TU WIEN Informatics

Interaktives Sampling für Klassenerkennung in Unstrukturierten Daten

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Lukas Fitz, BSc
Matrikelnummer 11815353

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dr.in techn. Manuela Waldner, MSc
Mitwirkung: Assistant Prof. Dr.techn. Stefan Neumann, BSc, MSc

Wien, 12. Mai 2026

Lukas Fitz	Manuela Waldner

TU WIEN Informatics

Interactive Sampling For Class Discovery In Unstructured Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Lukas Fitz, BSc
Registration Number 11815353

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr.in techn. Manuela Waldner, MSc
Assistance: Assistant Prof. Dr.techn. Stefan Neumann, BSc, MSc

Vienna, May 12, 2026

Lukas Fitz	Manuela Waldner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Lukas Fitz, BSc

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

Wien, 12. Mai 2026

Lukas Fitz

Danksagung

Ich möchte meiner Betreuerin Manuela Waldner danken für ihre unermüdliche Unterstützung während dieser Arbeit. Ihre Unterstützung war für mich äußerst wertvoll, sei es durch ihr Fachwissen, ihre Sorgfalt oder die kontinuierliche Begleitung durch regelmäßige Gespräche. Das hat mir beim Verstehen und Schreiben geholfen, sowie dabei, den Fokus nicht zu verlieren.

Weiters möchte ich auch meinem Co-Supervisor Stefan Neumann danken für sein wertvolles Feedback und hilfreiche Anregungen, die mir geholfen haben, weitere, mir im Vorhinein nicht bekannte Methoden zu finden. Dies hatte einen wesentlichen Einfluss auf diese Arbeit.

Auch möchte ich mich bei meinen Freundinnen und Freunden bedanken, die mich während meines Studiums und darüber hinaus immer unterstützt und aufgebaut haben, falls es mal nicht so lief wie erwartet. Im Speziellen möchte ich dabei meine Verlobte Lisa hervorheben, die eine wichtige Stütze ist und mir immer volle Unterstützung zusichert.

Meinen Eltern möchte ich auch für die kontinuierliche Unterstützung danken und dafür, dass sie mir immer den Rücken gestärkt haben, damit ich jetzt an diesem Punkt angelangt bin.

Acknowledgements

I would like to thank my advisor, Manuela Waldner, for her tireless support throughout this project. Her assistance was incredibly valuable to me, whether in terms of her expertise, her attention to detail, or her ongoing guidance through regular meetings. This helped me understand the subject matter and write this thesis, as well as stay focused.

I would also like to thank my co-supervisor, Stefan Neumann, for his valuable feedback and helpful suggestions, which helped me discover additional methods I had not considered beforehand. This had a significant impact on this thesis.

I would also like to thank my friends, who have always supported and encouraged me throughout my studies and beyond, especially when things did not go as expected. In particular, I would like to highlight my fiancée Lisa, who is a pillar of strength and always assures me of her full support.

I would also like to express my gratitude to my parents for their continuous support and for always having my back, which has enabled me to reach this point today.

Kurzfassung

Die Analyse unstrukturierter Daten wie Bilder stellt aufgrund der hohen Dimensionalität eine erhebliche Herausforderung für die explorative Datenanalyse dar, da die Daten zunächst in Embeddings überführt werden müssen und in niedriger Dimensionalität dann näher beieinander liegen. Sampling ist unverzichtbar, um diese Datensätze in Visualisierungen für Menschen verständlicher zu machen. Die Motivation dieser Arbeit besteht darin, wie interaktive Systeme auch bei großen und unbalancierten Bilddatensätzen repräsentative und interpretierbare Stichproben in kurzer Zeit bereitstellen können. Standardmethoden wie Random Sampling können dabei an ihre Grenzen stoßen und erfassen seltene Klassen häufig nicht, was zu Verzerrungen in der Interpretation führt. Innerhalb eines standardisierten Interaktionsprotokolls werden verschiedene datengetriebene Strategien (z. B. Farthest Sampling und D^{α} Sampling) und modellbasierte Strategien (z. B. Min-Margin und Disagreement) mit Random Sampling verglichen. Ziel ist es, zu untersuchen, welche Strategien innerhalb eines festgelegten Interaktionsbudgets das beste Gleichgewicht zwischen schneller Klassenerkennung, hoher Modellgenauigkeit und geringer Latenz bieten. Die Datenpunkte werden durch das Modell klassifiziert, deshalb handelt es sich um ein simuliertes Experiment. Die Ergebnisse zeigen, dass datengetriebene Methoden in den frühen Phasen des iterativen Prozesses stark sind, da sie den Datenraum erkunden und zu einer schnelleren Erkennung neuer Klassen führen. Im Gegensatz dazu bieten modellbasierte Methoden in späteren Phasen Vorteile, da sie die Entscheidungsgrenzen verfeinern und die Genauigkeit mit zunehmender Datenmenge steigern. Die Überlegenheit der gezielten Stichproben gegenüber zufälligen Stichproben zeigt sich bei unausgeglichenen Datensätzen besonders deutlich. Des Weiteren zeigt die Arbeit, dass die GPU-Beschleunigung die Latenz im iterativen Zyklus reduziert und somit die kritische Schwelle von weniger als einer Sekunde pro Auswahlschritt einhält, was flüssige Interaktionen ermöglicht. Die Ergebnisse legen nahe, dass die Strategien einen vielversprechenden Ansatz darstellen, um komplexe Bilddatensätze effizienter und gezielter zu untersuchen.

Abstract

Analyzing unstructured data such as images presents a major challenge for exploratory data analysis due to their high dimensionality. The data must first be transformed into embeddings, which results in lower dimensionality where the data are more closely grouped. Sampling is essential to make these datasets more understandable for humans through visualization. This work aims to explore how interactive systems can provide representative and interpretable samples quickly, even from large and unbalanced image datasets. Standard methods like random sampling can reach their limits and often fail to capture rare classes, leading to biases in interpretation. Within a standardized interaction protocol, various data-driven strategies (e.g., farthest sampling and D^{α} sampling) and model-aware strategies (e.g., min-margin and disagreement) are compared with random sampling. The goal is to investigate which strategies offer the best balance between fast class discovery, high model accuracy, and low latency within a defined interaction budget. The data points are classified by the model; therefore, this is a simulated experiment. The results show that data-driven methods are strong in the early stages of the iterative process, as they explore the data space and lead to faster discovery of new classes. In contrast, model-aware methods offer advantages in later stages, as they refine the decision boundaries and efficiently increase accuracy as labeled data become available. The superiority of targeted sampling over random sampling is particularly evident with unbalanced datasets. Furthermore, the work shows that GPU acceleration reduces latency in the iterative cycle, thus maintaining the critical threshold of less than one second per selection step, enabling smooth interactions. The results suggest that these strategies represent a promising approach for more efficiently and selectively investigating complex image datasets.

Contents

CHAPTER 1

Introduction

Visualizations are an indispensable tools for data exploration through human perception and interactivity [KVHD17]. However, analyzing unstructured data such as images, audio, or video files presents a particular challenge, as they do not follow a fixed scheme and are often inconsistent, which makes it difficult to extract meaningful insights without appropriate analytical methods [BHS25]. To gain insights from such data, exploratory data analysis (EDA) is essential, because it aims to maximize data visibility, identify outliers, and visualize relationships between variables [KMSC16].

Dimension reduction techniques support this process by transforming high-dimensional data into manageable formats such as scatter plots [SRHA23]. However, this leads to a new problem. Displaying all data points leads to overloading or cluttering in large datasets, causing patterns to be lost in the noise and making the visualization unreadable.

To ensure readability, sampling is necessary. A common approach is random sampling, where data points are selected uniformly. However, this method has its limitations, as it often fails to capture rare or unusual cases. This increases the likelihood of biases and spurious correlations, which can ultimately lead to misinterpretations [DE02].

While statistically driven alternatives exist, such as blue-noise sampling, which achieves a uniform distribution of points, or farthest sampling, which maximizes the minimum pairwise distance and reduces dense clusters [YGW+15], newer distance-based strategies like D^α sampling show that the choice of α values significantly influences clustering performance and data coverage [BNS23]. Nevertheless, these methods often remain purely statistical processes by design without direct integration into the interactive user workflow.

Modern approaches attempt to integrate the model uncertainties directly into the visualization (e.g., ScatterUQ) to make prediction fluctuations visible [LJHW23]. One promising approach is class-incremental learning (CIL), in which the system continuously acquires new knowledge about the classes without losing old knowledge. In this context,

the selection of informative samples for human annotation, also called human-in-the-loop sampling, is crucial to maximizing accuracy [HCL+24].

Current research in this area has significant gaps:

- Systems like mVis focus on multivariate tabular data but do not scale for high-dimensional embeddings and offer few metrics [CBB+19].
- Comparisons between Visual-Interactive Labeling (VIL) and Active Learning (AL) show that VIL can increase efficiency and solve cold-start problems, but the integration of different sampling techniques into this process is still insufficient [BHZ+17, MZW24].

Current research indicates that sampling is mostly viewed as an isolated statistical or model-driven step. It lacks deep integration into interactive visual exploration workflows. This limits the discovery of rare classes and reduces the efficiency of labeling processes, especially with large and complex datasets.

This thesis addresses the gap between statistical and model-aware sampling and interactive visual analysis. The aim is to investigate how different sampling strategies can improve class discovery and the learning efficiency of models. An approach is developed that views sampling not as a static preprocessing step but as an interactive tool within visual exploration. The core idea is to view sampling as part of a closed feedback loop. First, the candidates are added to a panel, and a batch is selected for labeling, which updates the model, and then the process is repeated while comparing data-driven and model-aware strategies with identical interaction protocols. In this setting, the human is simulated by an optimal learner, which provides labels according to an optimal decision process and thus isolates the effects of the sampling strategies while keeping the labeling behavior consistent across experiments. This overcomes the limitations of random sampling and optimizes the usability of high-dimensional data spaces in terms of class discovery speed, accuracy gain, and selection latency. It further analyzes how adaptive exploration-optimization plans influence performance in realistic, iterative environments.

Another important aspect is the response time of the interactive tool, which should match the human's thinking process. True real-time response is faster than 0.1 seconds, which shows instant feedback where users can quickly scan patterns and compare what changed without losing attention. The time between 0.1 and 2–3 seconds is still manageable because it does not affect the users' reasoning process. Everything longer than 10 seconds is interruptive and flow breaking [ERT+17]. For an iterative, interactive discovery process, operations such as refreshing the candidate pool, updating selections, and recomputing scores should stay within the first two response regimes. Otherwise, it is not really interactive because the pauses in between are too large. Longer-running computations should therefore be designed as explicit batch steps rather than blocking the ongoing exploration of the model.

In order to gain insight into what happens during the iterative task, it is advisable to keep the interactive updating process within one second not to break the reasoning and pattern-identification flow. Zgraggen et al. [ZGC+17] show that this one-second threshold is important because most participants liked the instantaneous feedback of the visualizations. Beyond this, latency forces users to pause, which leads to reorientation after each iteration and harms continuous exploration. Their study compares instantaneous, progressive, and blocking visualizations. Users preferred immediate feedback, while blocking visualizations reduced exploration and activity. Even if the total delay to a fully accurate result is constant, users prefer progressive updates with approximate intermediate results over a loading screen. This suggests that low perceived latency is important for sustaining the iterative discovery process.

These observations are directly relevant because they imply that selection, prediction, and update steps should occur within a latency range that ensures the user's mental continuity across several rounds. This also motivates the evaluation not only of accuracy and discovery fraction but also of the time required for the whole process itself, especially for the selection.

1.1 Research Question

- RQ1 – Performance: How do data-driven and model-aware sampling strategies compare to a random baseline in terms of accuracy, class discovery and selection latency?
  - RQ1.1 – What happens if the panel size M is increased?
  - RQ1.2 – Which label budget k is most beneficial for performance gains?
  - RQ1.3 – How does class imbalance affect class discovery and model accuracy under the identical interaction budget?
- RQ2 – Model awareness: How do model-aware sampling strategies differ from data-driven sampling strategies with respect to accuracy, class discovery, and latency under the same labeling budget, and how do these differences evolve across iterations?
- RQ3 – Which components of the loop benefit most from GPU acceleration, and what speedup (CPU vs. GPU) is achieved per selector under the <1s selection-latency constraint?

1.2 Expected Results from Research Questions

To answer research question 1 (**RQ1**), various data-driven and model-aware sampling strategies are compared using a standardized interaction protocol. The evaluation is based on class discovery, test accuracy, and selection latency. The expectation is that

these more sophisticated methods improve class discovery and accuracy, while the latency remains similar.

Regarding **RQ1.1**, it is expected that increasing the panel size M does lead to a higher accuracy. While a larger panel may contain more potentially informative candidates, it also increases the computational cost of the selection process. Therefore, larger values of M are expected to increase selection latency, while gains in accuracy and class discovery remain limited.

For **RQ1.2**, smaller label budgets are expected to favor the exploratory behavior, as only a few points but potentially very diverse points are selected per round. Larger label budgets, on the other hand, should improve the accuracy because more labeled information is fed into the training at each iteration. At the same time, larger batches could increase computational effort and can worsen selection latency. Therefore, a trade-off between class discovery, accuracy, and selection latency is expected.

Regarding **RQ1.3**, class imbalance is assumed to make the class detection more difficult as rare classes are less common in the unlabeled pool and are therefore harder to select. This likely leads to the outcome that some sampling strategies suffer from worse performance than others. Furthermore, the differences between strategies are expected to be clearer in a class imbalance than in balanced datasets.

To answer **RQ2**, model-based strategies are compared with data-driven strategies. It is investigated whether additionally considering the model state when selecting new instances leads to better results. Model-aware methods are expected to offer advantages, especially when sufficient labeled data is already available, which allows for the meaningful use of the uncertainty estimates or model-aware selection criteria. Data-driven strategies are likely to be the strongest in early iterations because they do not require a reliable model state and rely more on structural information of the embedding space.

For **RQ3**, computationally intensive components such as distance- or density-dependent selection, predictions on large candidate sets, and potentially repeated model training are expected to benefit significantly from GPU acceleration. The greatest speed gains are anticipated for highly parallelizable components, such as pairwise distance computations, density estimation, and matrix operations during model training. The goal is to demonstrate that GPU acceleration significantly speeds up the iterative loop without noticeably impacting prediction quality and that this enables better adherence to the interactivity threshold of less than one second per selection step.

CHAPTER 2

State of the Art and Foundations

This chapter provides an overview of the literature that forms the conceptual and methodological foundation of this work. The focus is on interactive labeling workflows for high-dimensional image data, in which visual analysis, human feedback, and machine learning are connected and integrated in an iterative pipeline. Rather than viewing labeling as an isolated annotation task, it should be demonstrated how sample selection, model updates, visualizations, and user interaction influence the performance and usability of such systems. The chapter is divided into several thematic sections and the analyzed literature shows that successful interactive labeling systems must balance prediction accuracy and usability. They must not only identify informative examples but also be transparent, controllable, and responsive enough to allow for continuous human interaction. This motivates the experimental pipeline developed in this work, which investigates how different sampling strategies behave within a unified iterative labeling process.

2.1 Explorative Analysis of High-Dimensional Data

Image and video analysis has gained considerable importance due to advances in artificial intelligence and deep learning and is now applied in many fields, such as medicine, surveillance, sports analysis, video editing, education, and augmented reality. Visualizations are created out of the data to make it more interpretable. It becomes clear that visualizations contribute to a better understanding, filtering, summarizing, and modeling of large and complex image and video datasets. At the same time, the research gaps become apparent with regard to scalability, interactive analysis of large datasets, the explainability of deep learning models, and the closer integration of computer vision and visual analysis [AGH+23].

Image processing models like DINOv2 or OpenCLIP, which is a text and image analyzer, provide highly informative representations for numerous tasks such as classification,

segmentation, depth estimation, image search, and video analysis. The features get extracted from the images or videos and generate a high-dimensional representation [ODM+24]. Analyzing high-dimensional data requires a suitable visual representation. Since such data spaces are not directly interpretable by humans, dimensionality reduction techniques are used to make their structure visible in a low-dimensional version. Dimension reduction techniques calculate the mapping of high-dimensional data into lower dimensions. The goal is to preserve the similarity and distance relationships of the original data as much as possible so that points that are close to each other are also close together in the final embedding. Although projection is often associated with a certain loss of information, techniques to reduce the dimensionality attempt to minimize the resulting mapping error. This enables a point-based visual representation of complex data and thus forms an important basis for exploratory data analysis [EHH12].

Dimensionality reduction is therefore a key element in the visualization of high-dimensional data, but not the only challenge. Especially with large and complex datasets, classical linear and nonlinear projection methods often reach their limits. This makes scalable and control-point-based approaches relevant. Furthermore, effective analysis depends not only on the projection itself, but also on suitable visual coding, interaction possibilities, and abstraction methods that support the exploratory investigation of complex data structures [LMW+17]. Thus, it forms an essential basis for the visual analysis of high-dimensional data but is always associated with a lack of reliability. Low-dimensional projections are very useful for exploratory data analysis but can only approximate the original structures and therefore lead to distortions. A study that examined 133 publications on reliability problems in dimensionality reduction-based visual data analysis shows that such uncertainties arise not only from the projection itself but are also influenced by parameterization, instability, and limited interpretability. Therefore, these projections should not only be understood as exact mappings but rather as helpful and critically evaluated approximations [JLK+25].

One remaining challenge is the visual overload caused by large datasets in scatter plots and parallel coordinate diagrams. Even in low-dimensional visualizations, significant overlaps often obscure important patterns, trends, and density differences. This makes it hard examine these areas in detail without losing sight of the data. Local sampling is a promising approach to address this issue. It can reduce overlaps and visual saturation while keeping the overall structure of the visualization and therefore better supporting exploratory analysis [EBD05].

2.2 Active Learning for Efficient Sample Selection

Active learning (AL) uses machine learning with an active learner to train a model in a supervised way. It describes a training method in which the model selectively chooses which unlabeled instances to label. The principle is that not all data points are equally informative. By selectively choosing fewer labeled examples, comparable or even better model performance can be achieved. This approach is particularly relevant in application

areas where data annotation is time-consuming, costly, or requires extensive expertise. In a pool-based approach, the model iteratively selects samples from a large pool of unlabeled data, which are then labeled by an oracle and added to the training dataset [Set10, CLL+16, ERT+17].

It becomes clear that active learning in real-world applications is more complex than in many idealized experimental setups. Practical scenarios often deviate from classical assumptions, for example, when labels are captured in batches rather than individually, annotators do not work flawlessly, or annotation costs fluctuate significantly. This makes not only the selection of informative instances relevant, but also their diversity, the reliability of the oracle, and the actual effort required for labeling. Furthermore, alternative query types, multi-task learning scenarios, and uncertain model classes expand the scope of active learning. Modern approaches, therefore, increasingly view active learning not only as a strategy for reducing training effort but also as a means of making learning processes more efficient and practical overall [Set11].

With the development of deep learning, the interest in active learning has also increased. While deep learning models achieve very high performances in many application areas, they rely on large amounts of labeled data. Since creating such datasets is often complex and time-consuming, and in many fields only accessible by experts, combining active learning and deep learning makes sense. The combination of both pursues the goal to maintain the performance of deep learning models with as few, carefully labeled data points as possible. While it is inefficient to have individual queries because they lack accuracy when only used alone, batch-based strategies are often necessary to improve the situation. Current research is therefore increasingly focusing on combining uncertainty, diversity, and density criteria, as well as other approaches that additionally incorporate unlabeled data to further reduce the labeling effort [RXC+21].

2.3 Visual Interactive Labeling

In contrast to this, visual interactive learning (VIL) is about interactive machine learning with positive or negative user feedback to support a given classification a model made. The goals might differ because AL wants to obtain a classifier with high performance while paying only a small contribution to the whole dataset. VIL aims to inform the user, including non-experts, who visually and interactively steer a model to a defined concept through iterative feedback between user input and model updates. Additionally, it adds the same advantage as AL to its portfolio because VIL achieves high accuracy with little budget [CLL+16, ERT+17, DK18, BZSA18].

Human-in-the-loop (HITL) refers to learning environments in which the human feedback is not limited to passive annotation but actively influences system behavior via the learning process. As explained in the context of dimensionality reduction, high-dimensional image embeddings are not directly interpretable by humans and must therefore be projected into a low-dimensional visual representation. For this purpose, methods like uniform manifold approximation and projection (UMAP) or t-distributed stochastic neighbor

embedding (t-SNE) can be used to create two-dimensional scatter plots that allow users to examine the local areas, clusters, and potential outliers. The goal is that the user monitors the AI-driven experiment and has the following benefits. The human can adjust the reward function to define the discovery target and help the system by marking or pointing to data points that are visually interesting for the human at each step of the progress. Further, the influence of the human can change the behavior of the system by managing the balance between exploration and exploitation. Also, the human can intervene to define objects of interest [BABEM24, PRL+25].

Bernard [Ber25] proposes the Human-Data-Model Interaction (HDMI) Canvas, an organizational and communication model for visual analytics that explicitly describes visual analytics systems as an interplay of three actors, which are the humans, the data, and the models. In contrast to classical process models, which primarily emphasize the flow of data to generate knowledge, the HDMI Canvas additionally focuses on the externalization of human knowledge, data exploration, and model explanation as equally important contributions in the EEE framework (Human Knowledge and Preference Externalization, Data Exploration and Model Explanation). Central to this is an expanded view of feedback loops where the user feedback includes not only labels and parameter selection but can also influence similarities, weights, groupings, anchors, and labels. Simultaneously, models are no longer understood merely as output producers but as active actors whose explainability, interpretability, and reliability should be systematically integrated into the interaction. The canvas thus serves two purposes. It possesses descriptive power to more precisely differentiate existing visual analytics approaches and generative power to structurally design new, user-centered, and interdisciplinary visual analytics designs. In this sense, visual interactive labeling systems can be understood as concrete visual analytics workflows that implement these human-data-model feedback flows in practice. The following paragraphs present some representative frameworks that operationalize the HDMI perspective for iterative labeling by coupling model outputs, visual representations, and user decisions in a recurring loop.

Visual Interactive Labeling (VIAL) is a framework that merges the core ideas from active learning and visualization into an iterative, visual-interactive workflow. The key distinction is that the labeling has not only meaning to the model training but is more of an explorative activity. This means that the growing labeled dataset, the improved learning model, and the domain knowledge for the user gained while interacting with the system are all equally important. VIAL introduces a concept that sees labeling and instance selection not as isolated actions but as a part of a recurring cycle in which the model outputs, the visual representations, and the user decision continuously influence one another. The process integrates machine learning with interactive visualization to transform raw data into a refined model while having a human expert deliver input to the sampling process. It begins with preprocessing and feature extraction, which cleans the data and maps it into a usable feature space. The main consideration is to make it visible to the user because interpretable features can aid understanding and control but also introduce bias in complex representations. The model uses the user-labeled

feedback to make predictions and updates quickly enough to support interactive use. The visualization is the core interface for the user to allow exploration of the model output to see if a pattern arises. For the interface to present accurate judgments, there is a candidate suggestion system that highlights the balance between automated selection and user-driven exploration [BZSA18]. Therefore, it is relevant to this work because it presents interactive labeling as a coupled process of visual inspection, candidate recommendation, user input, and rapid model adaptation, rather than a purely sequential annotation task.

Visual interactive labeling tool for object detection, or short VILOD, is an object detection and labeling system that combines active learning, visualization, and annotation in an interactive workflow. It enables experts to explore unlabeled data, verify the current state of the model, and create bounding box annotations through a single user interface. Studies show that a balanced, user-driven strategy can outperform an automated baseline and improve both labeling efficiency and the traceability of AI-powered data preparation [Hol25].

Based on VIAL and VILOD, interactive labeling systems benefit when model suggestions are embedded in visual interfaces that remain understandable and controllable for the user. They further illustrate, however, that the success of such systems depends not only on the underlying model but also on how well the interface supports orientation in the candidate selection. This motivates approaches that place even greater emphasis on the visual guidance itself.

VisGIL by Grimmeisen et al. [GCT22] is a visual interactive model that uses visual cues to guide users in the selection process. Visual analytics proves to be useful because it focuses on the visual presentation of data using varying projection methods like plots or glyphs and is an effective way to transfer information to the user and increase their understanding. The method starts with the user selecting and labeling instances, while the person is guided from the very beginning by suggested useful instances with visual cues. This helps the model to avoid a cold start. The user selects candidate points by click or lasso, refines the selection in a detailed view, and assigns labels, which are added to the training set so that the classifier is iteratively trained. It continues until the performance is satisfactory and the model starts to assign labels to unlabeled points automatically. The approach has two main components, which are candidate suggestions and result visualization. The first process is driven by an uncertainty calculation where the margin of error is minimized and representativeness where the information density is calculated using Euclidean similarity. Both uncertainty and representativeness need to be present and are traded off between each other depending on the problem setting. Further, the result visualization is relevant because it projects high-dimensional data into 2D and overlays it with model information on predicted or assigned labels with color. Different icons and icon sizes are used for labeled and unlabeled points, with the addition that the user gets a signal for recommended instances.

ScatterUQ extends the visual representation by making model uncertainties contextually interpretable. Based on uncertainty-aware models, it combines class probabilities with outlier values and explains predictions through distances to class prototypes in an

embedding space. A key contribution is the shift from a global to a local 2D scatter plot view, which preserves neighborhoods and distances without clutter. Different projection methods are compared to each other, like principal component analysis (PCA), t-SNE, and UMAP. ScatterUQ thus transforms uncertainty into a visual object for analysis, supporting debugging, data quality assessment, and more informed decision-making in iterative labeling workflows [LJHW23].

Amershi et al. address the central challenge of the human-AI collaboration. AI functions make decisions under uncertainty, which leads to inconsistent behavior and potentially confuses users and therefore undermines trust. Classic usability heuristics of consistency and error prevention are only partially effective because AI models are probabilistic and change over time. Therefore, they propose 18 design guidelines for human-AI interactions, which serve as a framework for testing and design. These guidelines are not specifically designed for interactive machine learning or human-in-the-loop approaches but are intended for the generalization of the human-AI collaboration [AWV+19]. The guidelines are listed in Table 2.1.

The guidelines are derived from over 150 existing recommendations from research and industry. In the discussion, the authors categorize the guidelines as a general, observable interface framework. They emphasize trade-offs between guidelines like complex models versus transparency and see an emerging need for domain-specific additions for high-risk areas. Overall, these guidelines provide a compact, tested set of heuristics for designing such AI functions in a way that users understand and control AI behavior better [AWV+19].

The use case they consider is opportunistic composition, where applications are opportunistically assembled. The Opportunistic Composition Engine (OCE) uses reinforcement learning for this purpose and receives feedback when users accept, reject, or modify assemblies via an interface. The analysis shows that while OCE does not require machine learning expertise, it generates a high workload because the users have to frequently review suggestions and the effect of actions is not very transparent. To improve this, the Amershi Guidelines [AWV+19] are used explicitly, plus two additions of making the system clear about how it works and providing interpretable output. Better explanations of capabilities and limitations, more understandable representations, and flexible strategies are suggestions, therefore. Particularly relevant areas for further research are highlighted by formalizing and communicating uncertainty and more granular feedback at multiple levels to guide learning more precisely. In conclusion, it emphasizes that interactive machine learning is hardly practical without good human-AI interactions because the system must fulfill criteria to be fair, transparent, and secure to work with users [DTAA25]. These design principles are relevant, as the usefulness of an interactive labeling system depends not only on the quality of its sampling strategy but also on whether users can understand why certain data points are selected as candidates and displayed to them and how much effort each interaction requires.

