, especially the option to modify settings for components globally, rather than just individually. In particular, the settings for dimensionality reduction and labels for the scatter plots and the scatter plot matrix should be global options.

Adding more visual feedback, especially for button clicks and model inclusion, as well as exclusions, would also increase the usability of the tool.

Supporting the upload of larger and more model log files in a reasonable time would significantly improve usability, since the model upload takes the most time in the user's workflow.

Conducting further iterations of usability studies after additional suggestions are implemented would help to refine the design and better align the dashboard with real-world workflows and tasks.

Next steps also include coordinating with the ClustPy owners to commit the Logger Class into the ClustPy tool, along with instructions for users to create log files themselves. It also includes developing a plan to integrate the dashboard into ClustPy, either with references to the repository or a direct integration into ClustPy's repository. Furthermore, the Logger Class must be made available to an increasing number of deep clustering algorithms within ClustPy.

After the integration process, another round of long-term studies will be conducted, focusing on user reception of the tool. This includes a study on the number of downloads, new feature requests, and upcoming issues that come up after the publication.

9.3. Conclusion

This work studied visualization to support and facilitate the interpretation of deep clustering algorithms. More specifically, a visualization dashboard was developed, supporting domain experts of the Data Mining and Machine Learning group of the University of Vienna. For the development, an in-depth requirement analysis was created in collaboration with the domain experts of the research group. Interviews revealed ten tasks that need to be performed by the domain experts. Some of the tasks could already be fulfilled to some extent by dashboards identified in the related work for standard clustering algorithms. However, essential tasks were not yet feasible for deep clustering algorithms, for example, to gain further insights into the embedding during the training process. Other tasks,

like the investigation of loss development throughout the training, were already solvable, but not in combination with different tasks like misclustering detection. This forced the domain experts to use multiple non-tailored tools or have blind spots in their analysis of deep clustering algorithms.

The dashboard created in this work supports all tasks identified during the requirements analysis. It allows the domain experts to visualize the results of various deep clustering algorithms with different hyperparameter settings for different benchmark data sets. Thereby, it lets the user track the embeddings over the training epochs and compare the embedding development with the loss development, by linking different views. Users can switch between different dimensionality reduction methods to visualize their data, as well as different labels applied to their embedding view. Furthermore, their decisions on clustering quality are supported by different metrics, which also enable evaluation of quality in a higher-dimensional latent space. Lastly, misclustered data points can be identified, visualized, and mapped back to the original data, allowing users to also investigate the quality of the dataset.

All this supports the users in answering the research questions stated in chapter 1:

- RQ_{latent} How can we make the latent space more interpretable to gain further information for training and optimization?

The dashboard improves latent space interpretability by providing interactive visualizations such as the training scatter plot and scatter plot matrix. These allow users to observe embedding evolution over epochs, relate it to loss reduction, and inspect cluster separation in detail. Misclustering detection and point-level tracking offer insights into unstable or shifting embeddings, enabling a deeper understanding of training dynamics and optimization potential. Scenarios 3 and 4 demonstrated how these capabilities help identify structural patterns, relationships between clusters, and problematic data points.

- $RQ_{cluster}$ How can the deep clustering process be supported to increase the understanding and make it more efficient?

The dashboard supports the deep clustering process by enabling efficient comparison of multiple algorithms and hyperparameter configurations, as described in Scenario 1 and Scenario 2. Filtering, histograms, and tables allow users to focus on promising candidates. Linked visualizations, like scatter plots, confusion matrices, and line charts, facilitate the detection of optimal algorithm and hyperparameter combinations. These tools also reveal the minimum number of epochs needed for meaningful cluster separation, supporting resource-efficient training through early stopping.

The evaluation of the dashboard shows great potential, but room to improve, which is normal, since the dashboard is still a work in progress. More guidance, especially for non-domain experts, would make the tool more accessible to beginners and support them in understanding the general principles of deep clustering algorithms during the training. Optimization and incorporation of features, bugs, and uncertainties is the next step to improve the dashboard's quality further. Finally, an evaluation after the publication of the tool needs to be conducted to gain further suggestions for improvement, as well as insight into how users generally accept the tool.

Bibliography

[Abd07] Hervé Abdi. Metric multidimensional scaling (mds): analyzing distance matrices. *Encyclopedia of measurement and statistics*, pages 1–13, 2007.