	AI design guideline
G1	**Make clear what the system can do.** Help the user understand what the AI system is capable of doing.
G2	**Make clear how well the system can do what it can do.** Help the user understand how often the AI system may make mistakes.
G3	**Time services based on context.** Time when to act or interrupt based on the user's current task and environment.
G4	**Show contextually relevant information.** Display information relevant to the user's current task and environment.
G5	**Match relevant social norms.** Ensure the experience is delivered in a way that users would expect, given their social and cultural context.
G6	**Mitigate social biases.** Ensure the AI system's language and behaviors do not reinforce undesirable and unfair stereotypes and biases.
G7	**Support efficient invocation.** Make it easy to invoke or request the AI system's services when needed.
G8	**Support efficient dismissal.** Make it easy to dismiss or ignore undesired AI system services.
G9	**Support efficient correction.** Make it easy to edit, refine, or recover when the AI system is wrong.
G10	**Scope services when in doubt.** Engage in disambiguation or gracefully degrade the AI system's services when uncertain about a user's goals.
G11	**Make clear why the system did what it did.** Enable the user to access an explanation of why the AI system behaved as it did.
G12	**Remember recent interactions.** Maintain short term memory and allow the user to make efficient references to that memory.
G13	**Learn from user behaviour.** Personalize the user's experience by learning from their actions over time.
G14	**Update and adapt cautiously.** Limit disruptive changes when updating and adapting the AI system's behaviour.
G15	**Encourage granular feedback.** Enable the user to provide feedback indicating their preferences during regular interaction with the AI system.
G16	**Convey the consequences of user actions.** Immediately update or convey how user actions will impact future behaviour of the AI system.
G17	**Provide global controls.** Allow the user to globally customize what the AI system monitors and how it behaves.
G18	**Notify users about changes.** Inform the user when the AI system adds or updates its capabilities.

Table 2.1: Human–AI interaction design guidelines [AWV+19].

2.4 Class Discovery in Interactive Settings

A key motivation for HITL learning is that high model quality does not necessarily require extensive annotation if human feedback is used strategically. The value of the human lies not only in providing the labels but also in contributing domain knowledge that helps to identify the most informative, ambiguous, or novel examples. This is particularly important in environments with sparse monitoring, complex high-dimensional data, or heterogeneous class structures, where the purely automated processes could waste annotation budget on redundant and uninformative samples. From this perspective, the main question is not simply how to label more data, but how to select the right data so that each interaction contributes the most to model improvement and class discovery. The iterative loop thus serves as a mechanism to introduce human knowledge into the learning process while limiting the labeling effort [WXS+22].

These studies demonstrate that HITL systems are very effective when interaction is not limited to isolated corrections but supports a continuous feedback process beyond general human-AI collaboration. HITL is particularly relevant in the demanding field of class discovery. In such scenarios, the pool of unlabeled data commonly contains a mixture of examples of already known and previously unknown classes. This makes the issue fundamentally more difficult than the standard labeling of completed datasets because newly selected data points cannot simply be assigned to new categories. Instead, the system must distinguish between additional clues to known classes and genuinely new structures in the data. For an iterative labeling pipeline, this means that the sampling strategy must not only improve accuracy for known classes but also enable the identification of potential candidate examples that could reveal new classes in time [VHVZ22].

Novel class discovery and generalized category discovery differ fundamentally from the context considered in this work. The goal of discovering new classes is to identify entirely new classes in an unlabeled data pool without using labels. These classes are often only named after their discovery. This is more of an unsupervised or semi-supervised discovery. New classes are visible as groups but are not integrated into the training classification model immediately [HVZ19, JPA+22]. Generalized class discovery or category discovery is a different setting. At each round the model acquires a new batch of unlabeled data that contain images or image embeddings from classes that may or may not have been previously observed. The goal is to maintain performance of the previously observed categories while also discovering novel ones [ZA23]. This is different from novel class discovery, which assumes no overlap between labeled and unlabeled classes. But it is not a realistic scenario. It is far more realistic that the unlabeled set contains unlabeled and labeled classes [VHVZ22].

As shown in their study, Vaze et al. [VHVZ22] prove that their method is at least as good as a comparative baseline and a state-of-the-art novel class discovery method. The results show that the generalized class discovery outperforms the novel class discovery. It can be observed that these novel class techniques are good at detecting old classes but

lack accuracy on the newly found classes, which is interesting as they just select new classes in the iteration. The worse performance on the new classes leads to an overall lower accuracy because the generalized method is better balanced between old and new classes. It removes the limiting assumption by embracing real-world ambiguity. It simply acknowledges that data is messy and categories are not clearly defined [VHVZ22].

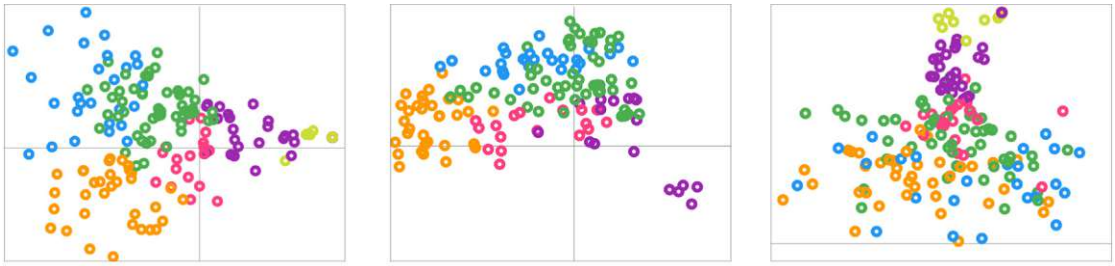
As part of this work, newly discovered classes are not only identified as those but also directly included in the labeled dataset and used for subsequent retraining of the model. The different procedure makes it much closer to active class discovery or human-in-the-loop class-incremental learning than a pure novel class discovery (NCD) clustering.

2.5 Interaction Constraints for Iterative Visual Analysis

In the field of visual, interactive data labeling, two fundamental interface paradigms can be distinguished, which are instance-centric and class-centric labeling. Instance-centric systems typically present many data points simultaneously and allow the user to examine and label individual instances or local groups. Class-centric systems, on the other hand, organize the interaction around one class at a time, reducing the labeling decision to a focused assessment of class memberships. The distinction is important because it impacts scalability, cognitive load, and usability when processing larger datasets. The following section discuss both perspectives and explain why this trade-off is relevant for iterative analysis environments.

First, there is instance-centric visual interactive labeling. Chegini et al. [CBB+19] present a technique called multivariate Visualizer (mVis), which is an interactive visual analytics system for labeling multivariate data with unknown labels at the beginning. The starting idea is a collection of records whose labels are unknown. The system cannot assume fixed ground-truth labels from the beginning, and the label set has to be constructed continuously. The analyst, therefore, needs to add, rename, and delete labels when the knowledge becomes clearer. The workflow starts with all data points being assigned to an unknown partition. The analyst creates a partition and selects some records belonging together and assigns and names the partition. The system is doing active learning from the partitions but also delivers suggestions to the user, which must be approved or rejected when enough ground truth labels exist. Further, AL is useful for improving label quality and reducing confusion in difficult regions of the data. This is an essential step for trust and keeping the control on the human side. In mVis the idea is operationalized by using scatter plots, SPLOMs, or similarity maps produced by PCA,t-SNE, multidimensional scaling, and parallel coordinates, so users are able to explore the data and manually create or refine partitions that correspond to new classes.

A machine learning algorithm is integrated as guidance rather than automation. The classifier can suggest additional records for a given partition under a controllable confidence threshold, and the active learning component recommends informative records to label as next. The user is always in control by accepting, rejecting, or revising suggestions. The system can therefore generate another inspectable visualization. This completes



(a) *MDS* **(b)** *PCA* **(c)** *t-SNE*

Figure 2.1: Different projection techniques from Chegini et al. [CBB+19].

the whole loop of exploration, labeling, and model feedback. Each new decision updates the partition and the visual structure of the data, and over rounds the whole label set becomes coherent and complete, while the user retains full control over the ultimate ground truth labeling.

One issue with VIL systems is that they are instance-centric and 2D scatter plots are generated by t-SNE or UMAP to show all points with the predicted labels. Showing everything might be effective for small and simple datasets, but the larger the dataset, the more it becomes a problem. The visualization looks dense and cluttered with overlapping classes, and projection distortions make selecting many correct labels difficult [MSB+25]. Furthermore, the problem is intensified with an increasing number of classes, as the color or shape coding and visual separation do not scale well in 2D projections, and the user has to repeatedly select from a large number of labels. To address these issues, Matt et al. [MZW24, MSB+25] propose class-centric VIL (cVIL). The methodology differs in that, instead of showing all instances in 2D, cVIL shifts the labeling task from assigning a class to an instance to assigning instances to a selected focus class, which thereby makes the decision a predominantly binary membership check and thus reduces the cognitive load. The interface only shows aggregated per-class distribution. Specifically, it combines a class overview for monitoring label progress and class imbalance with class-specific analysis views that support the selection of multiple instances and batch processing of labels. In simulation, cVIL matches AL for the instance labeling and outperforms it even for batch labeling on more complex datasets. In the controlled user study there are reports for higher final labeling accuracy and strong user preference for class-centric VIL compared to instance-centric VIL, like mVis. Additionally, the interaction and retraining times remain manageable. Overall, cVIL can be seen as a scalability-oriented redesign of visual interactive labeling. It reduces the information overload by focusing the attention on one predicted class at a time while maintaining an instance-level solution for difficult residual cases [MSB+25, MZW24].

For the thesis, the distinction between instance-centric and class-centric interaction is important, even though the proposed pipeline does not fully implement a dedicated cVIL interface. It illustrates a general design compromise. Displaying many candidates simultaneously allows for comprehensive exploration, while it also increases the cognitive

load and can impair the usability as the number of instances grows. This is directly relevant to the choice of panel size, update behavior, and the data point selection in the interactive loop examined.

Especially in the context of the visual interactive labeling, visual stability is crucial in progressive, continuously updated visualizations. The reason is that changes in the display disrupt the user's spatial orientation. It is a core problem when monitoring a sequence of results. If a view changes significantly, it becomes more difficult to read because it is no longer possible to reliably distinguish whether the users are seeing genuine change in the content or merely a display that is reordering or rescaling due to the ongoing process. For this reason, changes need to be limited to increase understanding. In visual analysis systems, visual stability is not merely cosmetic but a core functional criterion. Users must assess ongoing calculations by interpreting intermediate results within the overall context. For this to work, changes in the visualization must be proportional to the data changes. Constant visual jumps significantly hinder this and can lead to misinterpretations. It is further crucial for interaction. Every major shift forces the users to rebuild their mental reference. Stable anchors act as landmarks, which help users maintain a coherent mental map. This allows updates to appear as a logical progression rather than a disruption. Ultimately, visual stability is what enables the meaningful use of progressive systems by providing a continuous, interpretable representation of progress without limiting the dynamics [ASSS18].

CHAPTER 3

Conceptual Framework

This chapter defines the conceptual framework for the iterative labeling process investigated in this work. The focus is on linking sampling, user interaction, labeling, and model updates in a visual, iterative learning loop. First, the general idea of the coupled iterative process is introduced. Then, model-aware and data-driven sampling strategies are distinguished. Model-aware strategies use the current state of the learner to identify informative samples, while data-driven strategies are based on structural properties of the data such as distance, density, or coverage. Finally, the behavior of different learning models during the repeated updates throughout the labeling process is explained.

3.1 Conceptual Idea

Sampling is not to be understood as a one-off preprocessing step but as part of a coupled iterative system. In visual interactive labeling, selection, annotation, and model updates are linked across multiple rounds so that each step influences the next. Therefore, the quality of a sampling strategy cannot be judged based on a single choice but only on its behavior over several rounds, during which the consequences on learning and class exposure gradually become more apparent.

The process proceeds over repeated interaction rounds. In each round, only a subset of all data is visible and available. From this subset, the samples are selected for labeling, transferred into the labeled set, and then used to update the model before the next iteration begins. A new iteration is triggered as soon as labels have been assigned to the selected batch and integrated into the labeled dataset. In the simulated environment of this work, the trigger occurs automatically. After the learner has provided labels for the samples, the model is retrained or updated, samples are taken out of the pool of unlabeled data, and a new panel is generated. From one iteration to the next, no manual action is required.

Users should be able to interact with the visual representation so that it reflects their analytical thinking, such as moving, highlighting, annotating, or searching documents. It can be observed that when the user collaborates with the machine, the system interprets these actions as meaningful feedback. The model then updates its internal weights and recalculates the layout. It is important that the model and the visualization are tightly coupled and that the visualization is not a static output but rather a workspace for the user, which makes the whole system iterative [EFN12, BLBC12]. It further demonstrates that even a few iterations of interaction can improve the classification and generate more meaningful insights into the data [BLBC12].

This supports the idea of a coupled visual-interactive cycle. The user views a visual representation and interacts with it. The system interprets this interaction as feedback, updates the model, and presents a modified visualization in the next step. It is relevant to the concept of this work because the different steps of iterative learning are linked to each other. The quality of a sampling strategy should therefore be evaluated based on its behavior throughout the entire iterative process, not just on the samples selected in a single round.

3.2 Model-Aware Sampling

Model-aware sampling selects data points based on the current state of the training model. Unlike purely data-driven strategies, the utility of a sample is evaluated in relation to the current model prediction, model reliability, and areas with uncertain or unstable decision boundaries. This makes model-aware strategies appealing in iterative labeling loops because they can concentrate the annotation effort on the points that are expected to change the model most significantly. At the same time, they require repeated access to the updated model state and can generate redundant samples if uncertainty is concentrated in the same local area. Therefore, model-aware selection is closely linked to the costs of retraining and redundancy control. Filtering or diversifying candidates using model-based similarity retains the benefits of uncertainty-based sampling while reducing the repeated selection of essentially the same evidence for the model [HDG+21].

3.2.1 Uncertainty sampling

One of the best-known model-aware methods is uncertainty sampling, where the data point is selected to label next by picking the model's most uncertain ones. From possible classes $\Omega = \omega_1, ...\omega_C$, the new point x has a provided class probability of $p(\omega|x)$. In practice, these uncertain points can be found near the decision boundaries of a model. It is reasonable to assume that the closer it is to the decision boundary, the less confident the model is about that point and its class [HLM+24].

There are two types of probabilistic predictions, which are aleatoric and epistemic strategies. Aleatoric strategies are those in which the predicted uncertainty is only based

on the combination of informativeness scores with predictions of a deterministic model. The following belong to this category [HHN+24].

- Least confidence prioritizes data points where the model has low confidence about the prediction. It checks for those points whose highest probability is small:

$$\mathrm{LC}[y \mid x, \omega] = 1 - \max_y p(y \mid x, \omega). \tag{3.1}$$

- Margin sampling focuses on data points where the model is struggling to separate the top two candidates, which assumes that their predictive probability gap is small:

$$\mathrm{MS}[y \mid x, \omega] = 1 - (p(y_1 \mid x, \omega) - p(y_2 \mid x, \omega)), \tag{3.2}$$

  where y_1 and y_2 are the first and second most predictive labels.

- Predictive entropy [Sha48] measures the overall uncertainty of the predicted class distribution and selects points with high uncertainty:

$$H[y \mid x, \omega] = -\sum_{y=1}^{K} p(y \mid x, \omega) \ln p(y \mid x, \omega). \tag{3.3}$$

  where K is the number of classes, and $p(y = k \mid x, \omega)$ is the predictive probability assigned to the class k for input x under model parameters ω.

Uncertainty strategies that do not fall under this category belong to the epistemic ones. These refer to acquisition methods that rely on the predictive uncertainty of Bayesian or approximate Bayesian models. The predictive distribution in Equation 3.1 marginalizes over uncertain model parameters, which means that the resulting uncertainty captures not only data noise but also epistemic uncertainty due to limited knowledge.

The introduction to the topic of uncertainty clarifies the range of available methods for estimating the prediction distribution. This makes it possible to focus on selected model-aware approaches, which are then analyzed in more detail.

3.2.2 Min-margin

A common uncertainty sampling heuristic for multi-class classification is margin sampling, which picks the instances the model is most ambiguous about its labels. The margin, which is considered, is:

$$x_M^* = \arg\min_x p_1(x) - p_2(x), \tag{3.4}$$

where $p_1(x)$ and $p_2(x)$ are the highest and second-highest scoring probabilities under the model. These are defined as follows:

$$p_1(x) = \max_y p(y \mid x, \omega) \quad \text{and} \quad p_2(x) = \max_{y \neq \arg\max_{y'} p(y' \mid x, \omega)} p(y \mid x, \omega).$$

This method calculates the difference between the probabilities of the two most likely classes. It is considered more informative than the lowest confidence level strategy because it considers both the most likely and the second most likely class. Consequently, it can overcome a limitation of the lowest confidence level criterion. The idea behind it is that a small margin means that the model is uncertain between the two best choices, which therefore indicates that the instance is more informative. If the margin were large between these two probabilities, then it would be easy for the model to differentiate between the two labels. Despite this improvement, min-margin sampling ignores the rest of the output distribution for remaining classes, which remains a drawback for problems with immense label sets and is addressed by the next sampling technique [Set10].

3.2.3 Entropy-based disagreement sampling

The disagreement calculation relies on the high entropy, which assumes that the data points are maximally uncertain [NGH+25]. The entropy is the information-theoretic measure of the amount of information that is required to derive the predictive distribution. It measures the impurity or uncertainty of the model's prediction. The motivation behind it is that the other strategies only depend on the most likely label and all the information is withdrawn, while entropy aggregates the uncertainty over all classes and uses the whole posterior mass. An important qualitative nuance is that the entropy does not particularly favor points where it is nearly impossible to assign a single label but where the remaining density is distributed among others. The label, which is close to zero, is defined as incorrect by the model. The overall distribution is less uncertain than a truly uniform one. This indicates that the entropy can focus on distinguishing competitive labels rather than ranking uninformative ones. The entropy-based uncertainty criterion selects the samples whose predictive distribution is maximally uncertain:

$$x_H^* \;=\; \arg\max_{x \in \mathcal{U}} \; H_\theta(y \mid x), \tag{3.5}$$

where $\mathcal{U}$ is the unlabeled pool and $H_\theta(y \mid x)$ is the Shannon entropy defined by 3.3. For the binary case, $C = 2$, the uncertainty sampling looks for the class close to 0.5, which is random distribution. In multi-class classification, the most informative samples are those whose predicted class probabilities most closely follow a uniform distribution, as this corresponds to the highest model uncertainty. This suggests that the probability mass is spread across several classes, indicating strong competition between them and low confidence in the individual predictions [NGH+25]. Since entropy is defined solely on the basis of prediction probabilities, it can be generalized well. It can be applied when a model provides a probability distribution over the outputs and it is further possible to use entropy-based uncertainty sampling for models that are not explicitly probabilistic by constructing approximate posterior distributions [Set10].

3.2.4 Agreement sampling

There is a closely related technique to disagreement-based querying that can be viewed as its complement. It refers to agreement- or confidence-based acquisition. Instead of

prioritizing data points where the model is uncertain or divergent, the agreement-based method exploits the model's high-confidence regions. Therefore, it reduces the human labeling and propagates labels automatically. The key is the estimated probability that a predicted label is correct, which is derived from the predictive distribution at each iteration and label category. The following is the predictive distribution output of the model over the category's label space V_m:

$$\pi_t^m(v) = p(y_t^m = v \mid x_{1:t}), \qquad v \in V_m. \tag{3.6}$$

As it exploits the high-confidence region, the maximum a posteriori (MAP) estimation is as follows

$$\hat{v}_t^m = \arg\max_{v \in V_m} \pi_t^m(v). \tag{3.7}$$

This prediction is summarized into a confidence score that estimates whether the chosen label is correct:

$$c_t^m = \max_{v \in V_m} \pi_t^m(v) = \pi_t^m(\hat{v}_t^m). \tag{3.8}$$

The maximum predicted probability means that the higher the c_t^m is, the more the model agrees strongly, and the prediction is assumed to be reliable enough to be accepted. If the $\pi_t^m(\hat{v}_t^m)$ can be misleading, the probabilities are poorly calibrated [NGH+25].

Taken together, these model-aware criteria operationalize different conceptions of uncertainty and confidence. They are relevant to this work because they allow the user to influence the current state of the learner, which candidates are presented next, and thus directly link sample selection to the evolving data representation of the model.

3.3 Data-Driven Sampling

Data-driven sampling bases sample selection on the structural or geometric properties of the data rather than the current state of the classifier. It comes from the distinction between data-driven and model-aware sampling in active learning. The basic idea is to identify informative samples, often based on their position in the feature space, for example, through diversity, density, coverage, or distance to previously selected points. Such strategies are particularly appealing in early iterations, because the model may still be weakly or poorly calibrated at this stage, and purely uncertainty-based assessments are therefore less reliable. In this sense, data-driven methods use the geometric coverage of the data manifold and can uncover underrepresented regions even before the classifier has developed meaningful confidence estimates [LHW+22].

Since exact optimizations are combinatorially expensive, the use of heuristics is encouraged. Practical solutions are therefore greedy approaches and incremental k-medoids, which start with a greedy initialization and then improve the representatives through local swaps, which leads to decreasing distances within the assigned regions. Both methods provide incrementally expandable selection and therefore form a robust basis for the following data-driven sampling methods in the interactive pipeline [YK10].

3.3.1 Farthest optimization

The farthest selector starts from an initial center position among the already selected points. Iteratively, the points are added whose distance to their nearest center is maximal. Mathematically, $\Delta(\cdot,\cdot)$ is the distance in an embedding space, and S_t is the current set of selected centers after t steps, which are initialized by any element from the already labeled set. The farthest-first (k-center greedy) update is

$$u_t \;=\; \arg\max_{i\in\mathcal{U}\setminus S_t}\; \min_{j\in S_t} \Delta(x_i, x_j), \qquad S_{t+1} \;=\; S_t \cup \{u_t\}, \tag{3.9}$$

i.e., the points whose distance to the nearest selected center is maximum [Gon85]. That is why it is often also called a minimum-maximum selector. This rule is used to approximately solve the k-center covering problem

$$\min_{S:\,|S|\le k}\; \max_{i\in\mathcal{X}}\; \min_{j\in S} \Delta(x_i, x_j), \tag{3.10}$$

which selects the k centers such that the maximum distance of each point to its nearest center is minimized. This expansion constructs a set of k centers such that the worst assigned radius can be within a factor of 2 of the optimal k-center solution. Each newly chosen center must be at least some distance h away from the other previously chosen centers. This leads to the assumption that the true optimum is never smaller than h, while the greedy solution's max radius is at most $2h$, which can be seen in the work by Gonzalez [Gon85], where

$$\max_{i\in\mathcal{X}}\; \min_{j\in S_{\text{greedy}}} \Delta(x_i, x_j) \;\le\; 2\,\text{OPT}. \tag{3.11}$$

Sener & Savarese [SS18] connect this farthest-first/k-center greedy approach to batch active learning by ranging selection as a core-set problem. They formulate selection as choosing s_1 under budget b on top of the current labeled set s_0 to minimize the distance to the center

$$\min_{s_1:\,|s_1|\le b}\; \max_{i\in[n]}\; \min_{j\in s_0\cup s_1} \Delta(x_i, x_j), \tag{3.12}$$

which solves the k-center goal by a greedy farthest-first update if it is proven that $\max_i \min_{j\in S_1\cup S_0} \Delta(x_i, x_j) \le 2\times \text{OPT}$. The performance is controlled by how well the labeled subset covers the full dataset. Under the Lipschitz assumption that near inputs have the same or similar class probabilities, they limit the core set loss by the coverage radius δ upwards, so the actual goal is to select new points in order to minimize, which is exactly the goal of k-center. It is done by a greedy algorithm, which is essentially identical to the original implementation from Gonzalez [Gon85] and does repeated selection [SS18].

3.3.2 Core density sampling

Another sampling strategy is to use density-peaks clustering as a selector. Rodriguez and Laio [RL14] propose clustering by fast search and find of density peaks. It is built on the

idea that cluster centers should lie in regions of high local density and are far away from any other data point with high density. The method needs a pairwise distance function d_{ij} and can, just with this little information, identify non-spherical clusters without fitting parametric distributions or iteratively optimizing an objective as in k-means.

For each point i, density-peaks clustering computes two quantities, which are a density p_i and a distance-to-higher-density δ_i. The simple density estimator counts neighbors with a given cutoff distance d_c:

$$\rho_i \;=\; \sum_j \chi(d_{ij} - d_c), \qquad \chi(x) = \begin{cases} 1 & x < 0, \\ 0 & \text{otherwise}, \end{cases} \tag{3.13}$$

so that ρ_i is the number of the points closer than d_c to i. In their paper, it is mentioned that for large datasets, the results are robust to the exact choice of d_c and suggest choosing d_c such that it has, on average, $\approx$ 1–2% of the dataset as neighbors. Next, δ_i is set as the minimum distance from i to any point with higher density:

$$\delta_i = \min_{\rho_j > \rho_i} d_{ij}, \quad \delta_{i^*} = \max_j d_{i^* j}. \tag{3.14}$$

Points that have both a high value for ρ and δ stand out as the density peaks and are the cluster centers. Points that simultaneously exhibit high local density and a large distance to denser points are natural candidates for cluster centers, as they are both representative of a local region and well separated from competing dense regions.

The scalar value that is used in practice is

$$\gamma_i \;=\; \rho_i \, \delta_i, \tag{3.15}$$

where a large γ_i indicates a point that is both dense and well separated from denser areas, which makes it a strong candidate for a center. Once the centers are selected, the density-peak clustering assigns each remaining point to the same cluster as its nearest neighbor with higher density:

$$\text{cluster}(i) \;=\; \text{cluster}\left(\arg \min_{j:\, \rho_j > \rho_i} d_{ij} \right). \tag{3.16}$$

This leads to a one-time assignment instead of iterative refinement. The method also provides an idea of halo points, which are points on the edges or points with low reliability. It is examining whether the cluster boundaries are near d_c and core points are separated from ambiguous areas, which is useful for dealing with noise or outliers without setting a global density threshold [RL14].

An important motivation for density-based selection in learning scenarios is that pure uncertainty-based strategies can oversample query outliers. Points can be uncertain simply because they are atypical, so flagging them does not improve the performance in the overall data distribution. Settles discusses this issue and justifies information density

as a principled way to balance informativeness against representativeness [Set10]. More specifically, the idea is to start from a base acquisition value $\phi_A(x)$ (e.g., entropy, margin, QBC) and then incorporate a concept of how representative x is of the unlabeled pool so that highly informative but isolated points are downweighted and queries are biased towards dense regions of the input space [Set10]. This helps mitigate the uncertain outlier error mode.

In an iterative labeling loop, selecting such peaks prioritizes points that are likely to represent many nearby unlabeled points and distribute the selection across different areas of the data manifold. The density selection can select the top k points according to γ_i within the current candidate pool and treat them as representative anchors for subsequent labeling and model updating [Set10].

This is advantageous for the interactive loop because density peaks can serve as representative anchors from different regions of the embedding space. As a result, the selector focuses less on uncertain outliers and instead promotes a broader structural coverage of the candidate pool.

3.3.3 Distance-based sampling

The k-means problem is formulated by the potential for a set of centers:

$$\phi(C) = \sum_{x \in X} \min_{c \in C} \|x - c\|^2. \tag{3.17}$$

The k-means algorithm alternates between each point being assigned to its nearest center and each center being recalculated as the mean of the points assigned to it. This process is repeated until the assignments no longer change. A standard identity used to justify the improvement is that for a cluster S with means $c(S)$ and any point z,

$$\sum_{x \in S} \|x - z\|^2 - \sum_{x \in S} \|x - c(S)\|^2 = |S| \cdot \|c(S) - z\|^2, \tag{3.18}$$

the choice of mean minimized the squared distances within a cluster and therefore reduced ϕ. This makes the method attractive because it is simple and usually fast. However, it only guarantees that a local optimum will be reached. This can lead to poor solutions as the centers are assigned poorly [AV07].

An important improvement is k-means++ seeding. Instead of choosing arbitrary centers, the first center is selected randomly, and then centers are repeatedly added by focusing the selection on points that are far from the current set of centers. They define $D(x) = \min_{c \in C} \|x - c\|$, but in k-means++ the next center x is selected with a probability proportional to $D(x)^2$. The D^2 distortion makes it likely that every true cluster will be hit by at least one initial center and thereby prevents errors of random initialization [AV07].

The D^α-sampling technique corresponds to the direct generalization of the idea from before.

$$p_X^{(\alpha)}(z) = \Pr(z \text{ chosen next}) = \frac{D(z)^\alpha}{\sum_{x \in X} D(x)^\alpha}, \qquad \alpha \geq 0. \tag{3.19}$$

Here the α-powered bias replaces the squared distance bias, which is often called D^α seeding. The α controls the seeding exponent in D^α-sampling. The value has the following different behaviors:

- $\alpha = 0$ reduces to uniform sampling
- $\alpha = 2$ is the k-means++
- Large α values mimic the selection of the farthest-first principle

Recent analysis in Bamas et al. [BNS23] has shown that the best α can be dataset dependent. In their paper they show that different values of α lead to different outcomes. One indicator of the difference is the dataset itself. It leads to the conclusion that larger α-values can be beneficial for a given task. Experiments in [BNS23, BDW19] show that time and accuracy can vary for different values, but especially values larger than the standard k-means++ prove to be solid, leading to decreasing time consumption and increasing accuracy. The core formula is

$$p_X^{(\alpha)}(z) = \frac{\min_{c \in Z_t} \|z - c\|^\alpha}{\sum_{x \in X} \min_{c \in Z_t} \|x - c\|^\alpha}, \tag{3.20}$$

and the justification is that $\alpha = 2$ adopts the classic k-means++ guarantee and $\alpha > 2$ is well-founded, since the optimal aggressiveness of exploration depends on the cluster geometry, the variance inequality, and outliers. So adjusting the value can improve both coverage and quality of the clustering [BNS23, BDW19].

This makes D^α sampling especially interesting for the present work, as the exponent α provides a direct mechanism for controlling the intensity of exploration of the selector. Smaller values result in uniform coverage, while larger values more strongly favor distant and potentially new regions of the embedding space.

3.4 Model Update Strategies in Iterative Loops

A useful distinction exists between nonparametric (instance-based) and parametric (optimization-based) learning methods. Nearest-neighbor methods are explicitly described as non-generalizing as they primarily store the training instances and answer queries by comparing new points with the stored examples. In an iterative loop, the respective protocol is followed. Newly labeled examples are simply added to the pool of stored labeled data, and the decision rule changes immediately because the reference set has changed. The disadvantage is that the computation shifts to inference time. The neighbor search becomes more computationally intensive as the size of the labeled dataset increases, which can increase the latency per round [PVG+11]. In contrast, logistic regression and multi-layer perceptron are parametric models whose updates require an explicit optimization step. Logistic regression in scikit-learn is typically treated as a batch estimator, which is retrained on the current label set from scratch rather than being

updated. Neural networks such as MLPs are also trained through iterative optimization (e.g., stochastic gradient methods). This means that updating the learning model is more computationally intensive than simply storing additional instances. Thus, different learning models can exhibit different practical behavior with the same label budget. KNN tends to update instantly but can become slower at prediction with more data, while logistic regression or multi-layer perceptron incurs training costs that can dominate iteration time, especially as the labeled set grows [PVG+11].

Logistic regression and multilayer perceptrons could be used as incrementally updated learning algorithms. However, warm-starts have significant challenges. Previous works show that warm-start models generalize less effectively than models trained from scratch and sometimes achieve lower test accuracy, especially in early rounds of an iterative loop with limited data available. One reason for this is that warm-start models tend to converge faster and adapt less effectively to new data [AA20, SYM+24]. A more pragmatic alternative in iterative labeling loops is regular retraining from scratch with the growing set of labels. This can yield better results than pure fine-tuning, especially in early phases where there are few labels available. Later, the difference usually diminishes [GDE24]. In the active learning setup, the approach is typically described as repeated selection, relabeling, and then retraining until the budget is exhausted [AAA+25]. It is not specifically evaluated in this study because the experiments do not reach a strict memory limit, which indicates that the used amount of data points stays low. Details of the design choice are discussed in the experimental design chapter 4.2.1. The main trade-off thus shifts from pure memory preservation to the latency of the whole pipeline, which later motivates the discussion about GPU acceleration.

As the number of newly acquired classes and data points rise, the growing amount is going to be an issue for the processing of the iterations. Each round increases the size of the whole setting, which can lead to an overload. To overcome this, one can drop all known data points and start from scratch each round. This might be helpful for fast processing but hinders accuracy because the pipeline keeps forgetting known classes, which is known as catastrophic forgetting [LSL+25]. It describes the problem that a model loses significant performance on the previously learned classes when incrementally learning new ones. To prevent this, the literature primarily focuses on two approaches. The first is the buffer approach in which a small set of representative examples from old classes is stored and reused during further learning. This refreshes the old knowledge [ZA23]. The second approach is forgetting without storage, where the model generates class-specific pseudo examples to maintain performance with previously learned classes, thus avoiding the need to store the original labeled images [JPA+22].

This view of the learning mechanics helps to interpret the results without having experimentally investigated catastrophic forgetting. The differences in the curves and latency can be explained as the resulting consequences of the cost of model updating, capacity, and inductive bias. This shifts the discussion to the central goal of the interactive loop, which is class discovery. The primary concern is not whether a learner stores knowledge indefinitely but rather how quickly and effectively the sampling and labeling process

operates.

3.5 GPU Acceleration

In interactive or frequently updated machine learning pipelines, latency is often as important as predictive quality. In iterative environments with human interaction, meaningful model updates require retraining, prediction, and candidate evaluation to be completed quickly enough to maintain responsiveness. GPU acceleration is therefore not only a technical optimization but also a methodological design choice. If iteration time becomes too long, the system can no longer support continuous exploration. Previous work comparing CPU-based scikit-learn implementations with GPU-accelerated alternatives such as RAPIDS with cuML shows that substantial speed improvements can often be achieved while maintaining comparable predictive performance, although careful optimization may be required to match the default CPU settings [MMIS22].

The main argument for GPU acceleration methods such as cuML from RAPIDS is pragmatic. Classical scikit-learn models are widely used and often achieve high accuracy, but purely CPU-based training and prediction can become a bottleneck, especially when repeated retraining and fast inference are required. Motylinski et al. [MMIS22] therefore evaluate the same model families using GPU-accelerated cuML. Their results show that the default cuML configuration initially performs worse than the corresponding CPU-based implementation, partly because the library was still under active development. However, after hyperparameter optimization, the accuracy became much more competitive. The authors compare several models to illustrate the familiar trade-off between predictive performance and computational time. kNN trains quickly but can be slow at prediction because inference requires scanning neighbors across the training set. Overall, the study suggests that GPU acceleration can substantially reduce retraining and inference time while preserving competitive accuracy. This makes more frequent model updates and faster real-time inference feasible.

Further, one can also rely on neural networks, which are not covered by RAPIDS but can be implemented with scikit-learn or PyTorch. The scikit-learn multilayer perceptron implementation is not intended for large-scale use and provides no GPU support, which can increase the wall-clock time for the retraining inside the human-in-the-loop cycles [PVG+11]. In contrast to this, PyTorch executes CUDA operations asynchronously by default, which enables high GPU utilization by parallelization and reduces the perceived iteration latency when training small MLPs repeatedly on the GPU [PGM+19]. To justify the performance claims, Abbas et al. investigate the extent to which surgery duration (DOS) and postoperative length of stay (LOS) can be predicted from preoperative patient characteristics using a large, real-world NSQIP dataset. They compared several conventional machine learning models with the simple neural network MLP and implemented it in both scikit-learn and PyTorch to highlight differences through implementation details. The preprocessing was performed entirely with scikit-learn, and the models were optimized using hyperparameter tuning. The results concerning the

learners showed that the MLPs do not automatically offer advantages over classical methods, although they can generalize well with sufficient data. At the same time, it becomes clear that in practice MLPs depend heavily on the training setup, tuning, and implementation that support the motivation to rely on dedicated deep learning frameworks. Therefore, the PyTorch implementation is also a suitable hardware backend for iterative settings, rather than considering small, non-large-scale MLP implementations as the standard solution [AML+22].

The reviewed literature suggests that effective interactive labeling systems require more than just a powerful classifier. They must combine an interpretable interaction design, appropriate candidate selection strategies, support for class determination in cases of ambiguity, and sufficient low latency for a smooth user experience. Previous works provide important building blocks for each of these aspects but deal with them in isolation. This motivates the present work, which investigates how data-driven and model-aware sampling techniques work in a single iterative loop that evaluates not only the prediction quality and recognition but also the computational power for the interactive use.

3.6 Embeddings and Caching

To ensure an efficient and reproducible experimental pipeline, image embeddings are calculated once per dataset and stored for later reuse. Without this preprocessing step, the iterative sampling loop would have to repeatedly process all the images through the feature extractor, which would result in computational overhead, especially with larger datasets, and it is not recommendable to do because it is not compatible with interactivity. Modern image encoders also typically require input images scaled to 224×224 pixels, which further increases preprocessing time and memory consumption. The reason for this is that CLIP encoders are trained on 224×224 pixel images, while DINO is also trained at fixed resolutions such as 224×224 and 416×416 [RKH+21, ODM+24]. Therefore, the extracted features are cached in files on disk, with the option to continue caching via memory mapping. This allows the embeddings to be calculated only once and then be reused in all subsequent iterations.

This makes the actual comparison of sampling strategies significantly more efficient because all following iterations access only pre-calculated feature vectors instead of raw images. The benefit of caching depends on the dataset and its size. For Fashion-MNIST, the embedding extraction is relatively fast and can be completed within seconds, while for CIFAR-100, it takes considerably longer. For TinyImageNet, however, repeated embedding extraction becomes extremely resource-intensive, which makes persistent storage of the embeddings essential for a practical workflow. This further aligns with the design of the experimental loop, in which the embedding function is treated as pre-calculated and cached before the interactive sampling iterations begin.

An important requirement of this pipeline is that the mapping between the images, indices, and cached embeddings remains consistent across all iterations. If the order of the embedding vectors changes between runs, the mapping between the feature vectors

and labels becomes unstable. This leads to unreliable training data and a significant reduction in the model performance. Therefore, the generation of embeddings and the indexing of the datasets after cache creation must remain unchanged. Randomness is only introduced in later stages of the pipeline, such as the sampling selection or model initialization, while the embedding space itself remains constant across all experiments.

The performance of this task depends, for the majority, on the quality of the extracted features. Recently foundation models in computer vision have shown that training supervised or self-supervised datasets, achieve features that generalize well to many tasks. DINO proves to be one of the strongest among the evaluated general-purpose vision foundation models. It is built upon the teacher-student principle, where two views are generated and both student and teacher receive an altered version. The student tries to match the teacher's output, whereas the teacher updates with the exponential moving average of the student weights. Fine-tuning of DINO offers an efficient way to obtain representations that reaches state-of-the-art performance [HLL+22, RKW+24].

As a second embedding approach, CLIP is used. CLIP is trained contrastively on extensive image-text pairs and learns a shared multimodal embedding space in which matching images and text are located close together, while mismatched pairs are further apart. Although only the image encoder is used, CLIP features are interesting because they capture the semantic similarities at a high level and have demonstrated strong zero-shot and transfer performance in a variety of visual tasks [RKH+21]. Comparing DINO and CLIP embeddings allows us to assess whether the behavior of the sampling strategies depends on the choice of visual representation space or on the selectors.

CHAPTER 4

Experiment Design

This chapter describes the experiment design for evaluating sampling strategies for visual interactive labeling under realistic interface conditions. Instead of requiring access to the entire pool of unlabeled elements, the experiment models a panel-based workflow. In each round the user only sees M candidate elements, selects k elements to label, and the model is updated before the next panel is displayed. This approach reflects the practical limitations of visual interfaces, which are limited display capacity and time, and allows for comparable sampling outcomes across different methods as each strategy operates with the same visibility and labeling budget.

The design underlines that sampling should not be viewed as a one-off optimization problem, but rather as part of a coupled iterative system controlled by the subsequent cycle. First the selection happens, then the labels are annotated, the model updates, and the cycle goes on with the next selection. The quality of a strategy can only be observed over several rounds. Consequently, the evaluation considers not only the accuracy of the final result but also the speed at which new classes are discovered and whether the selection process remains sufficiently efficient for interactive use, which is reflected in the selection latency.

4.1 Design Goal

The goal of this experiment is to fairly compare data-driven and model-aware sampling strategies in an interactive labeling loop. To this end, a standardized and reproducible protocol is defined in which all strategies are evaluated under identical visibility, label, and iteration restrictions.

The focus is not only on which strategy ultimately achieves the highest accuracy, but also on how quickly new classes are discovered and whether the selection decision can be made within a timeframe appropriate for interactive systems. In the simulation, the manual

annotations are replaced by the respective ground-truth labels of the selected data points, which allows the influence of the sampling strategy to be examined in isolation.

To make the experiment design explicit, independent and dependent variables are used for each research question. The purpose of this is to ensure that each experiment is connected to the corresponding research question and that the interpretation of the results is not confounded by uncontrolled changes in the protocol.

4.2 Interaction Protocol and Notation

For each dataset considered, a training set D_{train} and a test dataset D_{test} are used. The test dataset remains unchanged throughout the entire process and serves solely for evaluation purposes to prevent data leaks. The training dataset is initially treated as a completely unlabeled pool. Since this is a simulation, the user-assigned labels are replaced by the respective ground-truth label of the selected data points. This allows to investigate the influence of different sampling strategies without introducing additional confounding factors due to human labeling errors.

The process goes through T discrete interaction rounds. In each round t, the system does not have access to the entire unlabeled pool but only to a visible panel V_t. The panel size is controlled by the experimental parameter M, so that $|V_t| = M$. This panel models the limitations of real-world visual interfaces, where only a limited number of elements can be displayed and evaluated simultaneously per interaction. From this panel, the respective sampling strategy selects a subset $S_t \subseteq V_t$. The size of the subset is controlled by the label budget k, so that $|S_t| = k$. Thus, M and k define the interaction constraints investigated in this work, whereas V_t and S_t denote the round-dependent sets generated under these constraints. The newly annotated data points are then transferred to the cumulative labeled dataset and used to train the model.

- Panel Construction: A panel V_t of size M is constructed randomly from the current unlabeled pool. This panel represents the subset of items actually displayed to the user in round t. Displaying an entire dataset with more than 10 000 instances in a 2D projection simultaneously would result in severe visual clutter and overplotting, making the manual exploration infeasible. Therefore, M acts as visual capacity constraint. Since one research question focuses on the influence of this constraint, different values of M are evaluated across configurations. Furthermore, the random generation of this panel is an intentional design choice. Rather than filling the panel with uncertain or boundary points, randomly selected points preserve the global context and density distribution of the underlying data, which ensures that the user retains a reliable mental map of the actual space.
- Selection: A sampling strategy selects a subset S_t from the displayed candidates V_t to be labeled in this round. The size of this subset is determined by the label budget k. As with M, the value k is fixed within one configuration but varied across configurations to analyze the influence of the labeling budget.

- Annotation (simulated): The labeling step is simulated for all items in S_t. In the simulation, this corresponds to accessing the ground truth labels.
- Updating the Label Set: The newly labeled data points in round t are added to the cumulative set L_t. By default, they are removed from the unlabeled pool U_t so that a data point is only selected once. Alternative configurations may optionally allow re-selection. However, this is not part of the standard protocol and not used in this work. The result section provides more information regarding the parameters for the experiments.
- Model Update and Subsequent Round: After adding the new labels, a learning process can be updated through retraining or an incremental update. The updated model then provides signals that are relevant for model-aware selectors in the next round.

This protocol should therefore be understood as an explicitly coupled interaction cycle in which data points are presented, selected by the strategy, labeled, and then used to update the model. Once the model has learned from these newly labeled examples, the cycle begins anew with the presentation of further data points. The quality of a strategy is not only evident in a single selection but also in the dynamics over several rounds.

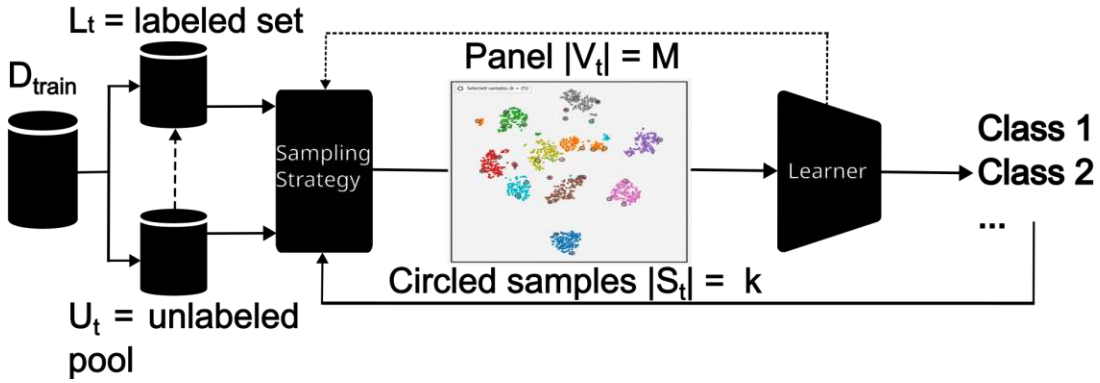


Figure 4.1: The iterative exploration loop is illustrated where the visualization, sampling and model learning interactively enhance the understanding of the data distribution over multiple rounds.

4.2.1 Panel-, Label-Budget and Retention Mechanism

Two budget parameters determine the interaction restrictions and are important for the analysis of the research question concerning panel and label budgets.

- M is the panel size and it indicates how many items are visible per round. It defines an upper limit of data points V_t, such that $|V_t| = M$, which are simultaneously visible to the user.

- k is the label budget per round and indicates how many items are actually selected from the panel M and labeled per round. The size of the selected subset S_t is determined, such that $S_t \subseteq V_t$ and $|S_t| = k$.

The distinction is important because M and k are experimental control parameters, whereas V_t and S_t are round-dependent sets derived from these parameters. Changing M changes the panel size for the selector, while changing k changes the number of labels acquired per interaction round.

Additionally, a parameter $r \in [0, 1]$ is introduced, which models the retention rate of the panel. This models the fact that visual interfaces in the real world are often not completely rebuilt but rather maintain a certain degree of visual continuity. The retained part of the previous panel contains only points that have not been selected for discovery. The input of r can be explained as follows:

- $r = 1$: The panel remains completely unchanged. This generates maximum stability and minimum novelty.
- $r = 0$: A completely new panel is generated each round, leading to maximum novelty and minimum stability.

In the experiments, r is set so that 25% of the data points are retained between two consecutive rounds. The parameter primarily serves to represent the interaction context more realistically and is not used as an independent comparison variable.

The following symbols complete the central sets and variables that define the interaction protocol.

D_{train} : Training dataset, initially treated as an unlabeled pool.

D_{test} : Fixed test dataset used only for evaluation.

T : Total number of interaction rounds.

$t \in \{1, \ldots, T\}$: Current interaction round.

M : Panel size, i.e., the number of candidate items visible in each round.

k : Label budget per round, i.e., the number of items selected for labeling in each round.

r : Panel retention rate, which controls how much of the current panel is kept for the next round.

Z : Fixed embedding representation of the training data used by the sampling strategies and learners.

U_t : Unlabeled pool after round t.

L_t : Cumulative labeled set after round t.

V_t : Displayed panel in round t, with $|V_t| = M$.

S_t : Queried subset selected for labeling in round t, with $S_t \subseteq V_t$ and $|S_t| = k$.

C_t : Set of discovered classes after round t.

f_t : Learner state after training on the labeled data available up to round t.

Y_t : Ground-truth labels assigned to the selected items in round t, with $Y_t = \{y_i : i \in S_t\}$.

$\hat{y}_t$: Predicted labels on the test set after round t.

τ_t^{sel} : Selection latency in round t.

τ_t^{train} : Training latency in round t.

τ_t^{pred} : Prediction latency in round t.

$\log[t]$: Evaluation record stored after round t.

The main idea behind this formalization is that all strategies operate under the same structural conditions with the same round horizon T, the same panel budget M, the same label budget k, and the same retention mechanism r. Across configurations, M and k are varied in order to analyze their influence on accuracy, class discovery, and selection latency. This prevents the comparison from being distorted by trivial advantages because one method might be capable of searching the entire pool, which is not needed for this experiment. Instead, it reflects the realistic constraint that selection decisions within the interaction system must be made locally, which means within the current view. At the same time, the notation allows for a precise description of the progress over time, including how the labeled set L_t, the unlabeled pool U_t, the discovered classes C_t, and the learner state f_t change from round to round.

Algorithm 4.1: Interactive Sampling Loop

Input: training dataset D_{train}; test set $(X_{\text{test}}, y_{\text{test}})$; embeddings Z;
panel size M, label budget k, retention rate r, rounds T;
panel policy BuildPanel$(\cdot)$;
selection strategy Select$(\cdot)$;
learner training procedure Train$(\cdot)$
Output: Per-round evaluation log $\log[1, \ldots, T]$

Initialize labeled set $L_0 \leftarrow \emptyset$, unlabeled pool $U_0 \leftarrow D_{\text{train}}$, discovered class set $C_0 \leftarrow \emptyset$, previous panel $V_0 \leftarrow \emptyset$;
Initialize learner state f_0 as unfitted;
for $t \leftarrow 1$ **to** T **do**
  $V_t \leftarrow \text{BuildPanel}(U_{t-1}, V_{t-1} \cap U_{t-1}, M, r)$;
  start_timer();
  $S_t \leftarrow \text{Select}(V_t, Z, f_{t-1}, k)$ with $S_t \subseteq V_t,\ |S_t| = k$;
  $\tau_t^{\text{sel}} \leftarrow$ stop_timer();
  $Y_t \leftarrow \{y_i : i \in S_t\}$;
  $L_t \leftarrow L_{t-1} \cup \{(i, y_i) : i \in S_t\}$;
  $U_t \leftarrow U_{t-1} \setminus S_t$;
  $C_t \leftarrow C_{t-1} \cup Y_t$;
  $L_t^{\text{train}} \leftarrow L_t$;
  start_timer();
  $f_t \leftarrow \text{Train}(L_t^{\text{train}})$;
  $\tau_t^{\text{train}} \leftarrow$ stop_timer();
  start_timer();
  $\hat{y}_t \leftarrow \text{Predict}(f_t, X_{\text{test}})$;
  $\tau_t^{\text{pred}} \leftarrow$ stop_timer();
  $\log[t] \leftarrow \text{EvaluateAndLog}(\hat{y}_t, y_{\text{test}}, V_t, S_t, L_t, C_t, \tau_t^{\text{sel}}, \tau_t^{\text{train}}, \tau_t^{\text{pred}})$;
end

The algorithm summarizes the iterative process of the experiments. In particular, it becomes clear that the selection decision is not made in isolation but forms the basis for the model update and thus also for the following rounds. The parameters M and k control the size of the visible panel and the number of selected labels, while V_t and S_t represent the panel and subset in each iteration.

The model comparison is embedded within the previously defined interaction framework. All sampling strategies and learning algorithms are evaluated using the same panel-based protocol to isolate the effects of the selector and the training model. The simulation replaces manual labeling with ground-truth labels for the selected samples, while the test dataset remains unchanged for evaluation. For comparability, all configurations use the same dataset partitioning, fixed cached embeddings, identical panel construction, the same number of rounds, and the same labeling budget per round. The embeddings

are calculated once and reused in all runs to ensure that the performance differences are due to the sampling strategy and not to changes in the feature representation. The specific configurations vary the panel size and label budget to analyze their impact on accuracy, class discovery, and selection latency. The memory limit is disabled in the main experiments, so each learner is trained on the entire cumulatively labeled dataset L_t. This avoids an additional effect of core set selection and focuses the comparison on the sampling strategies themselves.

The learner f_t has two roles in the simulated interaction loop. First, it provides the model state for the model-aware sampling methods. Selectors such as min-margin or disagreement sampling use the current predictions or uncertainty estimates of f_{t-1} to decide which samples from the visible panel V_t should be selected next. Second, the learner serves as a measure for the accumulated knowledge that would be gained by a human analyst in real-world settings. The metrics, which deliver useful insights, are accuracy, known-class accuracy, and class discovery over time.

Using the same learner for model-aware selection and for the evaluation is an intentional simplification. It ensures that the selected samples are evaluated with respect to the learning process that also guides model-aware selection. It does not imply that the learner is equivalent to a human analyst. In a visible analytics system, the understanding of a human may differ from the machine, and the model used for the interface guidance could also be different from the model used for quantitative evaluation.

4.3 Datasets

Three image datasets of varying complexity are used to evaluate the method, which are Fashion-MNIST, CIFAR-100, and TinyImageNet. This selection allows the behavior of the sampling strategies to be investigated on both a relatively simple grayscale dataset and more complex color image datasets with a significantly higher number of classes.

- **Fashion MNIST**: The dataset consists of Zalando's article images and contains 60 000 examples in train and 10 000 examples in test. Each image has the size of 28×28 and is a grayscale example for a total of 784 pixels in total. Each pixel is associated with lightness or darkness with pixel values between 0 and 255 and the respective meaning that higher numbers indicate dark values and lower values gray or even white shades. Each training and test example is assigned to one of the following labels [XRV17]:
  - 0 = T-shirt/top
  - 1 = Trousers
  - 2 = Pullover
  - 3 = Dress
  - 4 = Coat

- 5 = Sandal
- 6 = Shirt
- 7 = Sneaker
- 8 = Bag
- 9 = Ankle boot

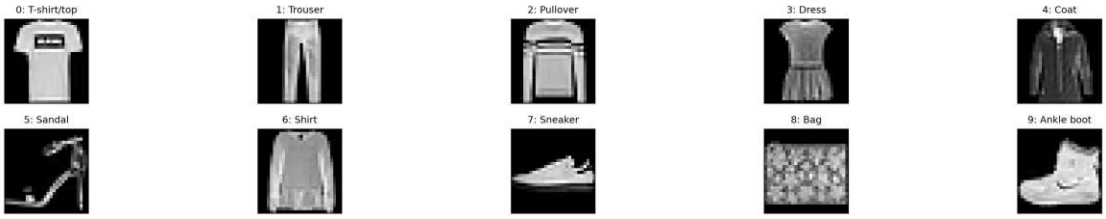


Figure 4.2: Randomly selected samples from Fashion-MNIST.