[AC15] Manuela Aparicio and Carlos J Costa. Data visualization. *Communication design quarterly review*, 3(1):7–11, 2015.

[AK98] E. Alpaydin and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[Ans73] Francis J Anscombe. Graphs in statistical analysis. *The american statistician*, 27(1):17–21, 1973.

[ANS13] Jyoti Agarwal, Renuka Nagpal, and Rajni Sehgal. Crime analysis using k-means clustering. *International Journal of Computer Applications*, 83(4), 2013.

[ANSS07] Morten Sieker Andreasen, Henrik Villemann Nielsen, Simon Ormholt Schrøder, and Jan Stage. What happened to remote usability testing? an empirical study of three methods. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1405–1414, 2007.

[AR13] Charu Aggarwal and Chandan Reddy. *DATA CLUSTERING Algorithms and Applications*. 08 2013.

[Asi85] Daniel Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

[ASJ+21] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[AWC+07] Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. In *Artificial Intelligence and Statistics*, pages 11–18. PMLR, 2007.

[B+96] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[Bai12] Stephen Bailey. Principal component analysis with noisy and/or missing data. *Publications of the Astronomical Society of the Pacific*, 124(919):1015, 2012.

[BB19] Benjamin Bengfort and Rebecca Bilbro. Yellowbrick: Visualizing the scikit-learn model selection process. *Journal of Open Source Software*, 4(35):1075, 2019.

[BB23] Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[BCH+22] Alex Bäuerle, Ángel Alexander Cabrera, Fred Hohman, Megan Maher, David Koski, Xavier Suau, Titus Barik, and Dominik Moritz. Symphony: Composing interactive interfaces for machine learning. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2022.

[Bis06] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128, 2006.

[BKM08] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BRF+22] Francesco Bodria, Salvatore Rinzivillo, Daniele Fadda, Riccardo Guidotti, Fosca Giannotti, and Dino Pedreschi. Explaining black box with visual exploration of latent space. In *EuroVis (Short Papers)*, pages 85–89, 2022.

[BWM+24] Anna Beer, Pascal Weber, Lukas Miklautz, Collin Leiber, Walid Durani, Christian Böhm, and Claudia Plant. Shade: Deep density-based clustering, 2024.

[Car09] Stuart Card. Information visualization. In *Human-Computer Interaction*, pages 199–234. CRC press, 2009.

[CBJD19] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019.

[CD18] Marco Cavallo and Çağatay Demiralp. Clustrophile 2: Guided visual clustering analysis. *IEEE transactions on visualization and computer graphics*, 25(1):267–276, 2018.

[clu] Clustpy: A python library for advanced clustering algorithms. `https://github.com/collinleiber/ClustPy`. Accessed: 2024-11-18.

[CNL11] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[CSHB18] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[DDN+23] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

[Dem17] Çağatay Demiralp. Clustrophile: A tool for visual clustering analysis. *arXiv preprint arXiv:1710.02173*, 2017.

[DF09] Joseph Dumas and Jean Fox. *Usability testing: Current practice and future directions*, pages 1129–1149. 01 2009.

[EDF08] Niklas Elmqvist, Pierre Dragicevic, and Jean-Daniel Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, 2008.

[Fod02] Imola K Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2002.

[GDZ+23] Caroline X Gao, Dominic Dwyer, Ye Zhu, Catherine L Smith, Lan Du, Kate M Filia, Johanna Bayer, Jana M Menssink, Teresa Wang, Christoph Bergmeir, et al. An overview of clustering methods with guidelines for application in mental health research. *Psychiatry Research*, 327:115265, 2023.

[GGLY17] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Ijcai*, volume 17, pages 1753–1759, 2017.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[HPRC19] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Polo Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.

[HTF17] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction, 2017.

[HW13] Julian Heinrich and Daniel Weiskopf. State of the art of parallel coordinates. *Eurographics (state of the art reports)*, pages 95–116, 2013.

[HZZL02] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl_1):S145–S154, 2002.

[IMI+10] Stephen Ingram, Tamara Munzner, Veronika Irvine, Melanie Tory, Steven Bergner, and Torsten Möller. Dimstiller: Workflows for dimensional analysis and reduction. In *2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 3–10, 2010.

[Jam13] Gareth James. An introduction to statistical learning, 2013.

[KAKC17] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.

[KBZ+21] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, and Stefan Decker. Deep learning-based clustering approaches for bioinformatics. *Briefings in bioinformatics*, 22(1):393–415, 2021.

[KER+24] Jacob Kauffmann, Malte Esders, Lukas Ruff, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. From clustering to cluster explanations via neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):1926–1940, February 2024.

[KEV+17] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE transactions on visualization and computer graphics*, 24(1):142–151, 2017.

[KEV+18] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F. Stewart, and Adam Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):142–151, 2018.

[KR23] Siva Rajesh Kasa and Vaibhav Rajan. Avoiding inferior clusterings with misspecified gaussian mixture models. *Scientific Reports*, 13(1):19164, 2023.

[Laz05] Jonathan Lazar. *Web usability: A user-centered design approach.* Addison-Wesley Longman Publishing Co., Inc., 2005.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[Lew06] James R. Lewis. Sample sizes for usability tests: mostly math, not magic. *Interactions*, 13(6):29–33, 2006.

[LFH17] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. Chapter 10 - usability testing. In Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser, editors, *Research Methods in Human Computer Interaction (Second Edition)*, pages 263–298. Morgan Kaufmann, Boston, second edition edition, 2017.

[LMPB23] Collin Leiber, Lukas Miklautz, Claudia Plant, and Christian Böhm. Benchmarking deep clustering algorithms with clustpy. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 625–632. IEEE, 2023.

[LS18] James R Lewis and Jeff Sauro. Item benchmarks for the system usability scale. *Journal of Usability studies*, 13(3), 2018.

[LYRL04] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[LZS20] Mingwei Li, Zhenge Zhao, and Carlos Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 5(3):e25, 2020.

[MPB20] Dominik Mautz, Claudia Plant, and Christian Böhm. Deepect: The deep embedded cluster tree. *Data Science and Engineering*, 5:419–432, 2020.

[Mun14] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[MV15] Tauno Metsalu and Jaak Vilo. Clustvis: a web tool for visualizing clustering of multivariate data using principal component analysis and heatmap. *Nucleic acids research*, 43(W1):W566–W570, 2015.

[NN19] Olfa Nasraoui and C-E Ben N'Cir. Clustering methods for big data analytics. *Techniques, Toolboxes and Applications*, 1:91–113, 2019.

[PB08] Saurabh Prasad and Lori Mann Bruce. Limitations of principal components analysis for hyperspectral target recognition. *IEEE Geoscience and Remote Sensing Letters*, 5(4):625–629, 2008.

[PHVG+18] Nicola Pezzotti, Thomas Höllt, Jan Van Gemert, Boudewijn P.F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, 2018.

[RC08] Jeffrey Rubin and Dana Chisnell. *Handbook of usability testing: How to plan, design, and conduct effective tests*. John Wiley & Sons, 2008.

[RG] Juan R. Terven Diana M. Cordova-Esparza Alfonso Ramirez-Pedraza Edgar A. Chavez-Urbiola Julio A. Romero-Gonzalez. Loss functions and metrics in deep learning. `https://arxiv.org/pdf/2307.02694`. Accessed: 2024-11-22.

[RLJF22] Octavio Ramos-Leaños, Jneid Jneid, and Bruno Fazio. Non-linear clustering of distribution feeders. *Energies*, 15(21):7883, 2022.

[RPY+25] Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S. Yu, and Lifang He. Deep clustering: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):5858–5878, 2025.

[RSVR18] Liam J Revell, Klaus Schliep, Eugenio Valderrama, and James E Richardson. Graphs in phylogenetic comparative analysis: Anscombe's quartet revisited. *Methods in Ecology and Evolution*, 9(10):2145–2154, 2018.

[SDV+16] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[Shl14] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[Shn03] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.

[SLH+13] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 18th Iberoamerican Congress, CIARP 2013, Havana, Cuba, November 20-23, 2013, Proceedings, Part I 18*, pages 117–124. Springer, 2013.

[SMM12] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics*, 18(12):2431–2440, 2012.

[SNHMS18] Nasir Saeed, Haewoon Nam, Mian Imtiaz Ul Haq, and Dost Bhatti Muhammad Saqib. A survey on multidimensional scaling. *ACM Computing Surveys (CSUR)*, 51(3):1–25, 2018.