- **CIFAR-100**: The image dataset contains 60 000 32×32 color images across 100 classes, including 600 images each, which are grouped into 20 superclasses. There are 500 training images and 100 testing images per class, and each image comes with a fine label to which fine class it belongs and a coarse label to which superclass it belongs. 11 superclasses are about living organisms, which include mammals, fish, and insects. Then there are also common human-made items like vehicles, household furniture, and electrical devices. Moreover, outdoor environments and constructions are also contained in certain superclasses [Kri09]. Therefore, the classification difficulty is increased, and it needs to be seen whether the selectors are able to find all classes for discovery.
- **TinyImageNet**: TinyImageNet comprises a total of 120 000 annotated images assigned to 200 different object categories. 100 000 images are for training, 10 000 are for testing, and 10 000 are for validation. Each of the 200 categories consists of 500 training images and 50 validation and 50 test images, which are all uniformly scaled down to 64×64 pixels. The training and validation set from TinyImageNet are used because the test set is for label prediction, which means that it does not provide any label, and therefore it is not usable for this setup [LY15].

The datasets are balanced in train and test, which should therefore always lead to total or large class discovery depending on the setting. However, in most real-world scenarios, the datasets are not perfectly balanced, which makes class discovery and rare class finding much more difficult than in an ideal world.

All datasets follow a fixed and consistent partitioning protocol to make the results comparable across different sampling strategies. For each dataset, the official training and test partitions are used, if they are available. If a dataset does not provide a standard split, a stratified split with a fixed starting value is created and kept constant across all

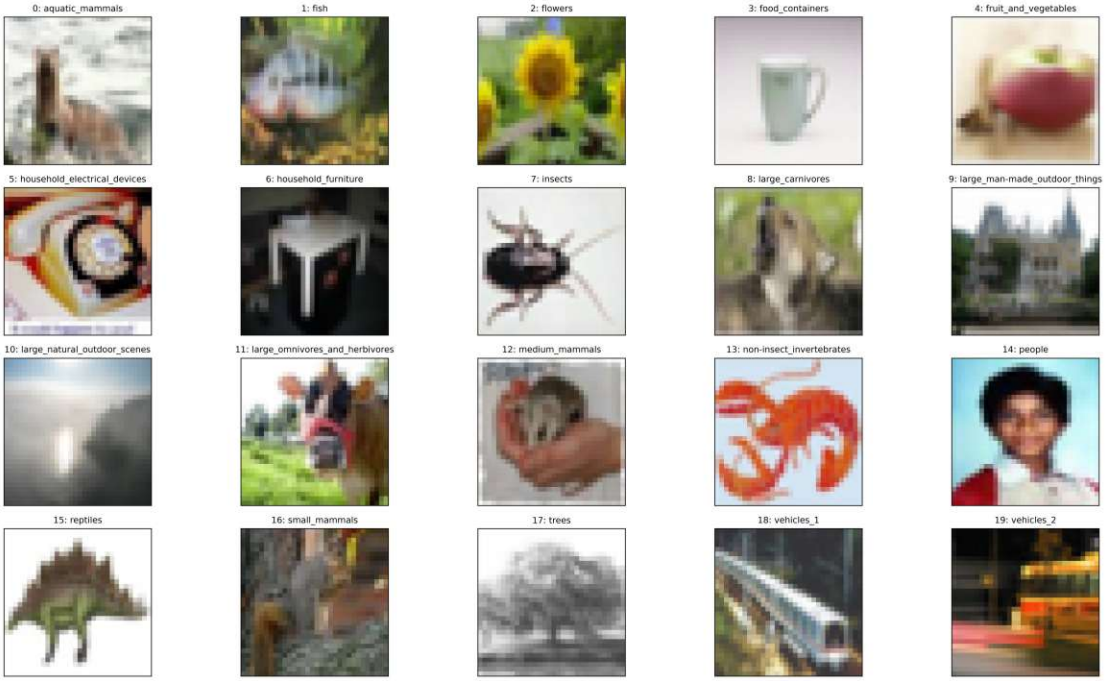


Figure 4.3: Randomly selected samples from CIFAR-100.

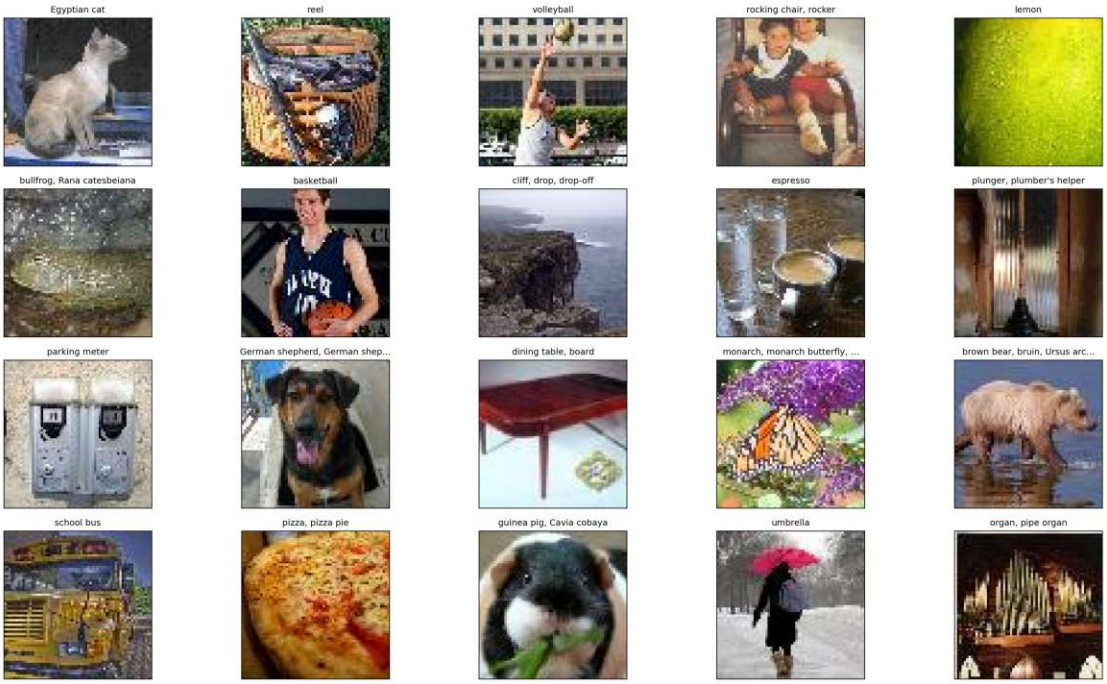


Figure 4.4: Extract of randomly chosen samples from TinyImageNet.

methods and runs. The initial unlabeled pool is drawn from the training portion, while the evaluation is performed exclusively on the held-out test set to avoid data leakage.

4.3.1 Imbalance handling

In addition to the balanced standard partitions, the sampling strategies are also evaluated under artificially generated class imbalances. This setting is intended to reflect real-world application scenarios where relevant classes are rare and therefore may not be captured by random or purely frequency-based selection strategies.

To generate the imbalance, the training dataset is undersampled for a subset of the classes, while the remaining classes stay unchanged. The selection of the reduced classes and the actual undersampling are controlled and seed-based to ensure that identical unbalanced training datasets remain reproducible across different runs. The test dataset remains unchanged to be able to continuously measure generalization on a fixed reference basis.

The imbalance is generated after the computation of the full embeddings by filtering the corresponding embedding vectors based on the reduced training indices. This ensures that the feature extraction and subsequent comparability between the balanced and unbalanced configurations remain consistent.

When the imbalance argument is enabled, only 20% of each class are retained for the minority classes, while all other classes are fully preserved. For Fashion-MNIST, the classes t-shirt/top, pullover, coat, shirt, and bag are undersampled, which is half of the classes. For CIFAR-100, classes 0 through 49 are undersampled, and for TinyImageNet, classes 0 through 99 are undersampled, which corresponds to half of the classes in each dataset. The list of classes that are undersampled can be found in Appendix, for CIFAR-100 in Table A.1 and TinyImageNet in Table A.2.

4.4 Selection Strategies

Two families are investigated in this thesis:

- Data-driven selectors: These techniques use only the geometry or statistic, like distance, density, and diversity, within the panel. They do not need any probabilities from the model. They could be relevant for early stages, solving the cold start problem and discovery, but they can be more computationally intensive.
- Model-aware selectors: These use the state of the actual model, which is uncertainty, margin, agreement, or disagreement. They can switch between exploration and optimization.

The separation is included and important because it is a main goal to investigate how those different families behave. Further, it is operationalized in the design as a comparison under an identical protocol.

Seven different strategies are evaluated within this work. Three each from data-driven and model-aware methods and the random baseline.

- Random sampling requires only the current candidate set from the panel V_t as input and uses neither distances nor model posteriors. Its objective is a uniform random selection S_t from V_t, which makes it a neutral reference that does not actively optimize any metric. The complexity is very low, which makes it an important reference for selection latency.
- Farthest selection requires all points in the panel. Its goal is a diversity-maximizing selection. S_t should contain points that are as far apart as possible in the embedding space and cover it well. This behavior might be beneficial for fast discovery in the early stages, while it will stay too explorative in the later stages.
- The density-based method requires all data points from the panel. The objective is to favor points that are in dense regions to achieve representative coverage of the frequent patterns and learn solid core regions. As it depends on the complexity of the density estimation, it might be more expensive to compute.
- D^α-sampling requires all data points from the panel. The hyperparameter α explicitly controls the trade-off between diversity and density. The aim is to combine the diversity component with a density component to balance exploration and exploitation. This is depending on the α. More explorative α means accelerated discovery in early rounds, while a more exploitative α stabilizes accuracy in later rounds.
- Min-margin sampling requires the model posteriors $p_\theta(y \mid x)$ for all points in the panel. The panel is only used indirectly via the learner. The objective is to select points where the current model is uncertain in order to rapidly improve the classifier by labeling decision boundaries. For margin sampling the lowest margin for the most uncertain points are used to increase the learning.
- Agreement sampling needs the model posterior $p_\theta(y \mid x)$ for the points in the panel where the maximum class probability $\max_y p_\theta(y \mid x)$ is used as a confidence score. The goal is to select the most confident data points with the maximum model confidence in order to collect stable, clean labels and stabilize the model, which can be particularly helpful with noisy labels.
- Disagreement sampling also requires the model posterior $p_\theta(y \mid x)$ for all points in the panel. In this concrete implementation, the entropy $H(P)$ of the posterior distribution is calculated as an uncertainty score. The aim is to prioritize difficult points where the model is uncertain, which is shown by the high entropy. These points are viewed as particularly informative because they point to ambiguous or previously insufficiently learned regions of the feature space.

4.4.1 Exploratory hybrid selector

In addition to the core selectors, two exploratory hybrid strategies are implemented as two-stage procedures. In both variants, the larger candidate subset is selected first using a data-driven criterion. From this subset, the final batch is selected using min-margin sampling. The first hybrid variant uses farthest preselection, while the second variant uses density-based preselection. Both hybrid strategies combine the preselection step with a model-aware refinement step. The min-margin stage prioritizes uncertain samples within the preselected candidate pool, thereby concentrating the labeling budget on informative candidates. The purpose of the hybrid selectors is not only to improve predictive performance, but also to investigate trade-offs between class discovery, accuracy, and selection latency. Because this selector was introduced late in the thesis, it could not be evaluated against the entire experimental use case of datasets, algorithms, and parameter settings. Instead, it is presented as an exploratory extension in the imbalanced CIFAR-100 setting, where the motivation to combine representative coverage with uncertainty-based refinement is high to produce even more informative results.

4.4.2 Sample selection

The actual selection of data points to be labeled always takes place within the underlying panel size but under different information conditions and it is restricted to the current visible panel V_t. Data-driven methods can be used as early as the first round, since they are based solely on the structure of the visible panel. Model-aware methods, on the other hand, first require a trained model to calculate the confidence or uncertainty values. Therefore, the process for these strategies begins with a random selection in the first round. Only from the second round onward can the selection be done by model-aware selectors due to the need for model-aware values.

Strategy type	t_0	t_1	...	t_r
Random	R	R	...	R
Data-driven	D	D	...	D
Model-aware	R	M	...	M
Hybrid	D	D	...	M

Table 4.1: Information used by each sampling strategy over the iterative labeling rounds.

The information from the table shows that R denotes random selection, D denotes data-driven selection based on the visible panel, and M denotes model-aware selection based on model-aware uncertainty or confidence values. Model-aware strategies cannot be applied in the first round because no model has been trained yet. This is why random is the initial selection. Data-driven strategies can be applied from the beginning. The hybrid version can combine both principles. First, it starts with data-driven selection but later switches to model-aware selection to refine the model.

4.4.3 Learners

The learner guides the pipeline by modeling the labels of the selected samples after each iteration. In doing so, it provides the system with how the acquired classes are separated and which regions are already covered well. When the sampling selector selects new points and their labels become known, these labels are transferred to the training memory and the learner is refitted. In this way, an initially unstructured pool is gradually transformed into an increasingly structured representation in which known classes can be recognized more reliably and new classes get visible as gaps or regions of uncertainty.

The learner has two roles. First, it is used as predictive model whose accuracy is evaluated after each round. This includes the overall test accuracy as well as the seen-class performance curve. Second, for the model-aware strategies, the learner provides the uncertainty or confidence signal that guide the selection of informative candidates. These signals are derived from posterior probabilities or uncertainty scores. These can be either points with high entropy (strong disagreement), a small margin between the top classes, or confidence (high agreement).

- kNN: The k-nearest neighbors algorithm serves as an obvious basic learning algorithm within the panel. The method requires no complex parameterization and is particularly suitable when the feature representation already exhibits a meaningful local neighborhood structure. Furthermore, uncertainty measures for model-aware selectors can be derived from the neighborhood relationships. A RAPIDS-based cuKNN model is used on the GPU, while the KNeighborsClassifier from scikit-learn is used on the CPU. For both learners, the neighborhood search is based on the cosine metric. The number of neighbors is defined in the model by the value 5. The prediction probabilities are not taken directly from a standard posterior distribution but are calculated from the neighbors using temperature-weighted aggregation. This leads to a more consistent confidence signal for the selection and evaluation steps.
- Logistic regression: LR is a linear, probabilistic learning method and serves as a more parameterized alternative to kNN. It provides directly interpretable a posteriori class probabilities and is therefore particularly well-suited for uncertainty-based selection criteria. Compared to kNN, its performance depends more heavily on the linear separability within the panel. On the GPU, culr from cuML is used, and on the CPU, logistic regression is used from scikit-learn. Both variants have set the maximum iterations to 2000 and the regularization parameter to 5. The class probabilities are obtained via *predict_proba*, and *sample_weight* is supported, which allows repeatedly selected classes to be weighted more heavily if needed.
- Multilayer perceptron: MLP complements the comparison with a nonlinear model that can represent more complex decision boundaries within the panel. This allows for investigating whether greater model flexibility not only improves accuracy but also provides further informative signals for model-based selectors. It is expected

that MLP requires a higher training effort compared to kNN and logistic regression. For the GPU execution, a customized TorchMLPClassifier based on PyTorch is used, while for the CPU execution, an MLPClassifier from scikit-learn is deployed. The GPU-based MLP consists of an input layer, a hidden layer with 256 neurons, LayerNorm, and a ReLU activation function. Training is performed using Adam, with a learning rate of 1e-3, a *weight_decay* of 1e-4, and a batch size of 256. New classes are integrated by dynamically extending the output head. The CPU-based MLP version is similar, as it uses a hidden layer of 256 neurons. The batch size is the same, and the optimizer is also Adam. The learning rate is 1e-3, and the maximum number of iterations is limited to 100. Further, early stopping is disabled.

To ensure comparability of the configurations, the learning models are used with fixed hyperparameters and under identical training conditions. After each round, the respective model is adapted to the currently labeled dataset. In all experiments, the learner is updated after each round on the current labeled pool. Warm-start, replay buffers, and catastrophic forgetting are not evaluated as separate strategies in this thesis. The experiments do not operate under a strict memory limit, which means that repeated retraining on the growing labeled pool is used as the main design choice.

4.5 Evaluation Methods

4.5.1 Metrics

To record the outcomes, the following metrics are used for evaluation. The metrics are also logged in a CSV and JSON file by the code itself.

- Test accuracy $Acc_{test}(t)$ is the classic accuracy on $\mathcal{D}_{test}$ after T rounds using kNN, logistic regression and multilayer perceptron.
- The average test accuracy is derived from the accuracies of all iterations within a sampling selector.
- Known-class accuracy measures the test accuracy restricted to the classes that have already been discovered up to round t. It helps to distinguish whether low overall accuracy is caused by poor classification of already discovered classes or by still undiscovered classes.
- Class Discovery measures how fast new classes are found over the whole iterative process. A steep curve means good exploration, while stagnation means less exploration.
  - C_t is the number of discovered classes after round t.
  - $Disc(t)$ is the discovery fraction, which is calculated by the amount of classes divided by all classes $\frac{|C_t|}{|C|}$.

  - Discovery Area under Curve (AUC) is the area under $Disc(t)$ over rounds as a compact measure of whether the discovery happened faster or slower.

- Selection latency measures the time a sampling strategy needs to determine the selected subset S_t from the current panel V_t. Since it is an interactive system, the practical suitability of a method is assessed not only based on its quality but also on its response time. Strategies whose selection time exceeds the interactive target range for the panel size M are not suitable for the intended application.

- GPU speedup measures the runtime improvement of the GPU-accelerated implementation compared to the CPU-based implementation. This metric is used to evaluate whether the runtime improves without changing the predictive quality. It is calculated as follows:

$$Speedup = \frac{\tau_{\text{CPU}}}{\tau_{\text{GPU}}}.$$

Both round-to-round trends and aggregated means across multiple samples are reported and logged. Therefore, 10 runs have to be conducted in order to observe these measures.

4.5.2 Statistical comparison

In order to assess whether the observed differences between sampling strategies are systematic across repeated runs, Friedman tests [Fri37] are applied separately for each dataset and learner. The null hypothesis states that all sampling strategies perform equally with respect to average test accuracy. If the p-value is below $\alpha = 0.05$, the result is considered statistically significant.

4.5.3 Qualitative analysis and monitoring

In addition to the quantitative metrics, two-dimensional projections of the data points are used for the qualitative analysis of the sampling behavior. These visualizations, which can be found in Appendix A.3, are not intended for the actual selection decision but solely for the interpretation of the iterative process and are therefore supplementary. They enable a visual understanding of the location of the considered visible points, the distribution of the selected samples, and the structure of potential clusters.

The methods used in this comparison are PCA, UMAP and t-SNE. While PCA represents linear structures only, UMAP and t-SNE are much better suited for representing nonlinear relationships in the panel. The visualization results should therefore be understood as aids to interpretation and not direct proof of global separability of the classes.

4.5.4 Repetitions and randomness control

To ensure reproducibility and comparable results, each experimental configuration is executed with ten independent, randomized seed values. Subsequently, the mean and standard deviation of all reported metrics are calculated over these seed values. This

includes test accuracy, selection latency, the discovery curve, and their aggregated discovery AUC. The following parts are separate in order to guarantee and control the process to make the experiments reproducible and reliable.

- Data Preparation: How data is divided or shuffled.
- Selection Process: How options are gathered and presented.
- Final decisions: How ties are resolved in the selection process.
- Model training: The inherent randomness involved in training a machine learning model.

The embeddings are treated as deterministic features and, as noted earlier, are cached. The cache stores the embedding vectors and the original dataset indices to guarantee that every vector is consistently mapped to the same image across runs. Consequently, only the selection and the learning components vary with the random seed, while the panel size remains fixed. This prevents artificial accuracy losses due to index misalignment or non-comparable feature representation.

The experimental protocol restricts all selectors to the visible panel V_t of size M. This increases the comparability between the strategies in this setting, but it also constrains the maximum potential of each selector. A selector can only choose the best samples among the currently visible candidates and cannot access the full unlabeled pool. Therefore, the performance of the measures reflects the suitability of a strategy for the proposed panel analysis scenario rather than its theoretical optimum in an unconstrained pool-based active learning setting. The random construction of V_t also introduces stochastic variation. To reduce the influence of the individual random panels, all configurations are repeated with multiple independent seeds, and the results report the means and standard deviation. The panel policy remains a design choice of the simulation and should be considered when interpreting the results.

4.6 Implementation

In this section, the technological basis of the proposed system is described. It summarizes the software stack used for implementation and experiments and explains the role of GPU acceleration in improving the efficiency of the iterative learning pipeline.

4.6.1 Technology stack

The entire experimental pipeline is implemented in Python. It includes loading and preprocessing the datasets, extracting and storing the embeddings, implementing the sampling strategies, training the learning models, and logging and visualizing the results. For data processing and basic components, NumPy and scikit-learn are primarily used. Image processing and embedding extraction are based on PyTorch, torchvision, and

timm. Matplotlib is used for visualization and displaying metric trends. GPU-accelerated learning and selection methods are implemented with RAPIDS, using cuML and PyTorch, respectively. A complete overview of all command line interface (CLI) options can be found in Appendix A.2.

4.6.2 GPU Acceleration

GPU acceleration is a key element of the implementation because the experiments under consideration consist of many successive selection, training, and evaluation steps. Without acceleration, the runtime per round would quickly become a bottleneck, especially with larger datasets due to the increased computational load. In the present pipeline, the GPU is utilized at multiple levels. First, the image embeddings are extracted using PyTorch and DINO on CUDA-based devices. Then, the selected learning methods, kNN, LR, and MLP, are trained and evaluated using GPU acceleration via cuML and PyTorch. Finally, the computationally intensive tensor operations, such as similarity calculations or determining the top k neighborhoods, are also performed on the GPU. This distribution is particularly relevant for the overall system because the round duration includes not only the model training but also sample selection and repeated predictions. Therefore, GPU acceleration not only reduces the training time of individual models but also improves the entire iterative loop.

CHAPTER 5

Results

This section presents the results of the experimental setup introduced in the previous chapter. The main analyses focus on several aspects of the iterative loop, including the impact of the GPU-accelerated components on the latency of the pipeline, the effect of the panel sizes M and the label budgets k, the performance on the balanced versus imbalanced datasets, and the comparative accuracies of the different sampling techniques. In most subsections, only the D^α variant with $\alpha = 4$ is reported. This keeps the tables and plots focused on the differences, while the influence of other α values is examined separately in the specific section 5.2.1. The following experiment settings 5.1 are used. Unless stated otherwise, the following experiment settings are used.

$$M = 1000,\ t = 50,\ k = 25, r = 0.25,\ \text{GPU-accelerated, and DINO embeddings.}$$

Deviations from this configuration are explicitly stated in the relevant sections. Since r remains constant in all experiments, it is not repeated in each subsection.

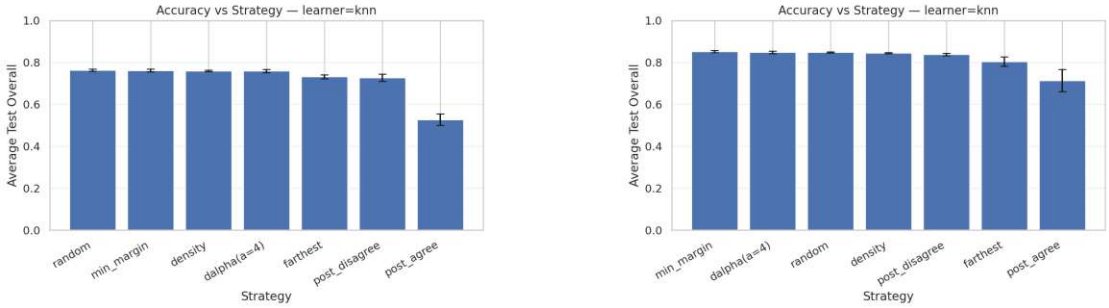
5.1 Preliminary Embedding Comparison

In this section, the influence of the embedding method on the performance of the iterative loop is examined. The comparison is first carried out on FMNIST and then on CIFAR-100 to include both a simpler and a more complex dataset. Based on these observations, one embedding method is chosen for the main experiments. The independent variables are the datasets, the panel size, the label budget, and the embedding method, with CLIP and DINO compared under the identical experimental setup. The dependent variables are the average accuracy on the test set, the standard deviation across runs, and the comparable behavior of the sampling strategies between both datasets.

As shown in Figure 5.1, CLIP performs only slightly worse than DINO on FMNIST. Since FMNIST is a comparatively simple and well-separable dataset, this leads to the outcome

Component	Tested configurations, with default in bold
Datasets	**FMNIST**, **CIFAR-100**, **TinyImageNet**
Dataset setting	**Balanced**, imbalanced
Embeddings	CLIP, **DINO**
Panel size M	500, **1000**, 2500, 5000
Label budget k	10, **25**, 50
Rounds T	**50**
Memory ratio r	**0.25**
Learners	**kNN**, **LR**, **MLP**
Selectors	**Random**, **agreement**, **disagreement**, **min-margin**, **density**, **farthest**, $\mathbf{D}^{\alpha}$, hybrid
D^{α} values	-2, -1, 1, 2, 3, **4**, 5, 6, 7, 8, 9
Hardware	CPU, **GPU**
kNN implementation	scikit-learn, **cuML**
Data-driven implementation	CPU-based, **GPU/PyTorch-based**

Table 5.1: Overview of the tested configurations. Default settings are shown in bold.

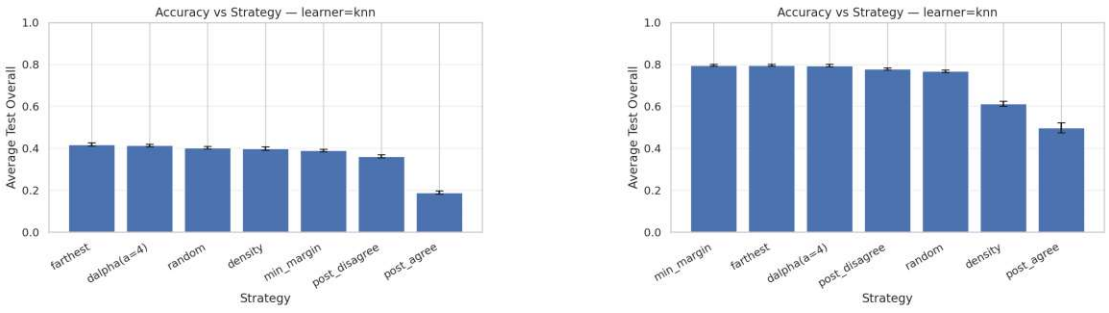


(a) CLIP (b) DINO

Figure 5.1: CLIP and DINO comparison on FMNIST.

that both embedding methods provide representations that are sufficiently informative for the selection strategies. Although the ranking changes slightly between CLIP and DINO, the overall differences remain small. This suggests that on a simpler dataset the choice of the embedding model is not too critical and has only limited effect on the final accuracy.

Since the embeddings do not have much of an impact on the trivial dataset, it is reasonable to try it also with the CIFAR-100 in order to see whether the embeddings lead to any differences. The comparison in CIFAR-100 gives a different picture. In Figure 5.2, the overall accuracy drops considerably in contrast to FMNIST, and the gap between DINO and CLIP becomes much more noticeable. DINO embeddings consistently support higher accuracies and lower standard deviation, which indicates that they provide a more separable and informative representation for this more complex dataset. While some data-driven selectors remain competitive with CLIP, the overall performance clearly



(a) CLIP (b) DINO

Figure 5.2: CLIP and DINO comparison on CIFAR-100.

benefits from the strength of DINO. This suggests that the embedding quality becomes a bottleneck as the dataset complexity increases.

This leads to the assumption that when the dataset is more trivial, the choice of embedding is less critical. When the data is more complex, it becomes a bottleneck because the better the embeddings, the higher the accuracy that is achievable. For simple datasets such as FMNIST, both embedding methods are strong and lead to similar outcomes, but for more complex datasets like CIFAR-100, the quality becomes more important. Further, a larger performance gap between the selectors is seen in CLIP compared to DINO. To isolate the effect of the proposed selection strategies and ensure the comparability of the experiments, the embedding method is kept fixed, and only the results for DINO embeddings are reported. A comprehensive evaluation of all combinations of selection strategies and embedding models would require a significantly larger experimental matrix and is reserved for future work because it exceeds the scope of this thesis.

5.2 Accuracy

In this section, the sampling strategies are compared with respect to their average test accuracy and under the default configuration mentioned at the beginning of this chapter. It answers one part of **RQ1**. The independent variables are the dataset, learner update mechanism, and the selection strategy, while the dependent variables are average accuracy, discovery AUC, and the average selection latency. The goal is to identify which combination of learner and selector achieves the best overall performance while remaining suitable for an iterative setting. Table 5.2 provides insight into the best-performing strategies for each learner and dataset under balanced conditions.

5.2.1 Accuracy under balanced conditions

In iterative learning loops, not only is the final model quality relevant, but also how responsive the system remains during the iterations. Each round includes selection, labeling, and a learner update. Depending on the learner update mechanism, different

runtime and quality profiles emerge. In this section, the three learners from the experiment design are compared to each other. The independent variable is the learner, which is either kNN, logistic regression, or MLP. The panel size, the label budget, selection strategy, and dataset are also considered to factor in because the best selector can switch across the learners and datasets. The dependent variables are average accuracy, discovery AUC, and average selection latency.

Dataset	Strategy	Learner	Disc. AUC	Avg. Accuracy	Avg. Latency
cifar100	min-margin	knn	0.938 ± 0.003	0.796 ± 0.004	15.828ms ± 1.932
cifar100	min-margin	lr	0.932 ± 0.005	0.794 ± 0.006	10.427ms ± 1.248
cifar100	min-margin	mlp	0.935 ± 0.003	0.799 ± 0.004	8.801ms ± 1.365
fmnist	min-margin	knn	0.978 ± 0.003	0.852 ± 0.005	14.922ms ± 0.835
fmnist	min-margin	lr	0.979 ± 0.002	0.880 ± 0.003	10.146ms ± 1.054
fmnist	min-margin	mlp	0.979 ± 0.002	0.883 ± 0.003	8.677ms ± 1.204
tinyimagenet	farthest	knn	0.900 ± 0.004	0.711 ± 0.004	68.903ms ± 4.738
tinyimagenet	min-margin	lr	0.876 ± 0.007	0.666 ± 0.009	13.695ms ± 1.305
tinyimagenet	farthest	mlp	0.900 ± 0.004	0.674 ± 0.005	76.634ms ± 2.797

Table 5.2: Best Strategies according to average accuracy for balanced datasets.

Table 5.2 for balanced data shows the configurations with the highest average accuracy (mean ± standard deviation across multiple runs) as well as their discovery AUC and average selection latency. No single learner-selector combination dominates across all datasets and settings. On the balanced FMNIST and the balanced CIFAR-100, one strategy remains consistently the best-performing across all learners, whereas on the TinyImageNet dataset, the strongest strategy depends on both the learner and the data. At the same time, it can be seen that the differences between the best configuration and several alternatives are often relatively small. Therefore, the top entries should not be interpreted as absolute winners, but rather as the ones with the most favorable trade-offs within a narrow performance range. This indicates that the pipeline is comparatively robust and the final choice of learner and selector can reasonably not only be guided by accuracy but also latency, hardware constraints, and the desired level of interactivity.

For further analysis, Friedman tests for each dataset-learner combination are used to check whether these observed differences are systematic across several runs. The null hypothesis is that all sampling strategies perform equally with respect to their strength. The Friedman tests show importance if the p-value is below $\alpha = 0.05$.

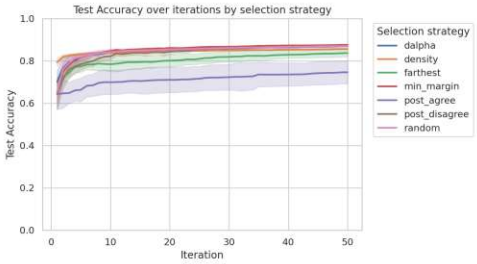
Table 5.3 reports the Friedman test results for the balanced setting. The test is done separately for each dataset and learner using the average test accuracy as the dependent variable. In all cases, the test is statistically significant, which suggests to the assumption that the choice of sampling strategy has a measurable effect on the average accuracy. At the same time, the results from Table 5.2 show that the values are often close to each other.

To better understand these differences, it is useful to consider the accuracies' trajectories

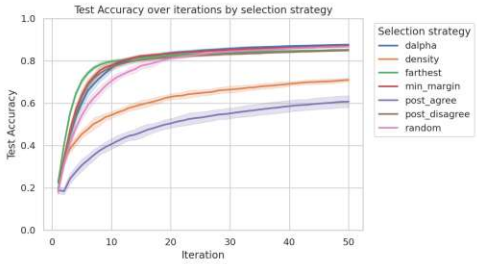
Dataset	Learner	Friedman χ^2	p-value	Significant
CIFAR-100	kNN	55.629	3.46×10^{-10}	Yes
CIFAR-100	LR	58.586	8.72×10^{-11}	Yes
CIFAR-100	MLP	56.914	1.90×10^{-10}	Yes
FMNIST	kNN	50.657	3.47×10^{-9}	Yes
FMNIST	LR	56.014	2.89×10^{-10}	Yes
FMNIST	MLP	57.686	1.33×10^{-10}	Yes
TinyImageNet	kNN	56.529	2.28×10^{-10}	Yes
TinyImageNet	LR	59.229	6.46×10^{-11}	Yes
TinyImageNet	MLP	57.086	1.76×10^{-10}	Yes

Table 5.3: Friedman test results for average test accuracy under balanced conditions. The tests compare the seven sampling strategies separately for each dataset and learner over ten repeated runs.

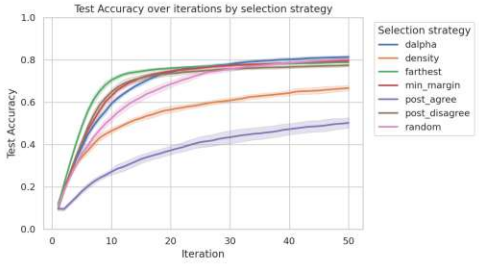
over rounds rather than only the final averages. This might be a good indicator of which selection strategy shows fast convergence or long-term exploitation, which strategy is superior, and which ones require more rounds to stabilize. Therefore, it might indicate whether potential hybrid measures could be taken into consideration to achieve even better results. Figure 5.3 shows how the selector's accuracy on kNN behaves over the iteration.



(a) FMNIST



(b) CIFAR-100



(a) TinyImageNet

Figure 5.3: Balanced accuracies for FMNIST, CIFAR-100, and TinyImageNet.

Figure 5.3 shows the development of the average accuracies over the iterative rounds for kNN on FMNIST, CIFAR-100, and TinyImageNet. On FMNIST, density starts at a relatively high level, but is overtaken by other strategies before round 10. Min-margin achieves the strongest performance throughout later rounds. On CIFAR-100, the farthest selector starts as best but is overtaken just after round 10, and min-margin again achieves the best accuracy. With the exception of density and agreement, all selection strategies are close together and achieve similar performance under the same configuration. On TinyImageNet, farthest selection has the lead, but it is later overtaken by D^α sampling. Five out of seven selectors achieve similar performances, while density and agreement are clearly behind.

D^α values

For the D^α sampling, the independent variables are the datasets, the panel size, the label budget and the exponent α of the respective strategy. The dependent variables are the same as before with average accuracy, discovery AUC, and average selection latency. All of the other parameters are kept fixed according to the default configuration, and the analysis is restricted to the kNN learner. By evaluating different α values, it seems that depending on the setup and the dataset, the optimal α can vary, which is expected as it is already explained in Bamas et al. [BNS23]. The value $\alpha = 0$ is not checked because it is equal to uniform random selection, which is already done by the random selector.

Since additional experiments with higher α values were introduced late in the thesis, they are not evaluated against the whole matrix of algorithms. Instead, an exploratory extension is reported for the kNN algorithm, as this setting most directly reflects the geometric effect of the D^α selector in the panel size.

α	Discovery AUC	Average Accuracy	Avg Selection Latency
-2	0.978 ± 0.002	0.836 ± 0.006	$59.292\ \text{ms} \pm 2.978$
-1	0.978 ± 0.001	0.843 ± 0.003	$58.981\ \text{ms} \pm 3.528$
1	0.979 ± 0.001	0.849 ± 0.005	$63.685\ \text{ms} \pm 2.271$
2	0.979 ± 0.001	0.848 ± 0.004	$57.448\ \text{ms} \pm 1.655$
3	0.980 ± 0.000	$\mathbf{0.850 \pm 0.005}$	$57.997\ \text{ms} \pm 3.017$
4	0.980 ± 0.000	0.848 ± 0.006	$57.817\ \text{ms} \pm 4.107$
5	0.979 ± 0.001	0.848 ± 0.004	$63.333\ \text{ms} \pm 4.967$
6	0.979 ± 0.002	0.848 ± 0.003	$63.118\ \text{ms} \pm 2.956$
7	0.979 ± 0.001	0.849 ± 0.004	$62.847\ \text{ms} \pm 4.377$
8	0.980 ± 0.000	0.845 ± 0.005	$61.234\ \text{ms} \pm 4.197$
9	0.979 ± 0.001	0.837 ± 0.007	$62.148\ \text{ms} \pm 4.172$

Table 5.4: Balanced FMNIST results for the kNN learner with extended D^α values.

With balanced FMNIST, the influence of α remains limited across the entire range. The AUC of the class discovery is very stable for all tested values. Average accuracy improves

from the negative α values to a local maximum at $\alpha = 3$, but the differences between the values 0 and 7 remain small. This suggests that FMNIST is not very sensitive to subtle changes in the exploration strength of the D^α sampling. Latency remains within a relatively narrow range, although higher values are observed for larger α settings.

α	Discovery AUC	Average Accuracy	Avg Selection Latency
-2	0.896 ± 0.011	0.736 ± 0.010	53.456 ms $\pm$ 2.791
-1	0.911 ± 0.006	0.756 ± 0.005	52.773 ms $\pm$ 2.085
1	0.925 ± 0.005	0.782 ± 0.005	54.756 ms $\pm$ 1.117
2	0.927 ± 0.006	0.785 ± 0.007	52.709 ms $\pm$ 3.287
3	0.932 ± 0.004	0.793 ± 0.005	52.941 ms $\pm$ 2.713
4	0.933 ± 0.002	0.795 ± 0.006	56.087 ms $\pm$ 7.431
5	0.935 ± 0.004	0.799 ± 0.006	56.651 ms $\pm$ 3.790
6	0.936 ± 0.004	0.799 ± 0.005	54.878 ms $\pm$ 3.877
7	0.935 ± 0.004	0.801 ± 0.005	55.241 ms $\pm$ 3.046
8	0.937 ± 0.005	$\mathbf{0.803 \pm 0.004}$	56.638 ms $\pm$ 2.602
9	0.938 ± 0.003	0.802 ± 0.005	58.457 ms $\pm$ 2.388

Table 5.5: Balanced CIFAR-100 results for kNN with extended D^α values.

For the balanced CIFAR-100, the effect of α is more concrete than for FMNIST. Both the discovery AUC and the average accuracy improve steadily from negative to a larger positive range. Greater weighting of distance is more beneficial for this complex dataset. The performance increases until $\alpha = 8$, while the highest discovery rate is at $\alpha = 9$. However, the accuracy is diminishingly smaller than the one from $\alpha = 8$. The selection latency remains largely comparable across the tested range, while the larger settings are slower.

α	Discovery AUC	Average Accuracy	Avg Selection Latency
-2	0.793 ± 0.013	0.607 ± 0.010	69.136 ms $\pm$ 3.684
-1	0.821 ± 0.012	0.636 ± 0.008	72.168 ms $\pm$ 9.073
1	0.851 ± 0.008	0.668 ± 0.009	74.994 ms $\pm$ 2.372
2	0.862 ± 0.006	0.677 ± 0.005	69.317 ms $\pm$ 4.053
3	0.870 ± 0.003	0.686 ± 0.005	69.557 ms $\pm$ 4.159
4	0.874 ± 0.004	0.690 ± 0.005	69.932 ms $\pm$ 5.221
5	0.877 ± 0.005	0.692 ± 0.004	73.173 ms $\pm$ 5.569
6	0.877 ± 0.005	0.693 ± 0.006	74.607 ms $\pm$ 6.838
7	0.881 ± 0.004	0.695 ± 0.003	74.284 ms $\pm$ 5.468
8	0.883 ± 0.004	0.699 ± 0.003	73.505 ms $\pm$ 4.899
9	0.884 ± 0.005	$\mathbf{0.701 \pm 0.006}$	73.620 ms $\pm$ 3.828

Table 5.6: Balanced TinyImageNet results for the kNN learner with extended D^α values.

In Table 5.6, the result of the balanced TinyImageNet dataset can be observed. The discovery AUC and the average accuracy improve during the whole range of the α values. This leads to the assumption that an even bigger parameter value could lead to the highest achievable accuracy and discovery AUC, but the gains in discovery and accuracy diminished towards the end of the tested range. Therefore, the table indicates that larger α values might be beneficial, but the gains become smaller and should not be interpreted as an unlimited improvement. Further experimentation would be necessary to see whether the performance reaches a plateau or decreases again.

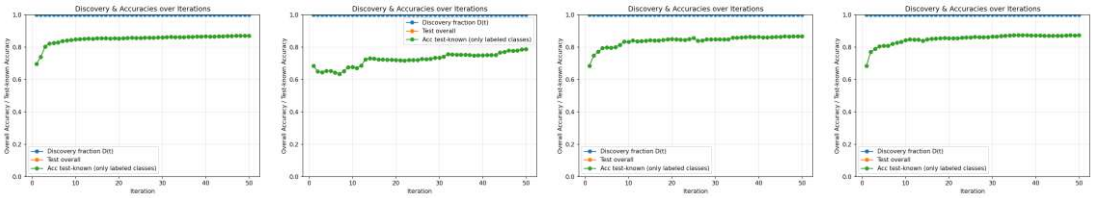
Taken together, the three tables confirm that α directly influences the exploration behavior of the sampling technique, but the strength of the effect depends on the dataset. The improvements also diminish at the upper end, so α appears not as a dominant factor but rather as a parameter for performance optimization. The extended analysis supports the interpretation that the optimal value of α is dataset-dependent. Moderate values are sufficient for simple datasets, while more complex datasets may benefit from larger values. This is consistent with the results of Bamas et al. [BNS23].

5.3 Class Discovery

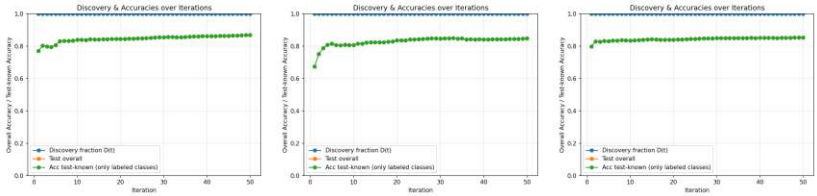
This section compares how quickly the different sampling strategies discover classes during the iterative loop, which supports **RQ1**. The independent variables are the dataset, the panel size, the label budget, and the selection strategy. The dependent variables are the class discovery rate $Disc(t)$, the overall test accuracy, and the test accuracy restricted to the already known and discovered classes.

The following figures show the evolution of the metrics used over the rounds of the loop. It represents the discovery rate $Disc(t)$ in blue, which describes the proportion of classes discovered up to the respective round. The overall accuracy in the test dataset is in green, and the accuracy for previously detected classes is in orange, which indicates the accuracy for the classes that were already labeled or discovered in the previous round. If the orange line is not visible in the plot, it does not mean that it is missing, but rather it overlaps strongly with the green line because both accuracies are very similar. This allows for a simultaneous view of both the progress of the class discovery and the evolution of the model performance over time.

Figure 5.4 illustrates the progression of class discovery and accuracy over the rounds of the iterative process for a single seed, and all selection strategies are considered. The results are based on the FMNIST dataset. For almost all strategies, the discovery fraction rises sharply in the early rounds and then quickly reaches the maximum discovery because all classes are discovered very early, and later rounds therefore only contribute more to consolidation than to the additional class coverage. A similar basic pattern is also evident in the accuracy curves of all strategies. After an initial increase, the test accuracy stabilizes, and then it only improves slightly. The curves for overall accuracy and the accuracy for the already known classes are often overlapping, which indicates that the model operates consistently for the already recognized classes and the overall performance



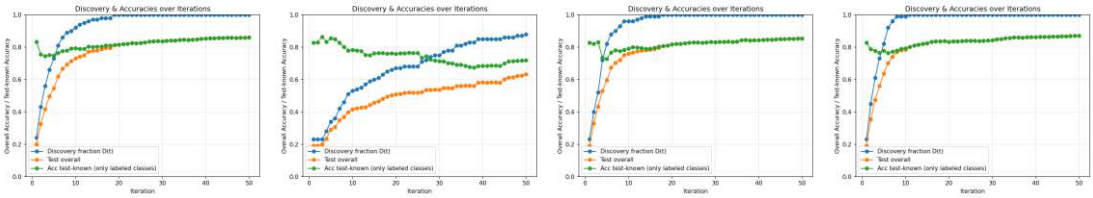
(a) Random (b) Agreement (c) Disagreement (d) Min-margin



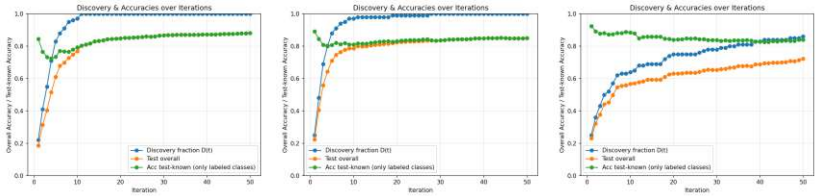
(e) D^{α} with $\alpha = 4$ (f) Farthest (g) Density

Figure 5.4: Comparison of the class discovery for FMNIST data with kNN as learner.

is stable. The agreement method stands out compared to the other methods due to its lower accuracy and more irregular accuracy trajectory, while the other selectors achieve more stable and higher values overall. However, it seems that FMNIST is too simple to reveal large differences between the sampling strategies. Therefore, CIFAR-100 might provide additional insight.



(a) Random (b) Agreement (c) Disagreement (d) Min-margin

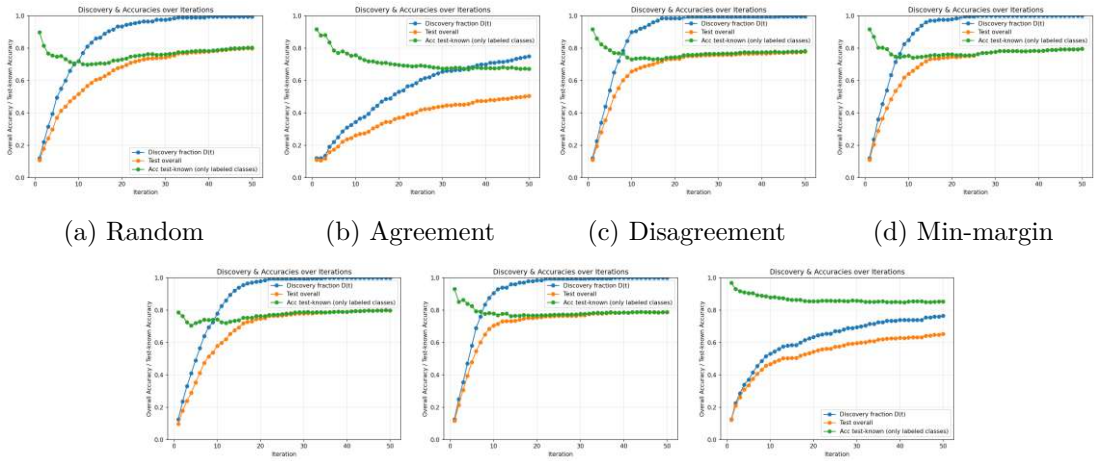


(e) D^{α} with $\alpha = 4$ (f) Farthest (g) Density

Figure 5.5: Comparison of the class discovery for CIFAR-100 with kNN as learner.

In this Figure 5.5, the progress of the class discovery and accuracy on CIFAR-100 using kNN as a learner for different selection strategies can be seen. It is again for a single seed. Strategies like random, disagreement, min-margin, D^{α}, and farthest demonstrate that class detection $Disc(t)$ increases rapidly in the early rounds and quickly improves

test accuracy. This suggests that these strategies can effectively select relevant and informative examples in a few rounds on CIFAR-100. In contrast, agreement and especially density show significantly slower progress. With agreement, the discovery rate increases only gradually, and the overall accuracy remains noticeably lower over many rounds than with most of the other strategies. It can be seen that the accuracy with the known test data exceeds the overall accuracy across the entire iterative process. This leads to the assumption that both measures only converge once all classes have been identified. Otherwise, at least a slight deviation is to be expected. Density also shows a steady increase but achieves lower accuracy scores and slower class discovery over time, suggesting that these methods are less efficient at discovering new classes in a more complex scenario.



(a) Random (b) Agreement (c) Disagreement (d) Min-margin

(e) D^{α} with $\alpha = 4$ (f) Farthest (g) Density

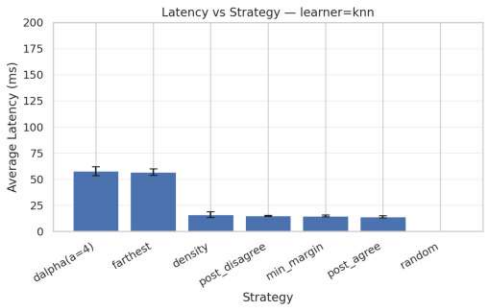
Figure 5.6: Comparison of the class discovery for TinyImageNet with kNN as learner.

In Figure 5.6, a similar pattern for TinyImageNet compared to the CIFAR-100 selector comparison can be observed. Disagreement, min-margin, D^{α}, and farthest show a rapid increase in the discovery fraction during the first iterations. At the same time, the overall test accuracy also increases quickly and approaches the accuracy of the already known test classes. Given that TinyImageNet has many classes, it can be seen as success that four selectors can find all classes reasonably well. Random sampling is slower this time which indicates that it has more issues to find new classes. Agreement and density selection are much slower in both discovery rate and accuracy gain. The gaps are visible until the end of the iterative process. This suggests that both are less suitable for discovering diverse new classes in this setting.

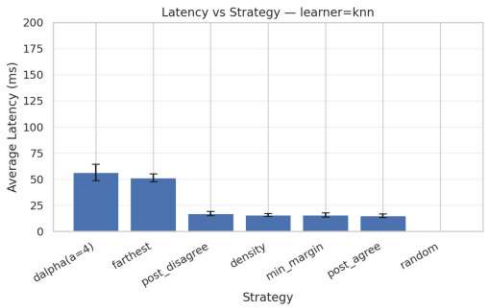
5.4 Selection Latency

The latency completes **RQ1** by evaluating the computational cost of the investigated selection strategies. While class discovery and accuracy describe the quality of the

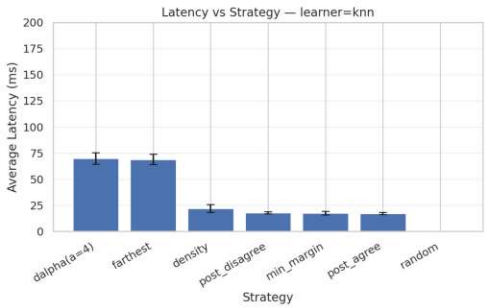
selected samples, latency determines whether a strategy is suitable for an interactive workflow. Latency only refers to the time required by the sampling strategy to choose the next batch of samples from the current panel.



(a) kNN latencies for FMNIST.



(b) kNN latencies for CIFAR-100.



(c) kNN latencies for TinyImageNet.

Figure 5.7: kNN latencies on FMNIST, CIFAR-100 and TinyImageNet for the investigated selectors.

Figure 5.7 shows that the selectors differ substantially in their computational cost. Across the three datasets, random sampling always has the lowest latency, as it does not need any computations or model-based scoring. The model-aware strategies, such as min-margin, agreement, and disagreement, also remain efficient. Their latency is higher than random, but still low enough for an interactive setting. In contrast, the data-driven methods introduce a clearly higher latency. Especially, D^{α} and farthest are the slowest strategies across all datasets, which is expected due to the distance computations for selecting samples.

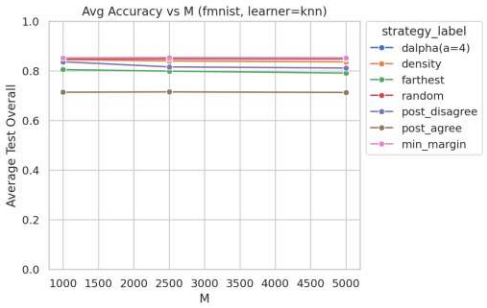
All reported average latencies remain in the millisecond range, which means that the evaluated strategies are feasible for interactive labeling. Model-aware methods offer the best latency-performance compromise, whereas D^{α} and farthest should be used when the additional discovery justifies the higher selection cost.

The same qualitative pattern from the kNN latency discussion can also be observed for logistic regression and MLP learners. In all cases, random selection and model-aware sampling remain the fastest, while the data-driven selectors previously named produce

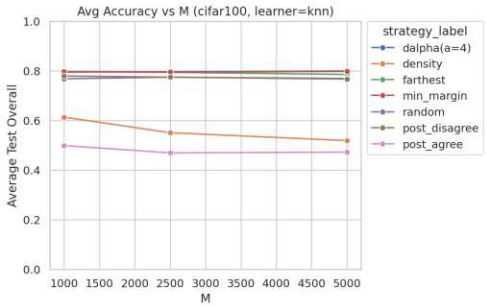
the highest selection latency. Therefore, the additional latency figures for LR and MLP are omitted to avoid redundant visualizations.

5.5 Effects of Panel Size M and Label Budget k

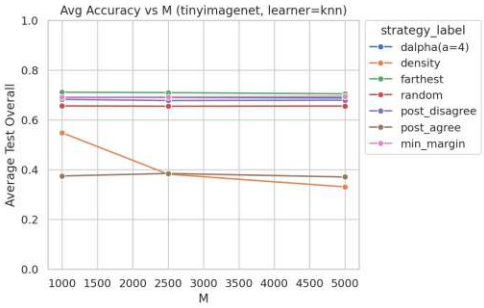
This section examines whether the panel size M and the label budget k influence accuracy and latency for **RQ1.1** and **RQ1.2**. While both parameters directly affect the amount of information available in each round, they do it in different ways. M controls the size of the panel from which the samples are selected, whereas k determines how many instances are labeled per round. The goal is to evaluate whether increasing or decreasing either parameter leads to an improvement of the predictive performance sufficiently to justify the additional computational cost. The independent variables are therefore the datasets, the selection strategies, M and k, while the dependent variables are the average accuracy of the test set and the average selection latency. For the experiment on M, $k = 25$ and $T = 50$ are kept fixed. For the label budget k, the parameters are that $M = 1000$ and $T = 50$ are fixed. The other parameters stay the same.



(a) kNN accuracies of different M sizes for FMNIST.



(b) kNN accuracies of different M sizes for CIFAR-100.