[SSJR22] Parvathaneni Naga Srinivasu, N Sandhya, Rutvij H Jhaveri, and Roshani Raut. From blackbox to explainable ai in healthcare: existing tools and case studies. *Mobile Information Systems*, 2022(1):8167821, 2022.

[SVM14] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.

[ten] Tensorboard: Tensorflow's visualization toolkit. `https://www.tensorflow.org/tensorboard`. Accessed: 2025-05-14.

[tfg] Tensorflow graph. `https://www.tensorflow.org/tensorboard/graphs`. Accessed: 2025-08-03.

[tfm] Tensorflow model analysis. `https://www.tensorflow.org/tfx/guide/tfma`. Accessed: 2025-05-14.

[tfp] A neural network playground - tensorflow. `https://playground.tensorflow.org`. Accessed: 2025-08-03.

[TKC18] Elad Tzoreff, Olga Kogan, and Yoni Choukroun. Deep discriminative latent space for clustering. *arXiv preprint arXiv:1805.10795*, 2018.

[Tor52] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[TZG17] Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part II 17*, pages 809–825. Springer, 2017.

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

[Vir92] Robert A Virzi. Refining the test phase of usability evaluation: how many subjects is enough? *Human factors*, 34(4):457–468, 1992.

[wan] Weigths biases : The ai developer platform. `https://wandb.ai/site`. Accessed: 2024-11-21.

[Was21] Michael L Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[WFW21] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern recognition letters*, 141:61–67, 2021.

[WYZ+23] Lirong Wu, Lifan Yuan, Guojiang Zhao, Haitao Lin, and Stan Z. Li. Deep clustering and visualization for end-to-end high-dimensional data analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8543–8554, 2023.

[XGF16] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[XUD+19] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural language processing and Chinese computing: 8th cCF international conference, NLPCC 2019, dunhuang, China, October 9–14, 2019, proceedings, part II 8*, pages 563–574. Springer, 2019.

[YML+15] Hamid Younesy, Torsten Möller, Matthew C Lorincz, Mohammad M Karimi, and Steven JM Jones. Visrseq: R-based visual framework for analysis of sequencing data. *BMC bioinformatics*, 16:1–14, 2015.

[YPMK23a] Hamid Younesy, Joseph Pober, Torsten Möller, and Mohammad M Karimi. Modex: a general purpose computer model exploration system. *Frontiers in Bioinformatics*, 3:1153800, 2023.

[YPMK23b] Hamid Younesy, Joseph Pober, Torsten Möller, and Mohammad Karimi. Modex: a general purpose computer model exploration system. *Frontiers in Bioinformatics*, 3, 05 2023.

[YS18] Rulei Yu and Lei Shi. A user-based taxonomy for deep learning visualization. *Visual Informatics*, 2(3):147–154, 2018.

[ZD21] Hongjing Zhang and Ian Davidson. Deep descriptive clustering, 2021.

[ZHS+23] Zhongliang Zhou, Mengxuan Hu, Mariah Salcedo, Nathan Gravel, Wayland Yeung, Aarya Venkat, Dongliang Guo, Jielu Zhang, Natarajan Kannan, and Sheng Li. Xai meets biology: A comprehensive review of explainable ai in bioinformatics applications. *arXiv preprint arXiv:2312.06082*, 2023.