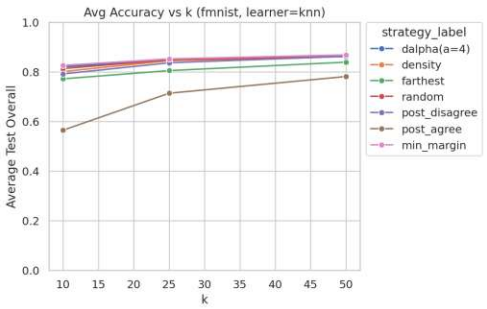


(c) kNN accuracies of different M sizes for TinyImageNet.

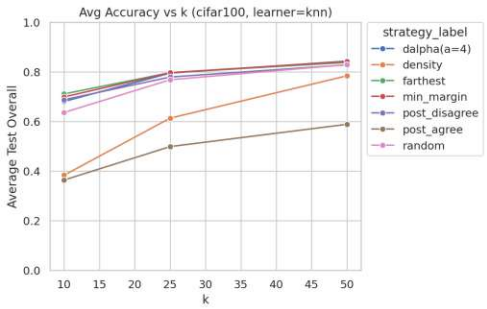
Figure 5.8: kNN accuracies on FMNIST, CIFAR-100 and TinyImageNet for different panel sizes M.

Figure 5.10 indicates that the panel size M leads to a stagnation of the average accuracy on

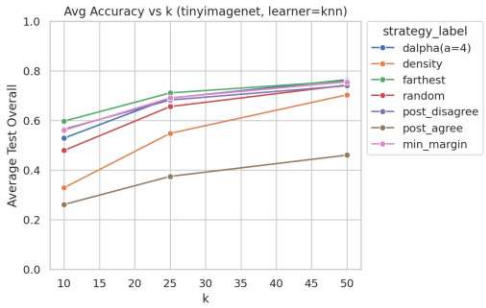
FMNIST, while for CIFAR-100 and TinyImageNet, lower performances can be observed, especially for the density-based strategy. Increasing M from 1000 to 5000 data points does not improve the average accuracy, because most selectors remain flat, which suggests that the strategy choice matters more than the panel size. This indicates that enlarging the candidate pool alone does not automatically provide more useful samples for this dataset.



(a) kNN accuracies of different k sizes for FMNIST.



(b) kNN accuracies of different k sizes for CIFAR-100.



(c) kNN accuracies of different k sizes for TinyImageNet.

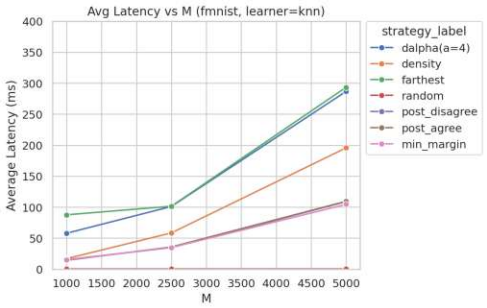
Figure 5.9: kNN accuracies on FMNIST, CIFAR-100 and TinyImageNet for different label budgets k.

In contrast, increasing the label budget k leads to a more clearly improved average accuracy. Smaller values of k provide less information per round, while larger batches allow the learner to update on a broader set of examples. For most selectors, the average accuracies become both higher and more similar to each other when k increases. This effect is visible for density and agreement sampling because they both benefit strongly from moving away from small batches. A similar trend can be observed on CIFAR-100 and TinyImageNet, where the gain from $k = 25$ to $k = 50$ is even more noticeable than on FMNIST. This is also more plausible for a more complex dataset that benefits from additional labeled data per round because the information gain is higher.

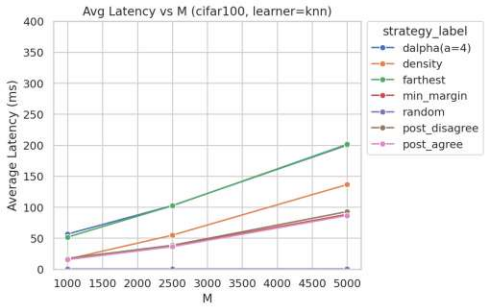
Additional insight is gained from agreement selection. The weak performance of agreement on the more complex dataset suggests that selecting only the most confident points is not

sufficient when the class structure is more ambiguous. In such settings, highly confident samples may be too redundant and may fail to provide enough rich and novel information for effective discovery and boundary refinement.

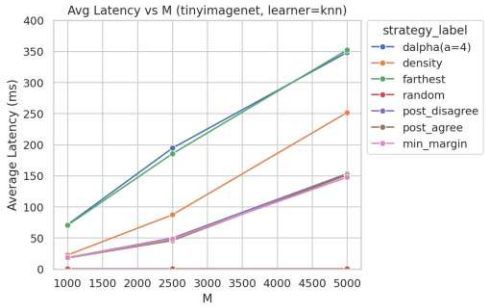
Therefore, it can be said that the panel size M has no impact on the accuracy, while the size of the batches k has more impact on the average accuracy. However, using the highest label budget does not automatically mean that the selector performs best because the latency also has to be observed. In the figures below, the average latency can be seen from the panel size to the left and the batches to the right.



(a) kNN latencies of different M sizes for FMNIST.



(b) kNN latencies of different M sizes for CIFAR-100.

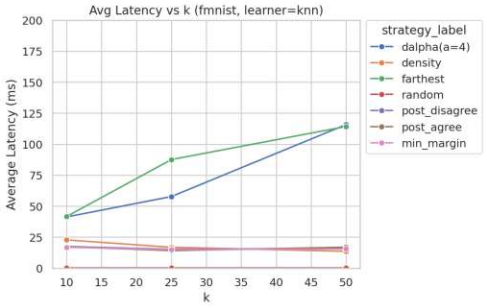


(c) kNN latencies of different M sizes for TinyImageNet.

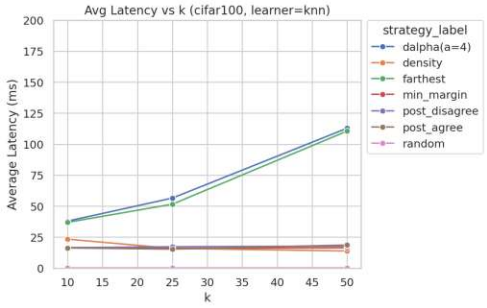
Figure 5.10: kNN latencies on FMNIST, CIFAR-100 and TinyImageNet for different panel sizes M.

The latency for the panel size shows that except for the random baseline, there are only increases in selection time per technique, which leads to the following assumption: A broader panel size does not only decrease the average accuracy. It further increases the average latency, which is the worst possible outcome. The gain is significant, as from 1000 to 2500 data points it almost doubles, and further from there to 5000, which leads to an even more strongly increased time consumption when the panel size is bigger.

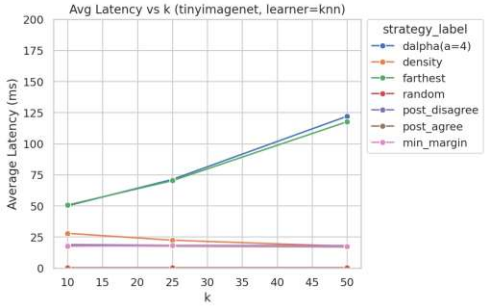
When k is increased, latency also increases, although the effect is smaller than for M and mainly affects the data-driven selectors, which can be seen in Figure 5.11. The random



(a) kNN latencies of different k sizes for FMNIST.



(b) kNN latencies of different k sizes for CIFAR-100.



(c) kNN latencies of different k sizes for TinyImageNet.

Figure 5.11: kNN latencies on FMNIST, CIFAR-100 and TinyImageNet for different label budgets k.

selector has the lowest latency for both M and k, which, in addition to the values of the average accuracy, leads to the best trade-off between accuracy and latency. However, also model-aware methods do not show an increase in latency for larger k, which indicates that these sampling methods remain computationally feasible and could be useful alternatives when sampling quality or accuracy is prioritized over minimal runtime.

The comparison shows that the panel size M has only minor influence on the final average accuracy, as the increased latency is the main bottleneck. The label budget is much more important. The highest label budget k also delivers the best average accuracies, but the latency is an issue for data-driven selectors. The gain in accuracy for the label budget from 25 to 50 data points is less strong, but the latency can be twice as high as for the smaller label budget for the respective selectors, whereas model-aware methods remain unaffected throughout the entire process.

5.6 Balanced vs. Imbalanced

The next research question **RQ 1.3** examines how class imbalance affects the performance of the selection strategies. The main independent variable is the dataset setting, which is balanced versus imbalanced. Additional variables are the panel size, the label budget, the learner, and the selection strategies. The dependent variables are the average test accuracy, the discovery AUC, the average selection latency, and the class discovery behavior over iterations. Unlike balanced datasets, unbalanced datasets are challenging for the selectors because more frequent classes dominate the learning process, while rare classes are more difficult to discover. Table 5.7 provides an overview of the strategies with the highest average accuracies under unbalanced conditions for each dataset and learner.

Dataset	Strategy	Learner	Disc. AUC	Avg. Accuracy	Avg. Latency
cifar100	farthest	knn	0.935 ± 0.005	0.789 ± 0.007	53.758ms ± 1.982
cifar100	min-margin	lr	0.913 ± 0.005	0.753 ± 0.005	8.688ms ± 0.453
cifar100	min-margin	mlp	0.917 ± 0.003	0.773 ± 0.005	8.071ms ± 0.322
fmnist	min-margin	knn	0.977 ± 0.003	0.843 ± 0.004	13.460ms ± 0.462
fmnist	min-margin	lr	0.976 ± 0.004	0.870 ± 0.006	9.729ms ± 0.228
fmnist	disagreement	mlp	0.977 ± 0.002	0.877 ± 0.003	7.417ms ± 0.092
tinyimagenet	farthest	knn	0.885 ± 0.006	0.690 ± 0.006	60.218ms ± 2.681
tinyimagenet	disagreement	lr	0.841 ± 0.004	0.620 ± 0.005	11.902ms ± 1.252
tinyimagenet	farthest	mlp	0.885 ± 0.006	0.643 ± 0.006	63.044ms ± 1.546

Table 5.7: Best Strategies according to average accuracy for imbalanced datasets.

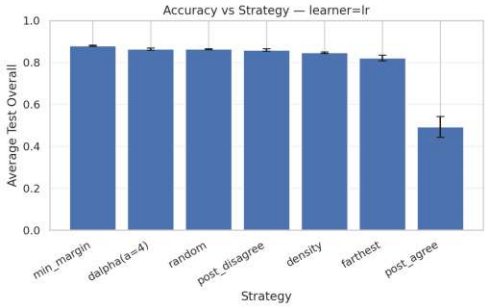
The details of the Table 5.7 show that the best strategies for imbalanced datasets vary depending on the dataset and learner. For TinyImageNet, farthest sampling achieves the best results for kNN and MLP, while disagreement is the strongest for LR. For FMNIST and CIFAR-100, min-margin and disagreement are among the best-performing strategies. The table further shows that the data-driven methods have higher latency than the model-aware methods.

To determine whether the sampling strategies differ significantly, Friedman tests are conducted. Table 5.8 shows that all Friedman tests are statistically significant. This means that the sampling strategies do not perform equally when the class distribution is imbalanced. It is further supported by Table 5.7. It can be concluded that the sampling strategy is not only an implementation detail but also an important factor for performance under class imbalance.

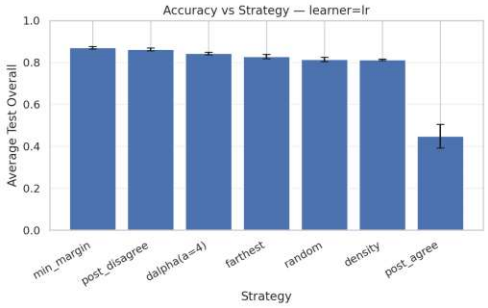
The results in Figure 5.12 for the imbalanced datasets show that the ranking of the selectors changes moderately on FMNIST. For logistic regression, min-margin remains the best selector in both balanced and imbalanced settings. For MLP, disagreement performs better in the imbalanced setting. Agreement selection is again the weakest selector and shows higher instability than the others.

Dataset	Learner	Friedman χ^2	p-value	Significant
CIFAR-100	kNN	58.757	8.05×10^{-11}	Yes
CIFAR-100	LR	60.000	4.50×10^{-11}	Yes
CIFAR-100	MLP	58.843	7.73×10^{-11}	Yes
FMNIST	kNN	55.843	3.13×10^{-10}	Yes
FMNIST	LR	56.057	2.83×10^{-10}	Yes
FMNIST	MLP	56.914	1.90×10^{-10}	Yes
TinyImageNet	kNN	57.129	1.72×10^{-10}	Yes
TinyImageNet	LR	58.929	7.43×10^{-11}	Yes
TinyImageNet	MLP	59.614	5.39×10^{-11}	Yes

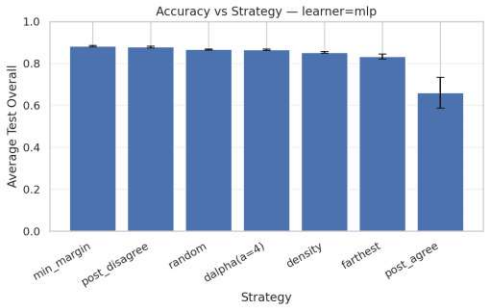
Table 5.8: Friedman test results for average test accuracy under imbalanced conditions. The tests compare the seven sampling strategies separately for each dataset and learner over ten repeated runs.



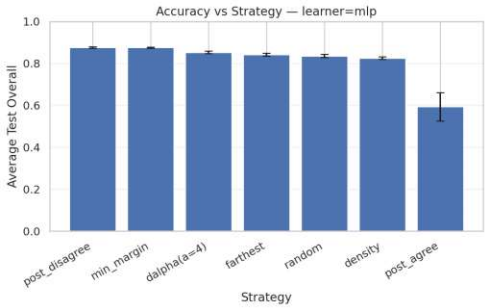
(a) LR, balanced



(b) LR, imbalanced

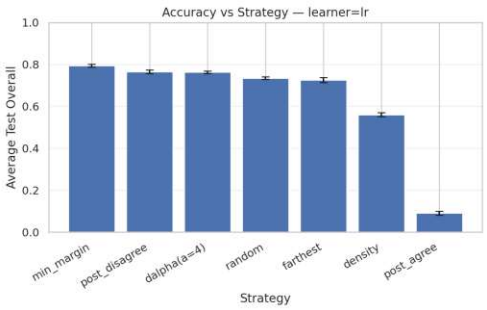


(c) MLP, balanced



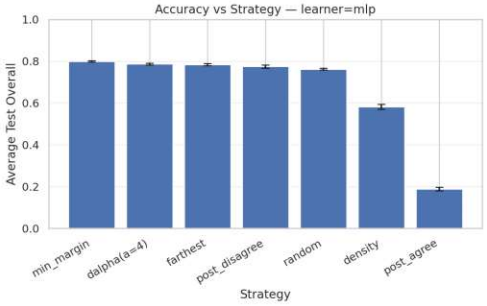
(d) MLP, imbalanced

Figure 5.12: Comparison of balanced and imbalanced FMNIST results for logistic regression and MLP.

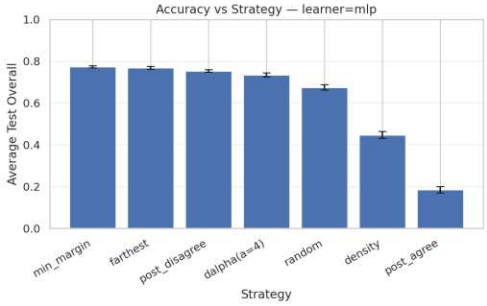


(a) Logistic regression, balanced

(b) Logistic regression, imbalanced



(c) MLP, balanced



(d) MLP, imbalanced

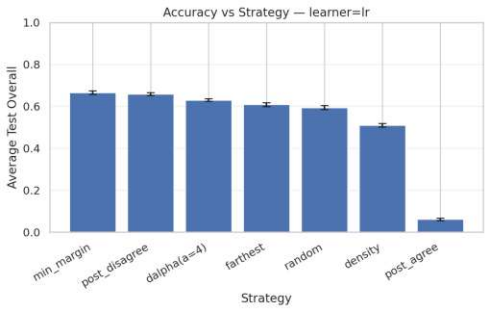
Figure 5.13: Comparison of balanced and imbalanced accuracy results on CIFAR-100 for logistic regression and MLP.

Figure 5.13 shows that for CIFAR-100 with logistic regression, the ranking of the selectors changes under imbalance. Min-margin remains the best strategy, while D^{α}, farthest, and disagreement also remain competitive. The results of random sampling are even worse in this case, suggesting that uniform random sampling is less reliable for capturing rare classes. Density and agreement remain the weakest selection methods.

For CIFAR-100 with the MLP learner, a similar tendency can be observed. Min-margin is again the strongest strategy, while farthest, disagreement, and D^{α} sampling remain competitive but do not surpass it in the imbalanced case. Density again drops clearly, and agreement performs worst by far. As before, random sampling remains a useful baseline in the balanced setting, but it cannot keep up with the strongest sampling strategies when the dataset becomes imbalanced.

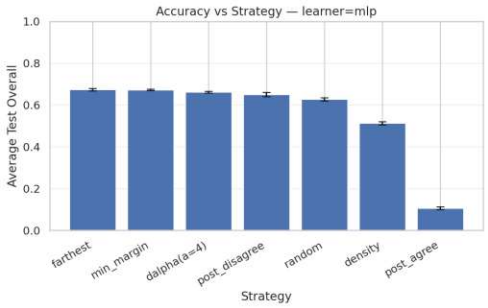
Figure 5.14 shows that for logistic regression, min-margin and disagreement achieve the highest results in the balanced setting, while disagreement is the best strategy under imbalance. D^{α} and farthest remain competitive, whereas random sampling and density are clearly weaker. Agreement sampling performs worst in both settings.

For the MLP learner, farthest and min-margin reach the highest accuracy scores in the balanced setting. Under imbalance, farthest remains the strongest strategy, followed by min-margin, disagreement, and D^{α}. Similar to logistic regression, random and density lose

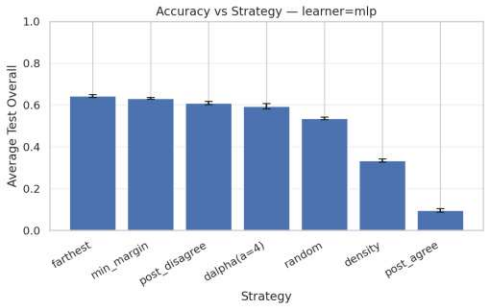


(a) Logistic regression, balanced

(b) Logistic regression, imbalanced



(c) MLP, balanced



(d) MLP, imbalanced

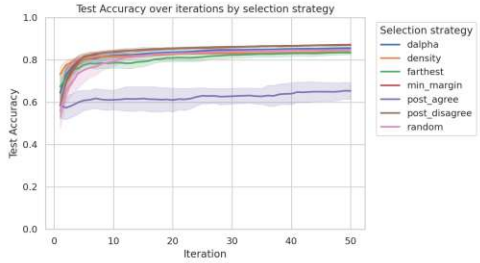
Figure 5.14: Comparison of balanced and imbalanced accuracy results on TinyImageNet for logistic regression and MLP.

performance. Density sampling drops clearly in the imbalanced setting, while agreement remains separate from all other strategies.

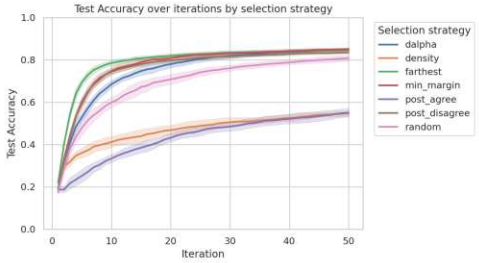
As in Figure 5.3 for the balanced data, it is possible to observe what happens over time for the imbalanced datasets.

The comparison of the imbalanced rounds in Figure 5.15 shows that the confidence intervals are of a similar order of magnitude to those of the balanced datasets. On FMNIST, density starts strong but is quickly overtaken by min-margin and disagreement. On CIFAR-100, farthest sampling is the best-performing selector during the first part of the loop, while min-margin catches up and overtakes it later. On TinyImageNet, farthest has the best start and leads in accuracy over iterations until round 40, but is then overtaken by D^{α} and min-margin sampling. These two, together with disagreement and farthest, have similar accuracies at the end, while random drops off and agreement and density struggle to improve accuracy with the more complex dataset. Compared with the balanced setting in Figure 5.3, the overall accuracy is lower and the curves increase more slowly.

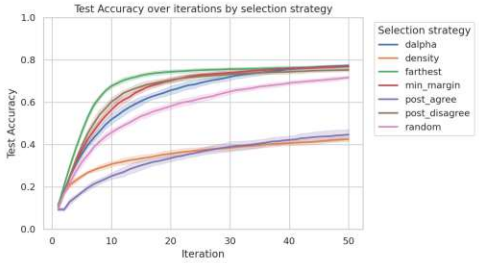
Figure 5.16 shows that the comparatively mild imbalance on FMNIST affects the early class discovery more than in the balanced case. Random and agreement require several additional rounds to recover the full class set, whereas the remaining strategies only need



(a) FMNIST

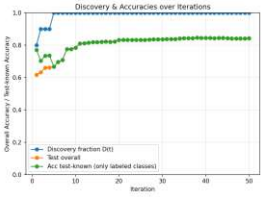


(b) CIFAR-100

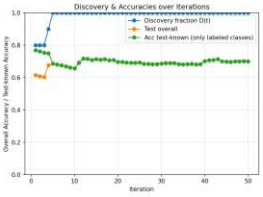


(a) TinyImageNet

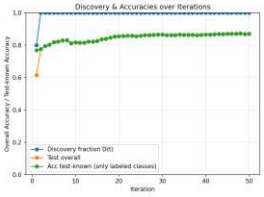
Figure 5.15: Imbalanced accuracies for FMNIST, CIFAR-100, and TinyImageNet.



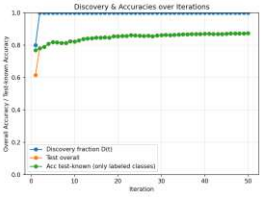
(a) Random



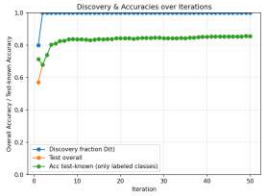
(b) Agreement



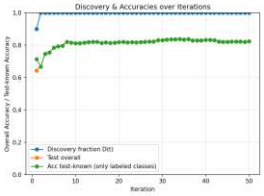
(c) Disagreement



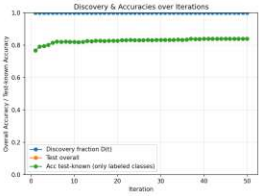
(d) Min-margin



(e) D^{α} with $\alpha = 4$



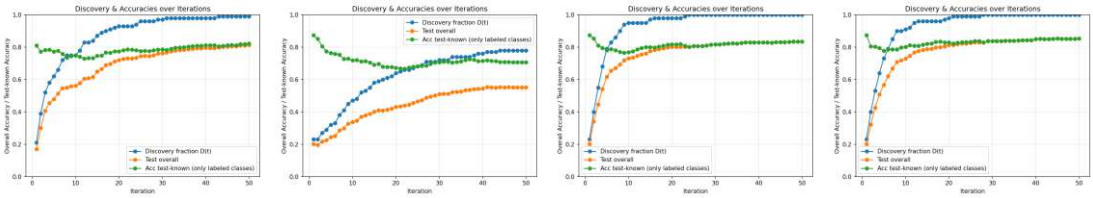
(f) Farthest



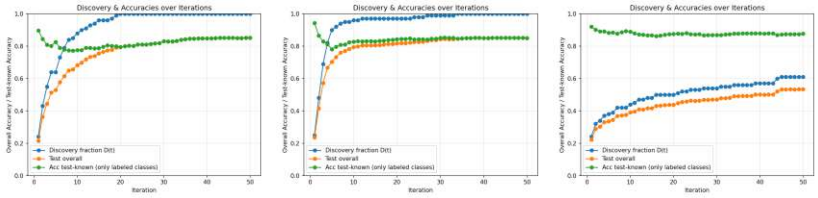
(g) Density

Figure 5.16: Comparison of the class discovery for imbalanced FMNIST data with kNN as learner.

about one extra round to have the full knowledge. Although the overall effect remains limited on FMNIST, the pattern already indicates that these weaknesses could be more exposed with more complex datasets.



(a) Random (b) Agreement (c) Disagreement (d) Min-margin

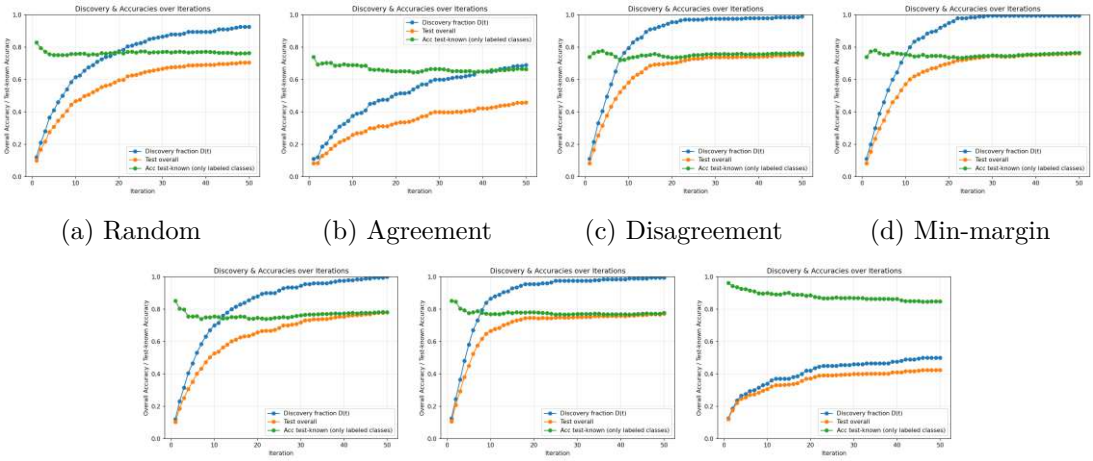


(e) D^{α} with $\alpha = 4$ (f) Farthest (g) Density

Figure 5.17: Comparison of the class discovery for imbalanced CIFAR-100 data with kNN as learner.

The effect becomes much clearer on the imbalanced CIFAR-100 in Figure 5.17, which reveals that the differences get more noticeable. The class discovery is fast for several selectors but slows down when more classes are found and undiscovered classes are rarer. It can be observed that most of the selectors are able to identify all 100 classes except the agreement and density selector. The fastest class discovery is achieved with the D^{α} sampling, followed by disagreement, min-margin, and farthest sampling. Random sampling improves only slowly and steadily, and agreement sampling does not reach a competitive level. This suggests that random sampling is not a sufficiently effective strategy for imbalanced datasets, as it tends to select already known classes from the unlabeled pool.

Figure 5.18 shows that compared to the balanced setting in Figure 5.6, the differences become more visible. Several strategies still discover classes quickly in the beginning, but the discovery curves flatten once the remaining undiscovered classes become rarer. Disagreement, min-margin, D^{α}, and farthest show the strongest class discovery behavior. They reach complete class discovery by the end of the iterative process and obtain good test accuracy. Random sampling improves steadily, but it discovers classes slower and does not discover every class. This leads to the assumption that random selection is less reliable when the class distribution is imbalanced. Agreement and density perform worse than the other selections. Both show slow class discovery and low overall test accuracy, but density is the worst of the two. The selector fails to find enough new classes for a stronger overall performance.



(a) Random (b) Agreement (c) Disagreement (d) Min-margin

(e) D^{α} with $\alpha = 4$ (f) Farthest (g) Density

Figure 5.18: Comparison of the class discovery for imbalanced TinyImageNet data with kNN as learner.

After observing the class discovery, it can be said that class imbalance lowers accuracy and slows down class discovery. The effect is small on FMNIST but becomes visible on CIFAR-100 and TinyImageNet. The selector ranking also changes under imbalance, which shows that the sampling strategy is an important factor when the class distribution is no longer uniform. The exploration-to-optimization behavior visible in the trajectories is analyzed in more detail in Section 5.7.

5.7 Model-Aware vs. Data-Driven Sampling

This section compares model-aware and data-driven sampling strategies with respect to accuracy, class discovery, and latency for the same labeling budget in order to answer **RQ2**. Previous sections presented the results for accuracy, class discovery, parameter sensitivity, and class imbalance separately. The focus here is the role of the selectors during iteration.

The results show that the two selector families differ primarily in when they are most useful during iteration. Data-driven selectors are beneficial in the early rounds because they promote coverage of the candidate pool and can thus support faster class discovery. This is illustrated in the class detection results for CIFAR-100 and TinyImageNet in Figures 5.5 and 5.6, where farthest and D^{α} sampling show strong early discovery behavior. Model-aware selectors, on the other hand, only become useful once the learner has observed a sufficient number of classes. From this onward, uncertainty-based assessments gain importance and can contribute to refining the decision boundaries.

This transition from exploration to optimization is also evident in the accuracy curves. In Figure 5.3, farthest sampling achieves strong results in the early rounds on CIFAR-100,

while min-margin catches up and achieves greater accuracy. A similar pattern can be observed with imbalanced datasets in Figure 5.15. On imbalanced CIFAR-100, farthest sampling is the best-performing selector in the first part of the iterative process but is later overtaken by min-margin. This suggests that early rounds benefit from exploration, while later rounds benefit more from the refinement of model-aware methods.

The class discovery graphs support this interpretation. With simple datasets like FMNIST (Figure 5.4), almost all strategies detect classes very quickly. Therefore, additional exploration offers only limited benefit. For more complex datasets, the differences between the selector families become more visible. On CIFAR-100 and TinyImageNet, the model achieves good results with the classes that have been discovered so far, while further progress depends on how quickly the unseen classes are found. This explains why exploratory selectors like farthest and D^{α} are useful in the early stages.

The main disadvantage of data-driven sampling is latency. Experiments with different panel and label budgets show that data-driven selectors are more sensitive to increases in M and k. Increasing M has a negative impact on the computational cost because more candidate points in the panel size need to be compared. Increasing k also has a greater impact on data-driven methods, as it requires additional distance-based updates during selection. Model-aware selectors are less affected by these changes because their scoring relies on the predictions of the learner or confidence estimates. This explains why min-margin often has a notable trade-off between speed and accuracy, although its accuracy is close or better to that of more computationally intensive data-driven selectors.

In the case of class imbalance, the distinction between selector families becomes even more critical. The results of the imbalanced class discovery in Figures 5.16, 5.17, and 5.18 show that rare classes with the same interaction budget are harder to discover. Selectors that stay too close to frequent, dense, or already safe regions are more likely to miss minority classes. This explains why random sampling, agreement, and density are more strongly affected by imbalances while the parameters farthest, D^{α}, min-margin, and disagreement are more robust.

The results under imbalance therefore show that exploration is beneficial when many classes remain hidden, especially with more complex datasets. However, exploration is not consistently better than model-aware sampling throughout the entire process. Once sufficient class coverage is achieved, further improvements depend more on coverage within the classes and the refinement of decision boundaries. For this reason, disagreement and min-margin combine stable class finding with significant accuracy improvements.

Model-aware and data-driven strategies have different benefits and should not be viewed as strictly competing alternatives, because they complement each other. Data-driven selectors are well-suited for early exploration and class discovery, while model-aware selectors become more important in later rounds to identify areas with uncertainty. This motivates the hybrid selectors evaluated in the following subsection, which combine data-driven preselection with model-aware min-margin refinement.

5.7.1 Hybrid selector with min-margin refinement

This hybrid method implemented for this follow-up experiment uses a two-stage process, which is followed by a min-margin refinement. The tested variants are density-based preselection followed by min-margin sampling, and farthest-based preselection followed by min-margin sampling. In both cases, the first stage selects a subset of data points according to the independent structural criterion in the panel size, while the second stage uses min-margin to refine the final selection using the uncertainty of the model. The motivation is not only to combine the best methods together, but also to test whether complementary selection principles can improve the trade-off between predictive performance and selection latency. Min-margin is chosen because it repeatedly achieved a strong performance, balancing accuracy and computational performance. Density is used as a first-stage method because it provides the data-driven criterion based on local structure and it is much faster than the more computationally expensive selectors such as farthest and D^{α} sampling. Although density is not the best-performing method, it may provide a useful pool for later refinement. Farthest sampling is included as the second preselection strategy because it promotes quick class discovery and good accuracy performance. The independent variable is the selection strategy, comparing hybrid selection against density, min-margin, and farthest sampling. The dependent variables are the discovery AUC, the average test accuracy, and the selection latency.

Learner	Selector	Discovery AUC	Avg. Accuracy	Avg. Latency
kNN	Random	0.851 ± 0.012	0.688 ± 0.012	0.088 ± 0.003 ms
kNN	Min-margin	0.922 ± 0.008	0.771 ± 0.008	14.623 ± 1.317 ms
kNN	Farthest	0.935 ± 0.005	0.789 ± 0.007	54.582 ± 1.818 ms
kNN	Farthest+Min-margin	0.937 ± 0.004	0.786 ± 0.006	119.974 ± 3.292 ms
kNN	Density+Min-margin	0.926 ± 0.005	$\mathbf{0.803 \pm 0.004}$	18.896 ± 2.133 ms
LR	Random	0.851 ± 0.012	0.596 ± 0.014	0.086 ± 0.009 ms
LR	Min-margin	0.913 ± 0.005	0.753 ± 0.005	8.688 ± 0.453 ms
LR	Farthest	0.935 ± 0.005	0.695 ± 0.009	50.217 ± 2.094 ms
LR	Farthest+Min-margin	0.933 ± 0.004	0.731 ± 0.005	119.811 ± 2.244 ms
LR	Density+Min-margin	0.926 ± 0.005	$\mathbf{0.772 \pm 0.008}$	14.970 ± 1.409 ms
MLP	Random	0.851 ± 0.012	0.675 ± 0.012	0.085 ± 0.004 ms
MLP	Min-margin	0.917 ± 0.003	0.773 ± 0.005	8.071 ± 0.322 ms
MLP	Farthest	0.935 ± 0.005	0.768 ± 0.006	55.874 ± 1.568 ms
MLP	Farthest+Min-margin	0.933 ± 0.005	0.771 ± 0.006	117.952 ± 1.627 ms
MLP	Density+Min-margin	0.926 ± 0.004	$\mathbf{0.790 \pm 0.005}$	12.075 ± 0.812 ms

Table 5.9: Comparison of hybrid selectors with relevant baselines on imbalanced CIFAR-100.

Table 5.9 shows that the two hybrid selectors have different trade-offs in this follow-up experiment. The density-based hybrid, density and min-margin, achieves the highest accuracy across kNN, LR, and MLP. Its selection latency is only moderately higher

than that of min-margin and clearly lower than farthest sampling. This indicates that density-based preselection provides an efficient candidate filtering mechanism before applying min-margin refinement. In contrast, farthest + min-margin remain competitive with regards to discovery AUC, but they do not improve accuracy over density and min-margin and show higher latency. In fact, the latency is higher than all other methods, which does not yield an improvement in predictive performance.

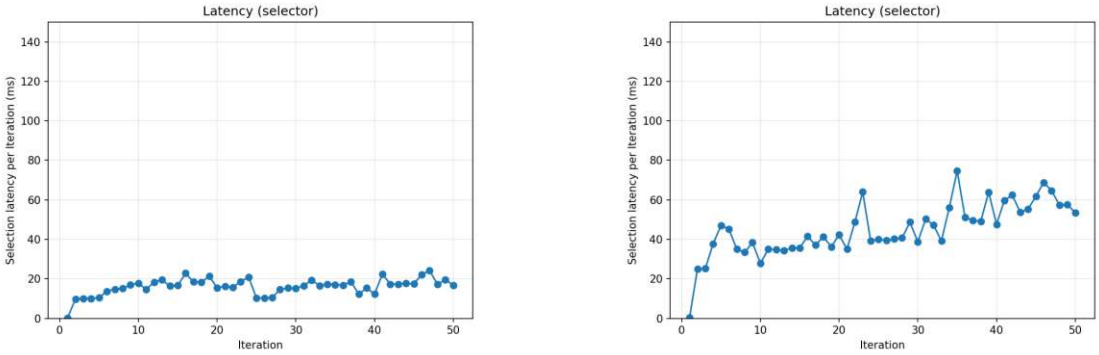
Furthermore, the AUC values for class discovery show that the hybrid methods remain competitive in class discovery, while farthest sampling remains the strongest. This shows the design of the hybrid methods. Instead of solely optimizing for maximum exploration, it combines preselection with model-aware uncertainty refinement. The results suggest that density is more effective as a preliminary filtering mechanism than as a standalone selector. Density and min-margin provide the most favorable compromise between predictive performance and selection latency, while farthest and min-margin support discovery performance but with higher computational cost.

5.8 GPU Performance

The final open research question concerns the effect of GPU acceleration on the latency of the iterative loop. To answer **RQ3**, the runtime of the main stages, which are selection, model fitting, and prediction, is compared between CPU- and GPU-based implementations. The main independent variables are the hardware implementation, which compares CPU-based and GPU-accelerated versions, the panel size, and the label budget. Depending on the experiment, the implementation of the learner or the selector varies, where scikit-learn is compared to cuML for kNN or CPU-based versus PyTorch/GPU-based distance computations for data-driven selectors. The dependent variables are selection latency per iteration, total runtime of the selection, fitting, and prediction stage. Further dependent variables are the speedup factor and the step differences between CPU and GPU execution. The goal is not only to determine whether GPU aids total runtime because this can be expected but also to identify which components benefit most.

As mentioned in the experiment design, model fitting, selection, and prediction are GPU-accelerated. This decreases time consumption of the system in general. In the following figure, two disagreement strategies are compared to each other. The left image is the GPU-accelerated iteration where kNN is from cuML, while the right one is from the scikit-learn library. The x-axis shows the ongoing iteration process from round 1 to 50, and the y-axis shows the selector latency in milliseconds (ms) per round.

Figure 5.19 compares the disagreement selection with a GPU-accelerated kNN learner from cuML and a CPU-based kNN learner from scikit-learn. The first iteration is not informative for the model-aware selector itself because the initial iteration is still random before the learner-based scores are available. While the selection latency increases approximately for every round for the CPU implementation, the GPU-accelerated version remains quite stable throughout the iterations. In total, the GPU-based version requires

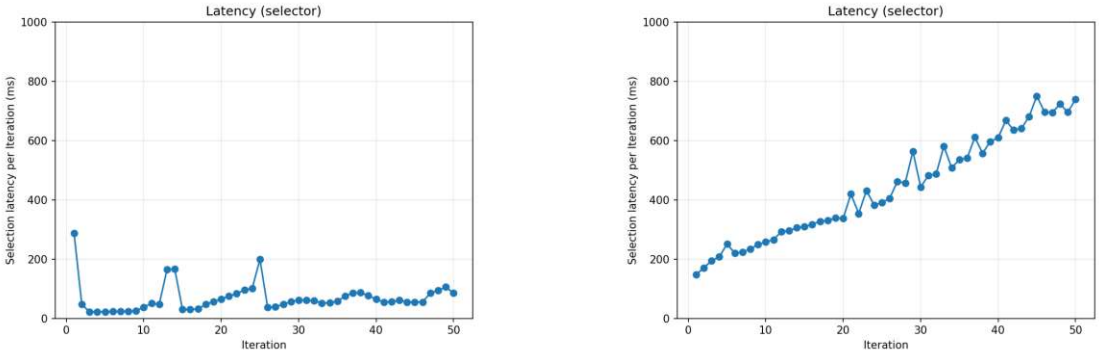


(a) cuML (b) scikit-learn

Figure 5.19: Latency curves comparison of kNN algorithm.

about 800 ms across all rounds, which corresponds to 16 ms on average per iteration. The scikit-learn configuration takes about 2250 ms, which is on average 45 ms. This means that the GPU-accelerated variant is almost three times faster, which makes it clearly preferable for interactive usage.

In addition, the impact of GPU accelerating data-driven selection methods is being investigated. The impact of switching from CPU to GPU is being examined to determine whether it leads to an improvement or whether there is little difference in the selection of data points. The data-driven method farthest is tested once with GPU acceleration and once without. The selection process is monitored by the latency plot per round, which is used to describe whether differences occur or GPU training is not needed.



(a) PyTorch (b) CPU torch tensor

Figure 5.20: Latency curves comparison of farthest selection latency using kNN.

From Figure 5.20, it can be seen that GPU acceleration proves beneficial. The selection method is far slower if it uses the CPU implementation. Both plots show an increase in latency per round due to the growing number of data points to remember and use. While the GPU-accelerated selection method shows only moderate increases, the CPU

implementation grows continuously in the measured latency per round. It is almost a proportional increase per round. It takes, on average latency, around 440 ms, which is concerning for an iterative selection process but still below the threshold. The GPU-supported iteration selection takes much less time per round. The latency reaches a peak of approximately 300 ms and decreases to around 75 ms in the later stages of the iterative process. The initial spike is due to a warm-up phase in which PyTorch initializes the CUDA context, loads the kernel, configures the necessary handles, and sets up the caching memory allocator. This phase also includes the memory allocation and data transfer from the CPU to the GPU. Three additional spikes can be observed where the same happens. Aside from these overheads, the latency does increase but remains within an acceptable range and well below the specified limit of one second.

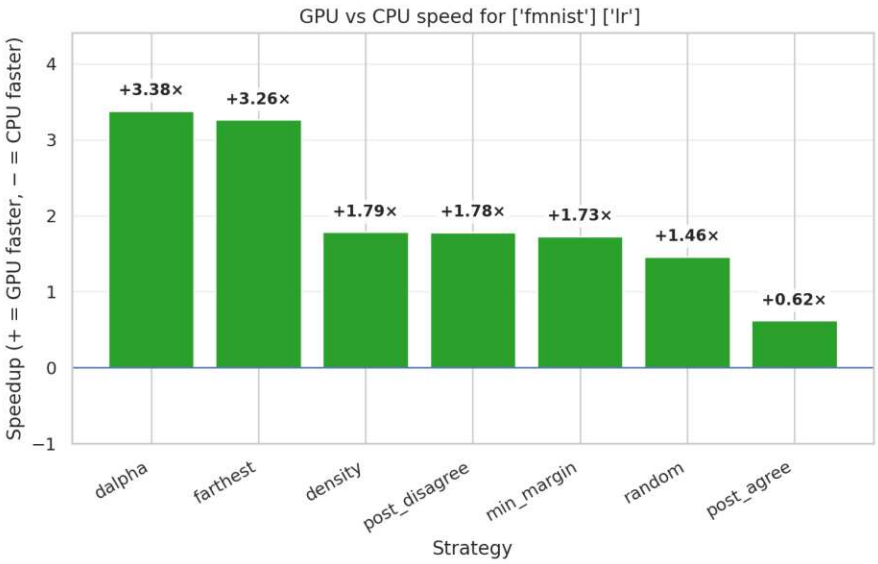
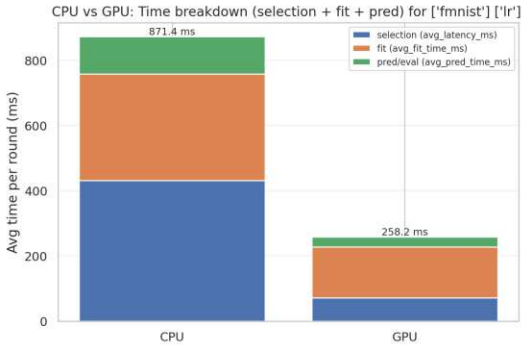


Figure 5.21: Speedup comparison between GPU and CPU.

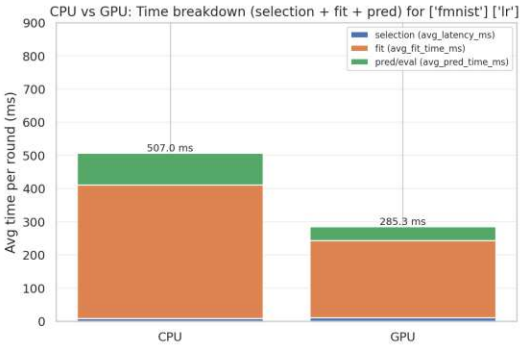
Figure 5.21 shows the overall acceleration per selector. The largest gains are observed in the data-driven methods, with a maximum acceleration of approximately 3.38× for the D^{α} sampling and the farthest with about a 3.26× speedup. The remaining selectors also benefit significantly, but their accelerations are lower and between 1.79× and 0.62×. This pattern confirms that GPU acceleration is particularly effective when the selector itself performs computationally intensive operations in the panel size, while the gains in the less computationally intensive selectors are mainly due to the accelerated learning algorithm. Even though the speedup might be less, in this thesis, a selection latency below one second is treated as a practical threshold for interactive usability, and therefore, each speedup can be considered useful. It is also worth noting that the achievable speedup depends on the available hardware and may vary accordingly.

To further illustrate the effect of GPU acceleration, Figure 5.22 shows the time breakdown per phase (selection, training, and prediction) along with the speedup per selector. As a

learner for the breakdown, logistic regression is used in the plot, and CIFAR-100 with LR can be observed in Appendix A.4. It is focused on LR because MLP shows qualitatively similar trends, while kNN is less informative for the comparison, as it is a lazy learner, performs little fitting during training, and does not provide meaningful fitting time gains.



(a) CPU vs. GPU time breakdown for D^{α} on FMNIST with LR.



(b) CPU vs. GPU time breakdown for disagreement on FMNIST with LR.

Figure 5.22: Time breakdown of each stage for FMNIST.

Figure 5.22 breaks the total runtime down into selection, fitting, and prediction phases. For FMNIST on LR with selectors, it can be seen that the GPU acceleration reduces the total runtime in both pipelines. While GPU acceleration improves fitting, selection, and prediction for D^{α}, only the fitting and prediction are improved for disagreement sampling. For D^{α}, the runtime decreases from 871.8 ms on the CPU to 258.2 ms on the GPU. The most improvement is made by the selection step, but also fitting and prediction become faster. For disagreement, the runtime decreases from 507.8 to 285.3 ms. The fitting dominates the runtime on CPU and GPU, while a lesser contribution from prediction and selection can be seen. The results further indicate that the additional CPU-GPU data transfer is not a bottleneck in the setting and the gains outweigh the transfer overhead.

5.9 Discussion of Results

The results show that random sampling remains competitive with simple and balanced datasets, but its performance decreases significantly with the increasing complexity of the data distribution, especially in the case of class imbalance. This already suggests that the benefit of more sophisticated sampling strategies strongly depends on the difficulty of the discovery problem.

The goal of this discussion is not only to summarize which strategies achieve higher accuracy, fast class discovery, or low latency, but also to understand why these differences occur under iterative loop conditions. The following discussion revisits the research questions of this work and relates the empirical findings to the trade-offs between exploration, refinement, and interactivity.

The first research question **RQ1** investigates how different sampling strategies affect accuracy, class discovery, and latency in the iterative loop. The results show that the effect of the sampling strategy depends on the complexity of the dataset and the class distribution. For simple and balanced datasets like FMNIST, random sampling remains a good choice because most classes are detected very early. The benefit of more complex sampling strategies is limited, and the differences in average accuracy are small. For more complex datasets like CIFAR-100 and TinyImageNet, sophisticated sampling strategies become more important under the balanced and imbalanced case, while random sampling is less reliable. The sub-questions of RQ1 further explain which experimental factors influence this behavior.

For **RQ1.1**, increasing the panel size M raises the cost of distance computation and operations, particularly selectors like farthest and D^α, which are slower when the panel size grows due to their reliance on geometric operations. Richer embeddings can help separate classes better for more complex datasets, but the experiments do not show a consistent increase in downstream performance when the panel size is enlarged. Instead, the dominant effect is the increase in latency. This suggests that larger panels are not automatically better and that interpretability can even degrade when complexity grows, mainly because the computational burden grows.

RQ1.2 is answered as larger k improves discovery and average accuracy because more points can be labeled per round. The gain decreases quickly because the step from a label budget of 10 to 25 is noticeable, whereas the next step to 50 is marginal. But a larger k also increases selection latency for non-trivial selectors, and the increase can be heavy for data-driven methods. Therefore, increasing k is beneficial up to a point, but beyond that point the additional computational cost is no longer justified by the modest performance gain for data-driven methods. Model-aware methods and random sampling continue the improvement, while the latency stays the same. However, also here the question arises whether the higher values of k justify the marginal gains.

To measure **RQ1.3**, the effect of the class imbalance within the iterative loop is tested. The experiments show that the required rounds to achieve complete class coverage are greater in the case of imbalance, as minority classes occur more infrequently and are therefore more difficult to detect in the early stages. As a consequence, the discovery fraction rises more slowly and the discovery AUC decreases. Imbalance also leads to a lower prediction performance because the learner receives a less balanced dataset and builds stronger representations for the majority classes than for the minority classes. This effect is noticeable in more complex datasets, where the minority classes are not only less frequent but additionally more difficult to distinguish. Consequently, the imbalance widens the gap between what the model has already learned well and what is still underrepresented in the existing data. Qualitative comparisons also show that this effect varies depending on the selector. Exploratory strategies restore the class coverage more reliably, while agreement, density, and random selectors are more affected.

The qualitative comparison of the class discovery reveals a consistent pattern. On the imbalanced FMNIST, random and agreement struggle early and need more rounds to

recover the missing classes, whereas the remaining strategies lose much less ground relative to the balanced setting. On imbalanced CIFAR-100, the effect is stronger. The discovery is initially fast but then slows down because it is harder to find novel classes. In the imbalanced variant, most selectors eventually discover all 100 classes except agreement, while farthest, D^{α} and disagreement reach high coverage much earlier. This supports the interpretation that imbalance primarily penalizes strategies that do not actively promote exploration of less represented regions.

For the accuracy trajectories, the curves match the typical exploration-to-optimization shape. Exploratory methods like D^{α} and farthest push discovery quickly toward saturation, but the accuracy continues to improve afterwards because the model still requires more within-class coverage and better decision boundaries on more complex datasets. In contrast, model-aware methods such as min-margin often reach comparable final accuracy with lower latency faster and close the gap between seen and unseen structure more efficiently once a sufficient level of discovery has been achieved.

The latency results show that random is essentially cost-free because it is the fastest picking selector, while model-aware selectors are still cheap and remain comparatively inexpensive. Data-driven selectors are clearly more costly because they repeatedly compute geometric relationships in the panel size, which therefore dominates runtime. The latency can depend strongly on the implementation and substantially benefits from GPU support. Nevertheless, under the default configuration the interaction remains within the practically usable range.

Overall for **RQ1**, both data-driven and model-aware methods can outperform random sampling by reaching higher accuracy and faster class discovery under the same labeling budget. This effect can especially be observed on harder or imbalanced datasets where the informed selection becomes more important. On simple and balanced datasets, however, the advantage over the random baseline is often very small. The main trade-off is latency. Model-aware sampling methods come closest to random sampling in terms of speed, whereas data-driven methods often provide stronger exploration at a higher computational cost.

RQ2 concerns the benefit of an adaptive exploration-to-refinement schedule, and the results support the intuition of the different families of sampling strategies. Data-driven methods are superior at first, as they maximize novelty and accelerate class coverage, which translates into steeper discovery curves and a shorter time to coverage, while model-aware methods are typically superior later on because they focus on refining decision boundaries. This leads to more accuracy gains once the class set is sufficiently established. The argument can be further strengthened because latency is part of the objective. In early rounds, more intensive exploration may be justified if it quickly discovers new classes, while in later rounds it may be a better effort to switch to more cost-effective uncertainty sampling to maintain responsiveness when the expensive selectors no longer provide proportional benefits. An adaptive schedule should improve average accuracy and average discovery for the same labeling budget while reducing the interaction costs by avoiding computationally expensive selections when the utility is low.

RQ3 can be answered by noting that the components benefiting most from GPU acceleration are selection, fitting, and prediction, with the strongest selector-specific gains observed for the data-driven methods. In terms of overall speedup for FMNIST data, D^{α} achieves about 3.38× and farthest about 3.26×, while the remaining strategies benefit by around 1.79× to 0.62×. This confirms that GPU acceleration is especially effective when the selector itself relies on heavy geometric calculations. For model-aware and random-based strategies, the speedup is still substantial but is driven more strongly by the accelerated learner than by the selector. Overall, GPU acceleration makes the iterative loop considerably better suited for interactive use, because when the memory size, the labeling budget, or the panel size is increased, the CPU-only execution falls behind much more clearly. Important to mention is that the speedup further relies on the speed of one's own hardware and is a bottleneck, which could be further improved depending on the given hardware and available resources.

CHAPTER 6

Conclusion

This work investigated how data-driven and model-aware sampling strategies can be used in an interactive, iterative labeling and discovery loop for unstructured data. The main goal was to analyze which strategies achieve the best balance between class discovery, model accuracy, and selection latency within a defined interaction budget. The key finding is that there is no overall best selection strategy. Rather, the usefulness of a sampling strategy depends on the complexity of the dataset, the imbalance within the data, the learner, and the phase of the iterative process. The results show that random sampling can be competitive on simple and balanced datasets, especially when the classes of the data are easy to separate. However, its limitations become more apparent with complex or imbalanced datasets, because rare classes are much harder to identify. In such cases, more sophisticated sampling strategies gain importance. Data-driven methods such as farthest sampling and D^{α} sampling are well-suited for early exploration and fast class discovery, while model-aware methods such as min-margin sampling and disagreement sampling are better suited for later refinement and offer a beneficial trade-off between accuracy and latency. This supports the idea that exploration and optimization should not be viewed as competitive alternatives, but rather as complementary phases of the same iterative process.

The experiments further demonstrate that the interaction parameters affect the loop in different ways. Increasing the panel size M does not consistently improve the accuracy but increases the latency, especially for the data-driven selectors. Conversely, increasing the batch size k improves class discovery and accuracy more but also increases computational overhead. Therefore, moderate batch sizes appear to represent a reasonable compromise between information gain and interactivity. GPU acceleration is also crucial for the usability of the loop, with computationally intensive selection and repeated learner updates. It reduces the latency without causing a significant loss of accuracy.