A. Appendix

```
class TrainingLogger:
    def __init__(self, log_file, data, data_labels, n_clusters, scaler=MinMaxScaler()):
        self.log_file = os.path.join(log_dir, log_file)
        self.all_epoch_data = []
        self.data = data
        self.labels = data_labels
        self.n_clusters = n_clusters
        self.scaler = scaler
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    def log_first_epoch(self):
        print("LOGGING RAW EMBEDDING")
        labels = self.labels
        X = self.data

        scaler_epoch1 = MinMaxScaler()
        normalized_embedded_epoch1 = torch.tensor(scaler_epoch1.fit_transform(X), device=self.device)

        normalized_embedded_cpu_epoch1 = normalized_embedded_epoch1.cpu().numpy().tolist()
        estimatorKMeans_epoch1 = KMeans(n_clusters=self.n_clusters)
        estimatorKMeans_epoch1.fit(normalized_embedded_epoch1.cpu())
        labelsKMeans_epoch1 = estimatorKMeans_epoch1.labels_
        labelsKMeans_cpu_epoch1 = labelsKMeans_epoch1.tolist()

        pca_epoch1 = PCA(n_components=2)
        pca_embedding_not_normalized_epoch1 = pca_epoch1.fit_transform(normalized_embedded_epoch1.tolist())
        pca_embedding_epoch1 = scaler_epoch1.fit_transform(pca_embedding_not_normalized_epoch1)

        tsne_epoch1 = TSNE(n_components=2, learning_rate='auto', init='random', perplexity=3).fit_transform(np.array(normalized_embedded_epoch1.tolist()))
        tsne_embedding_epoch1 = scaler_epoch1.fit_transform(tsne_epoch1)

        mds_epoch1 = MDS(n_components=2, normalized_stress='auto')
        mds_embedding_not_normalized_epoch1 = mds_epoch1.fit_transform(normalized_embedded_epoch1.tolist())
        mds_embedding_epoch1 = scaler_epoch1.fit_transform(mds_embedding_not_normalized_epoch1)