Several limitations must be considered when interpreting the results. The results are highly dependent on the quality of the embeddings. A good embedding can reduce the

visible differences between selectors, while weaker embeddings can make the choice of selector more important. Furthermore, the evaluation simulates an idealized, error-free user interaction with perfect ground-truth labeling and only approximates real-world user interfaces (UIs) via a panel protocol. A key limitation is the restriction of the candidate pool to a locally visible panel of size M. While randomly populating this panel is an intentional design choice to avoid visual overdrawing and preserve global density and topological context for the analyst, it artificially constrains the sampling algorithms. This means that the selectors cannot access or query the most globally informative or uncertain samples in the entire unlabeled dataset. Because algorithmic performance is affected by this local constraint, the results primarily apply to the sampling selectors under the idealized conditions of the pipeline with the same panel size, known labels, and fixed panel protocol and are not directly comparable to traditional, unconstrained active learning setups. Real-world user behavior, labeling errors, cognitive load, and UI effects are therefore not fully represented. Additionally, the scalability to very large image datasets is limited by computational and storage costs, and reproducibility depends on the implementation details such as stable panel indices and consistent experimental seed values. The experiments also do not analyze the diversity and stability of selected samples across iterations. Another limitation is the role of the learner in the simulation. The same learner is used as the model state for model-aware selection and as a metric for the accumulated knowledge. This coupling improves reproducibility and isolates the effect of the sampling strategies, but it produces results that reflect idealized conditions. In a real system, the user's model and the evaluation model would not necessarily be identical.

Future work should therefore focus on adaptive selection mechanisms that explicitly switch between exploration and optimization during the iterative process. The hybrid selectors, which are tested in this work, already suggest that combining complementary selection strategies can be advantageous, but further systematic investigations are needed. Future research should also include studies with real users to better understand the differences between the quality of the selection and the human perception of informative or interesting samples. Furthermore, more complex datasets, more realistic imbalanced scenarios, and noisy labels should be evaluated. A deeper analysis of selection diversity, stability, and minority class behavior would also contribute to a better understanding of why certain selectors perform better within the loop.

In summary, this work demonstrates that iterative sampling is a promising approach for class discovery in unstructured image data when accuracy, class discovery rate, and latency are considered together. More sophisticated sampling strategies offer significant advantages over random selection under more challenging conditions. However, these advantages only become apparent if the underlying pipeline remains computationally efficient. This work combines methodological insights into sampling strategies with the practical requirements of iterative systems and creates a basis for further research at the interface of visual analytics, user-centered learning, and class discovery.

APPENDIX A

Appendix

A.1 Undersampled Classes in CIFAR-100 and TinyImageNet

ID	Class	ID	Class	ID	Class
0	apple	17	castle	34	fox
1	aquarium_fish	18	caterpillar	35	girl
2	baby	19	cattle	36	hamster
3	bear	20	chair	37	house
4	beaver	21	chimpanzee	38	kangaroo
5	bed	22	clock	39	computer_keyboard
6	bee	23	cloud	40	lamp
7	beetle	24	cockroach	41	lawn_mower
8	bicycle	25	couch	42	leopard
9	bottle	26	crab	43	lion
10	bowl	27	crocodile	44	lizard
11	boy	28	cup	45	lobster
12	bridge	29	dinosaur	46	man
13	bus	30	dolphin	47	maple_tree
14	butterfly	31	elephant	48	motorcycle
15	camel	32	flatfish	49	mountain
16	can	33	forest		

Table A.1: CIFAR-100 classes undersampled in the imbalanced setting. The minority classes correspond to labels 0–49.

ID	WNID	Class	ID	WNID	Class
0	n02124075	Egyptian cat	50	n02823428	beer bottle
1	n04067472	reel	51	n02236044	mantis
2	n04540053	volleyball	52	n03393912	freight car
3	n04099969	rocking chair	53	n07583066	guacamole
4	n07749582	lemon	54	n04074963	remote control
5	n01641577	bullfrog	55	n01629819	European fire salamander
6	n02802426	basketball	56	n09332890	lakeside
7	n09246464	cliff	57	n02481823	chimpanzee
8	n07920052	espresso	58	n03902125	pay-phone
9	n03970156	plunger	59	n03404251	fur coat
10	n03891332	parking meter	60	n09193705	alp
11	n02106662	German shepherd	61	n03637318	lampshade
12	n03201208	dining table	62	n04456115	torch
13	n02279972	monarch butterfly	63	n02666196	abacus
14	n02132136	brown bear	64	n03796401	moving van
15	n04146614	school bus	65	n02795169	barrel
16	n07873807	pizza	66	n02123045	tabby cat
17	n02364673	guinea pig	67	n01855672	goose
18	n04507155	umbrella	68	n01882714	koala
19	n03854065	organ	69	n02917067	bullet train
20	n03838899	oboe	70	n02988304	CD player
21	n03733131	maypole	71	n04398044	teapot
22	n01443537	goldfish	72	n02843684	birdhouse
23	n07875152	potpie	73	n02423022	gazelle
24	n03544143	hourglass	74	n02669723	academic gown
25	n09428293	seashore	75	n04465501	tractor
26	n03085013	computer keyboard	76	n02165456	ladybug
27	n02437312	Arabian camel	77	n03770439	miniskirt
28	n07614500	ice cream	78	n02099601	golden retriever
29	n03804744	nail	79	n04486054	triumphal arch
30	n04265275	space heater	80	n02950826	cannon
31	n02963159	cardigan	81	n03814639	neck brace
32	n02486410	baboon	82	n04259630	sombrero
33	n01944390	snail	83	n03424325	gasmask
34	n09256479	coral reef	84	n02948072	candle
35	n02058221	albatross	85	n03179701	desk
36	n04275548	spider web	86	n03400231	frying pan
37	n02321529	sea cucumber	87	n02206856	bee
38	n02769748	backpack	88	n03160309	dam
39	n02099712	Labrador retriever	89	n01984695	spiny lobster
40	n07695742	pretzel	90	n03977966	police van
41	n02056570	king penguin	91	n03584254	iPod
42	n02281406	sulphur butterfly	92	n04023962	punching bag
43	n01774750	tarantula	93	n02814860	lighthouse
44	n02509815	red panda	94	n01910747	jellyfish
45	n03983396	pop bottle	95	n04596742	wok
46	n07753592	banana	96	n03992509	potter's wheel
47	n04254777	sock	97	n04133789	sandal
48	n02233338	cockroach	98	n03937543	pill bottle
49	n04008634	projectile	99	n02927161	butcher shop

Table A.2: TinyImageNet classes undersampled in the imbalanced setting. The minority classes correspond to labels 0–99, where labels are assigned according to the order of entries in `wnids.txt`.

A.2 Complete Overview of CLI Parameters

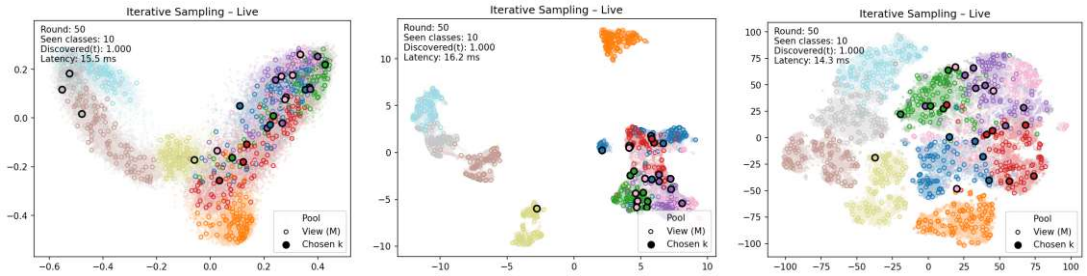
The experiments are steered by the command line interface (CLI) parameters, which allow for reproducible configuration of the core components of the iterative loop. For the actual comparison experiments, the parameters *–dataset, –M, –k, –r, and –T* are particularly relevant because they define the dataset, the panel size, the label budget, the update mechanism and the number of rounds. Furthermore, *–strategy, –learner, and –embed* determine the sampling strategy, learner, and the embedding space used. For the selector-specific D^{α} values, the *–dalpha_alpha* setting is important. The randomness of the experiments is controlled by *–seed*, while *–backend* distinguishes between CPU- and GPU-based execution. For the scenarios that are tested throughout the work, additional parameters are used, which are *–make_imbalanced*, to generate imbalanced datasets, and *memory_limit*, to limit the training dataset using a coreset. Parameters such as *–unseen_only and –sample_from_view_only* determine whether only the previously unseen points are selected and whether the selection is limited to the currently visible panel. The *–unseen_only and –sample_from_view_only* are always set to true to make the experiments reproducible and narrow down the points to select.

In addition to the core parameters described in the experiment design, the implementation supports further options for visualization, fine-tuning the model and for data-dependent special cases. These include *–viz, –viz_neighbors, –viz_min_dist, and –viz_perplexity* for the two-dimensional projections. For kNN, the number of neighbors is controlled by the *–knn_k* parameter. For uncertainty-based settings, *–uncertainty_entropy* is an available command. Further, it is also possible to use *–allow_reselect and –reweight_reselect* for variants where re-selection is needed or desired. The options *–max_items, –tiny_root, –out and –live* are used for data limitation, path management, the storage of the results and live visualizations. These options serve here only for the technical implementation or as sub-studies and are therefore not essential for understanding the basic experimental setup.

A.3 Visual Comparison of PCA, UMAP, and t-SNE

The visual comparison is a snapshot of PCA, UMAP, and t-SNE output in a 2D embedding. It is the live view of the iterative sampling loop. Each dot is one data point, and the layout shows which points are similar or dissimilar, whether they are nearby or far away, in the panel. The entire dataset is plotted faintly so it forms the background map where the data lives. It is made transparent that the current subset can be inspected better visually. The colored hollow circles are the current subset at each round. These are hollow so the viewer is still able to see the background. The edge color represents the class label. The actually selected points per round are bigger and have a thick black outline so the sampling decisions can be tracked. The legend in the top left corner indicates that this snapshot is from round 50. All classes from FMNIST are discovered, which is why the discovery rate is at 1.0 and the latency for this selection step is also displayed.

Both UMAP and t-SNE are sensitive to hyperparameters. While UMAP has parameters like *n_neighbors/min_dist*, t-SNE has perplexity and learning rate. Changing those parameters before or by using the pipeline command can lead to the merging or splitting of obvious clusters.

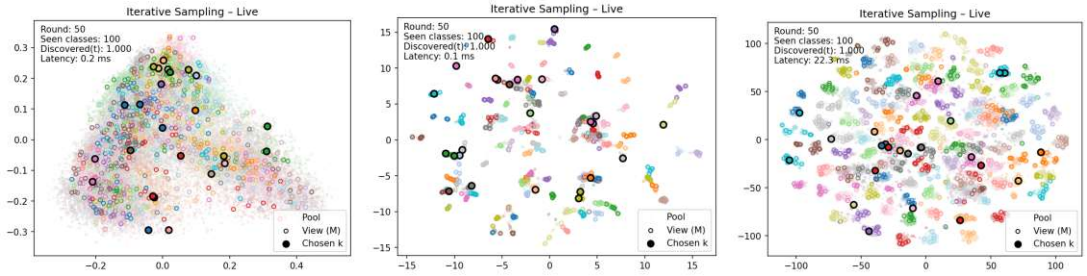


(a) PCA (b) UMAP (c) t-SNE

Figure A.1: FMNIST visualization comparison.

From the visual comparison, it can be observed that the PCA and UMAP plots are clearly distinguishable. In the PCA plot the data points of the different classes are more connected to each other and have less separation. The UMAP plot shows clearer separation between the classes, so they are more distinguishable. The reason for this difference is that PCA cannot deal with non-linearity like UMAP can, as it depends on linearity. Therefore, it has a much harder time to separate the classes because it cannot capture the non-linear nature of image data. Compared to UMAP, t-SNE makes the plot appear more like islands and more separated from each other: Many classes form denser, rounder blobs with more empty space between them, and the large mass on the right side in UMAP is pulled apart into mostly clearly separated subclusters. It also spreads groups over a much larger area, making gaps and cluster boundaries more visually prominent.

For the CIFAR-100 dataset, a similar pattern can be observed. In Figure A.2, the left picture shows that all data points form one large undifferentiated cluster without any possibility of distinction. It is impossible to derive a certain location of a class and



(a) PCA (b) UMAP (c) t-SNE

Figure A.2: CIFAR-100 visualization comparison.

therefore is not sufficiently useful for a user to inspect. UMAP is much more suitable for using as a visualization because the user can derive where each class is if enough colors are available or the visualization is actually presented with pictures. Here the distinctions between classes are clearly visible. For comparing t-SNE with UMAP again, it can be observed that the classes are clearly separated and each class forms its own bulk. There are almost no connections to other classes, which is similar to the FMNIST plot before. The classes are spread more equally over the whole plot.

However, this could only be a spacing effect from UMAP. One has to be careful analyzing this image because it might be overloaded due to the 100 classes the dataset contains. When projecting the embeddings into two dimensions, many areas could be collapsed into many neighborhoods that are on the same area and produce the apparent mixing. With 100 colors and thousands of points, the occlusion and color repetition do not help to increase interpretability. Therefore, it might be more useful to present fewer classes in a visualization. Visualizing a random subset or reducing CIFAR-100 to its 20 superclasses might be useful strategies to prevent visual overload. Another possibility could be to highlight only the current samples or selected points. This also reduces the color clutter.

Overall, PCA produces a largely continuous structure with significant overlap between classes, which reflects its linear nature. UMAP provides more clearly separated areas and more interpretable organization of neighborhoods, which is consistent with the nonlinear structure expected in image embeddings. Compared to UMAP, t-SNE places even more emphasis on preserving local neighborhoods and often produces visually appealing and distinct islands with large empty gaps. UMAP and t-SNE both capture the non-linear nature of the data well, with one advantage of UMAP over t-SNE being that it requires less computational time. To sum up, UMAP and t-SNE are suitable as qualitative tools for analyzing local structure and sampling behavior, whereas the absolute separability of classes should not be taken for granted.

A.4 GPU Acceleration

In this additional section, the comparison of CPU versus GPU for CIFAR-100 is shown with the logistic regression learner. Similar to the FMNIST results, the GPU reduces the total time for fit, selection and prediction by a large margin, while the accuracy stays the same.

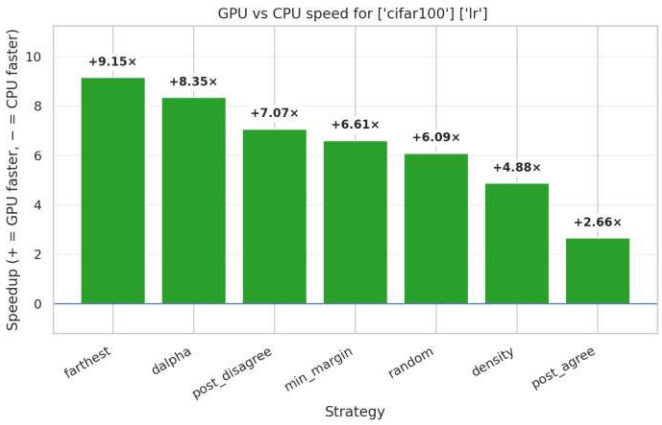
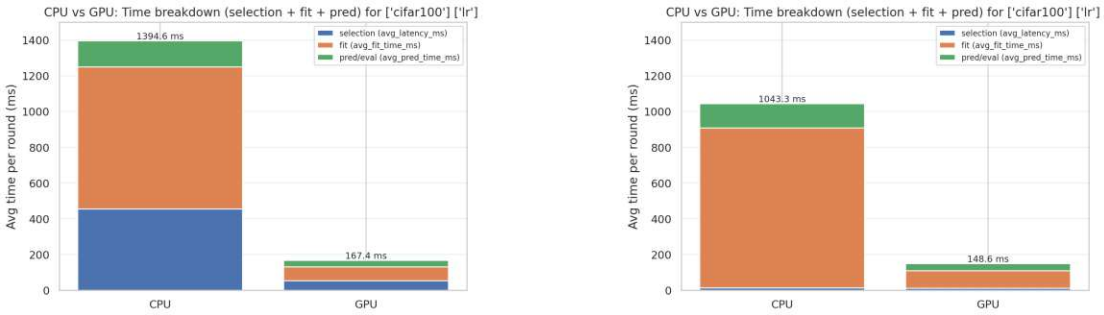


Figure A.3: GPU vs CPU speedup comparison on CIFAR-100.

It can be observed that the highest factor is here even over +9, while the smallest gain is for the Agreement selector with +2.66. The overall performance shows that the difference is bigger than for the FMNIST dataset. To further analyze the different methods. D^{α} for data-driven and disagreement for model-aware are evaluated for the differences in selection, fitting, and prediction.



(a) D^{α} sampling

(b) Disagreement sampling

Figure A.4: Time breakdown of each stage for CIFAR-100.

Here the fitting consumes more more time than selection and prediction. The most gain can be derived from using GPU acceleration for fitting the model. But also the decrease in selection time for D^{α} sampling has to be considered meaningful. Therefore, it can be assumed that further evaluation on TinyImageNet might not be useful because the whole

time consumption already takes over 1 second, even though the sampling is still below 500 ms. However, considered it is the most complex dataset it is not advisable to use the CPU-based iterative process on this particular dataset.

Übersicht verwendeter Hilfsmittel

ChatGPT was used as a language-support tool to improve the clarity, fluency, and stylistic quality of English sentences written by myself. It was not used to generate the scientific content, analysis, or results of this thesis. The responsibility for the final text and thoughts remains entirely with the author.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AA20] Jordan T. Ash and Ryan P. Adams. On warm-starting neural network training, 2020.

[AAA+25] Naif Alatrush, Sultan Alsarra, Afraa Alshammari, Luay Abdeljaber, Niamat Zawad, Latifur Khan, Patrick Brandt, Javier Osorio, and Vito D'Orazio. Advancing active learning with ensemble strategies. pages 57–66, 01 2025.

[AGH+23] Shehzad Afzal, Sohaib Ghani, Mohamad Mazen Hittawe, Sheikh Faisal Rashid, Omar M. Knio, Markus Hadwiger, and Ibrahim Hoteit. Visualization and visual analytics approaches for image and video datasets: A survey. *ACM Trans. Interact. Intell. Syst.*, 13(1), March 2023.

[AML+22] Aazad Abbas, Jacob Mosseri, Johnathan R. Lex, Jay Toor, Bheeshma Ravi, Elias B. Khalil, and Cari Whyne. Machine learning using preoperative patient factors can predict duration of surgery and length of stay for total knee arthroplasty. *International Journal of Medical Informatics*, 158:104670, 2022.

[ASSS18] Marco Angelini, Giuseppe Santucci, H. Schumann, and Hans-Jörg Schulz. A review and characterization of progressive visual analytics. *Informatics*, 5:31, 07 2018.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[AWV+19] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[BABEM24] Satyanarayana Bonakala, Michael Aupetit, Halima Bensmail, and Fedwa El Mellouhi. A human-in-the-loop approach for visual clustering of overlapping materials science data. *Digital Discovery*, 3, 02 2024.

[BDW19] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families, 2019.

[Ber25] Jürgen Bernard. The human-data-model interaction canvas for visual analytics, 2025.

[BHS25] Nick Barney, Katie Terrell Hanna, and Craig Stedman. What is unstructured data?, 3 2025.

[BHZ+17] Jurgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):298–308, August 2017.

[BLBC12] Eli T. Brown, Jingjing Liu, Carla E. Brodley, and Remco Chang. Disfunction: Learning distance functions interactively. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92, 2012.

[BNS23] Etienne Bamas, Sai Ganesh Nagarajan, and Ola Svensson. An analysis of d^α seeding for k-means, 2023.

[BZSA18] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. Vial: a unified process for visual interactive labeling. *Vis. Comput.*, 34(9):1189–1207, September 2018.

[CBB+19] Mohammad Chegini, Jürgen Bernard, Philip Berger, Alexei Sourin, Keith Andrews, and Tobias Schreck. Interactive labelling of a multivariate dataset for supervised machine learning using linked visualisations, clustering, and active learning. *Visual Informatics*, 3(1):9–17, 2019. Proceedings of Pacific-VAST 2019.

[CLL+16] Adrian Calma, Jan Marco Leimeister, Paul Lukowicz, Sarah Oeste-Reiss, Tobias Reitmaier, Albrecht Schmidt, Bernhard Sick, Gerd Stumme, and Katharina Anna Zweig. From active learning to dedicated collaborative interactive learning. In *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, pages 1–8, 2016.

[DE02] Alan Dix and Geoffrey Ellis. By chance enhancing interaction with large data sets through statistical sampling. *Proceedings of the Workshop on Advanced Visual Interfaces AVI*, 05 2002.

[DK18] John J. Dudley and Per Ola Kristensson. A review of user interface design for interactive machine learning. *ACM Trans. Interact. Intell. Syst.*, 8(2), June 2018.

[DTAA25] Kevin Delcourt, Sylvie Trouilhet, Jean-Paul Arcangeli, and Françoise Adreit. The human in interactive machine learning: Analysis and perspectives for ambient intelligence. *J. Artif. Int. Res.*, 81, January 2025.

[EBD05] Geoffrey Ellis, Enrico Bertini, and Alan Dix. The sampling lens: making sense of saturated visualisations. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, page 1351–1354, New York, NY, USA, 2005. Association for Computing Machinery.

[EFN12] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 473–482, New York, NY, USA, 2012. Association for Computing Machinery.

[EHH12] D. Engel, L. Hüttenberger, and Bernd Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. *OpenAccess Series in Informatics*, 27:135–149, 01 2012.

[ERT+17] A. Endert, W. Ribarsky, C. Turkay, B.L. William Wong, I. Nabney, I. Díaz Blanco, and F. Rossi. The state of the art in integrating machine learning into visual analytics. *Computer Graphics Forum*, 36(8):458–486, 2017.

[Fri37] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[GCT22] Benedikt Grimmeisen, Mohammad Chegini, and Andreas Theissler. Visgil: machine learning-based visual guidance for interactive labeling. *The Visual Computer*, 39, 09 2022.

[GDE24] Edrina Gashi, Jiankang Deng, and Ismail Elezi. Deep active learning: A reality check, 2024.

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[HCL+24] Zitong Huang, Ze Chen, Yuanze Li, Bowen Dong, Erjin Zhou, Yong Liu, Rick Siow Mong Goh, Chun-Mei Feng, and Wangmeng Zuo. Class balance matters to active class-incremental learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 9445–9454. ACM, October 2024.

[HDG+21] Rishi Hazra, Parag Dutta, Shubham Gupta, Mohammed Abdul Qaathir, and Ambedkar Dukkipati. Active^2 learning: Actively reducing redundancies in active learning methods for sequence tagging and machine translation, 2021.

[HHN+24] Denis Huseljic, Marek Herde, Yannick Nagel, Lukas Rauch, Paulius Strimaitis, and Bernhard Sick. The interplay of uncertainty modeling and deep active learning: An empirical analysis in image classification. *Trans. Mach. Learn. Res.*, 2024, 2024.

[HLL+22] Chengming Hu, Xuan Li, Dan Liu, Xi Chen, Ju Wang, and Xue Liu. Teacher-student architecture for knowledge learning: A survey, 2022.

[HLM+24] Arthur Hoarau, Vincent Lemaire, Arnaud Martin, Jean-Christophe Dubois, and Yolande Le Gall. Evidential uncertainty sampling for active learning, 2024.

[Hol25] Isac Holm. Vilod: A visual interactive labeling tool for object detection, 2025.

[HVZ19] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering, 2019.

[JLK+25] Hyeon Jeon, Hyunwook Lee, Yun-Hsin Kuo, Taehyun Yang, Daniel Archambault, Sungahn Ko, Takanori Fujiwara, Kwan-Liu Ma, and Jinwook Seo. Unveiling high-dimensional backstage: A survey for reliable visual analytics with dimensionality reduction. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery.

[JPA+22] K. J. Joseph, Sujoy Paul, Gaurav Aggarwal, Soma Biswas, Piyush Rai, Kai Han, and Vineeth N. Balasubramanian. Novel class discovery without forgetting. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, page 570–586, Berlin, Heidelberg, 2022. Springer-Verlag.

[KMSC16] Matthieu Komorowski, Dominic Marshall, Justin Salciccioli, and Yves Crutain. *Exploratory Data Analysis*, pages 185–203. 09 2016.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[KVHD17] Bum Chul Kwon, Janu Verma, Peter J. Haas, and Cagatay Demiralp. Sampling for scalable visual analytics. *IEEE Computer Graphics and Applications*, 37(1):100–108, 1 2017.

[LHW+22] Peng Liu, Guojin He, Lizhe Wang, Rajiv Ranjan, and Lei Zhao. A survey on active deep learning: From model-driven to data-driven. *ACM Computing Surveys*, 54, 02 2022.

[LJHW23] Harry X. Li, Steven Jorgensen, John Holodnak, and Allan B. Wollaber. Scatteruq: Interactive uncertainty visualizations for multiclass deep learning problems. In *2023 IEEE Visualization and Visual Analytics (VIS)*, page 246–250. IEEE, October 2023.

[LMW+17] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, March 2017.

[LSL+25] Yi-Chi Liao, Paul Streli, Zhipeng Li, Christoph Gebhardt, and Christian Holz. Continual human-in-the-loop optimization. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, page 1–26. ACM, April 2025.

[LY15] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[MMIS22] Michal Motylinski, Áine MacDermott, Farkhund Iqbal, and Babar Shah. A gpu-based machine learning approach for detection of botnet attacks. *Computers & Security*, 123:102918, 2022.

[MSB+25] Matthias Matt, Jana Sedlakova, Jürgen Bernard, Matthias Zeppelzauer, and Manuela Waldner. Scalable class-centric visual interactive labeling. *Computers & Graphics*, 129:104240, 2025.

[MZW24] Matthias Matt, Matthias Zeppelzauer, and Manuela Waldner. cvil: Class-centric visual interactive labeling, 2024.

[NGH+25] Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, Hsien-chin Lin, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, and Milica Gašić. A confidence-based acquisition model for self-supervised active learning and label correction. *Transactions of the Association for Computational Linguistics*, 13:167–187, 2025.

[ODM+24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[PRL+25] Utkarsh Pratiush, Kevin M. Roccapriore, Yongtao Liu, Gerd Duscher, Maxim Ziatdinov, and Sergei V. Kalinin. Building workflows for an interactive human-in-the-loop automated experiment (hae) in stem-eels. *Digital Discovery*, 4:1323–1338, 2025.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[RKW+24] Benedikt Roth, Valentin Koch, Sophia J. Wagner, Julia A. Schnabel, Carsten Marr, and Tingying Peng. Low-resource finetuning of foundation models beats state-of-the-art in histopathology. In *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2024.

[RL14] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[RXC+21] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2021.

[Set10] Burr Settles. Active learning literature survey. 07 2010.

[Set11] Burr Settles. From theories to queries: Active learning in practice. In Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov, editors, *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, volume 16 of *Proceedings of Machine Learning Research*, pages 1–18, Sardinia, Italy, 16 May 2011. PMLR.

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, 1948.

[SRHA23] Marina Sánchez-Rico, Nicolas Hoertel, and Jesús M Alvarado. Combination of cluster analysis with dimensionality reduction techniques for pattern recognition studies in healthcare data: Comparing pca, t-sne and umap, Jan 2023.

[SS18] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[SYM+24] Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, and Jose M. Alvarez. Step out and seek around: On warm-start training with incremental data, 2024.

[VHVZ22] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery, 2022.

[WXS+22] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381, 2022.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[YGW+15] Dong-Ming Yan, Jianwei Guo, Bin Wang, Xiaopeng Zhang, and Peter Wonka. A survey of blue-noise sampling and its applications. *Journal of Computer Science and Technology*, 30:439–452, 05 2015.

[YK10] Hwanjo Yu and Sungchul Kim. Passive sampling for regression. In *2010 IEEE International Conference on Data Mining*, pages 1151–1156, 2010.

[ZA23] Bingchen Zhao and Oisin Mac Aodha. Incremental generalized category discovery, 2023.

[ZGC+17] Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, and Tim Kraska. How progressive visualizations affect exploratory analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1977–1987, 2017.