```

```
36          lda_epoch1 = LinearDiscriminantAnalysis(n_components=2,
    random_state=0)
37          lda_embedding_not_normalized_epoch1 = lda_epoch1.fit_transform(
    normalized_embedded_epoch1.tolist())
38          lda_embedding_epoch1 = scaler_epoch1.fit_transform(
    lda_embedding_not_normalized_epoch1)
39
40          labels_cpu_epoch1 = labels.tolist()
41          accuracy_epoch1 = accuracy_score(labels_cpu_epoch1,
    labelsKMeans_cpu_epoch1)
42          ami_score_epoch1 = adjusted_mutual_info_score(labels_cpu_epoch1,
    labelsKMeans_cpu_epoch1)
43          ari_score_epoch1 = adjusted_rand_score(labels_cpu_epoch1,
    labelsKMeans_cpu_epoch1)
44          ch_score_epoch1 = calinski_harabasz_score(
    normalized_embedded_cpu_epoch1, labelsKMeans_cpu_epoch1)
45          sil_score_epoch1 = silhouette_score(
    normalized_embedded_cpu_epoch1, labelsKMeans_cpu_epoch1)
46
47          epoch_data_epoch1 = {
48                  'epoch': 0,
49                  'pca_embedding': pca_embedding_epoch1.tolist(),
50                  'tsne_embedding': tsne_embedding_epoch1.tolist(),
51                  'mds_embedding': mds_embedding_epoch1.tolist(),
52                  'lda_embedding': lda_embedding_epoch1.tolist(),
53                  'losses': [0],
54                  'loss_names': ["N/A"],
55                  'kmeans_labels': labelsKMeans_cpu_epoch1,
56                  'metrics_names': ["Accuracy", "Adjusted Mutual
    Information", "Rand index", "Calinski Harabasz Score", "Silhouette
    Coefficient", "k", "Average Gradient", "Maximum Gradient"],
57                  'clustering_metrics': [accuracy_epoch1, ami_score_epoch1,
     ari_score_epoch1, ch_score_epoch1, sil_score_epoch1, self.n_clusters,
     0, 0],
58              }
59          self.all_epoch_data.append(epoch_data_epoch1)
60
61      def log_epoch(self, epoch, total_loss, complete_embedding):
62          print("LOGGING EPOCH : ", epoch + 1)
63          labels = self.labels.tolist()
64          embedding_size = self.n_clusters
65
66          scaler = MinMaxScaler()
67          normalized_embedded = torch.tensor(
68              scaler.fit_transform(complete_embedding.cpu().numpy()),
    device=self.device
69          )
70          normalized_embedded_cpu = normalized_embedded.cpu().numpy().
    tolist()
71
72          estimatorKMeans = KMeans(n_clusters=self.n_clusters, n_init=10,
    random_state=0)
73          estimatorKMeans.fit(normalized_embedded.cpu())
```

```
74          labelsKMeans = estimatorKMeans.labels_
75          labelsKMeans_cpu = self.reorder_colors(y_pred=labelsKMeans.tolist
    (),y_true=labels)
76
77          pca = PCA(n_components=2)
78          pca_embedding = scaler.fit_transform(pca.fit_transform(
    normalized_embedded.tolist()))
79
80          tsne = TSNE(n_components=2, learning_rate='auto', init='random',
    perplexity=3)
81          tsne_embedding = scaler.fit_transform(tsne.fit_transform(np.array
    (normalized_embedded.tolist())))
82
83          mds = MDS(n_components=2, normalized_stress='auto')
84          mds_embedding = scaler.fit_transform(mds.fit_transform(
    normalized_embedded.tolist()))
85
86          lda = LinearDiscriminantAnalysis(n_components=2, random_state=0)
87          lda_embedding = scaler.fit_transform(lda.fit_transform(
    normalized_embedded.tolist()))
88
89          labels_cpu = labels
90          accuracy = accuracy_score(labels_cpu, labelsKMeans_cpu)
91          ami_score = adjusted_mutual_info_score(labels_cpu,
    labelsKMeans_cpu)
92          ari_score = adjusted_rand_score(labels_cpu, labelsKMeans_cpu)
93          ch_score = calinski_harabasz_score(normalized_embedded_cpu,
    labelsKMeans_cpu)
94          sil_score = silhouette_score(normalized_embedded_cpu,
    labelsKMeans_cpu)
95
96          epoch_data = {
97              'epoch': epoch + 1,
98              'pca_embedding': pca_embedding.tolist(),
99              'tsne_embedding': tsne_embedding.tolist(),
100             'mds_embedding': mds_embedding.tolist(),
101             'lda_embedding': lda_embedding.tolist(),
102             'losses': [total_loss],
103             'loss_names': ["DEC Loss"],
104             'kmeans_labels': labelsKMeans_cpu,
105             'metrics_names': ["Accuracy", "Adjusted Mutual Information",
    "Rand index",
106                               "Calinski Harabasz Score", "Silhouette
    Coefficient", "k"],
107             'clustering_metrics': [accuracy, ami_score, ari_score,
    ch_score, sil_score, self.n_clusters]
108         }
109
110         self.all_epoch_data.append(epoch_data)
111
112
113     def log_labels(self, kmeans_labels, dec_labels):
114         print("LOGGING LABELS")
```

```
115         label_counts = dict(Counter(self.labels.tolist()))
116         cm = confusion_matrix(self.labels.tolist(), kmeans_labels.tolist
    ())
117 
118         labels_data = {
119             'labels': self.labels.tolist(),
120             'labels_counter': label_counts,
121             'confusion_matrix': cm,
122             'dec_labels': dec_labels.tolist(),
123         }
124 
125         self.all_epoch_data.append(labels_data)
126 
127 
128     def log_model_data(self,clustering_epochs,batch_size,loss_fn,
    optimizer):
129         print("LOGGING MODEL DATA")
130         model_data = {
131                 'n_epochs': clustering_epochs,
132                 'batch_size': batch_size,
133                 'loss_fn': loss_fn,
134                 'optimizer_class': optimizer,
135                 'scheduler': scheduler.__class__.__name__ if scheduler
    else None,
136                 'learning_rate': self.clustering_optimizer_params
137             }
138 
139         self.all_epoch_data.append(model_data)
140 
141 
142 
143     def convert_numpy_types(obj):
144         if isinstance(obj, np.ndarray):
145             return obj.tolist()
146         elif isinstance(obj, (np.int64, np.int32)):
147             return int(obj)
148         elif isinstance(obj, (np.float64, np.float32)):
149             return float(obj)
150         elif isinstance(obj, dict):
151             return {convert_numpy_types(k): convert_numpy_types(v) for k,
     v in obj.items()}
152         elif isinstance(obj, (list, tuple)):
153             return [convert_numpy_types(item) for item in obj]
154         else:
155             return obj
156 
157     def reorder_colors(self, y_pred, y_true):
158         cm = confusion_matrix(y_true, y_pred)
159         row_ind, col_ind = linear_sum_assignment(-cm )
160         y_pred_permuted = np.argsort(col_ind)[y_pred]
161         return y_pred_permuted
162 
163     def _save_log(self):
```

```
164         all_epoch_data_temp = convert_numpy_types(self.all_epoch_data)
165         with open(self.log_file, 'w') as f:
166             json.dump({'clustering': all_epoch_data_temp}, f, indent=4)
167         print("LOGGING COMPLETE")
```

Listing A.1: Logger Class